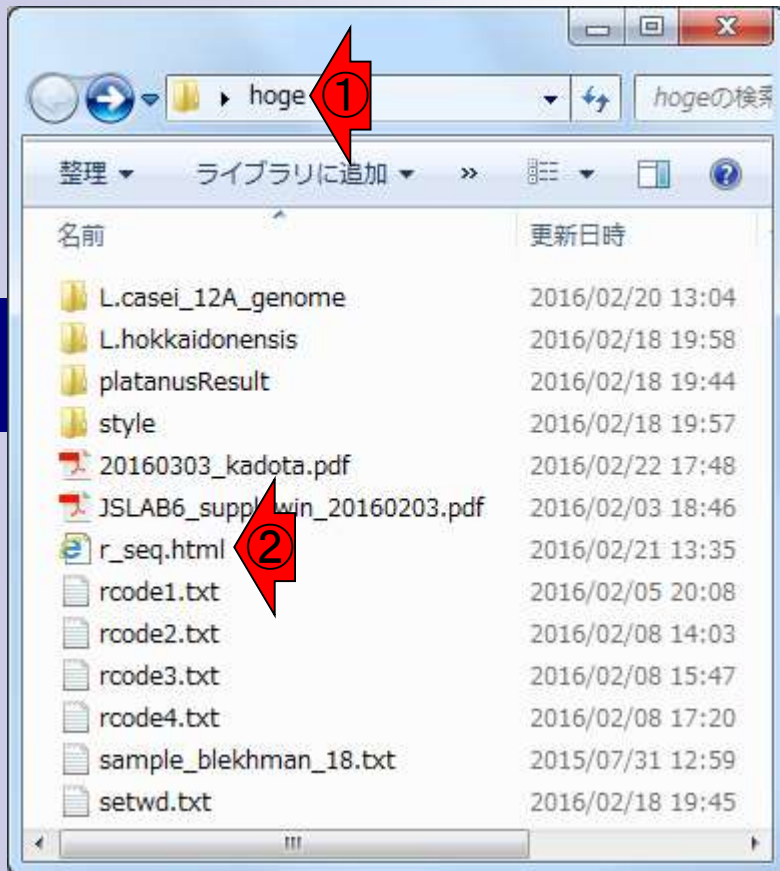


2016.03.05版

実習用PCのデスクトップ上に、①hogeフォルダがあります。この中に解析に必要な入力ファイルがあります。ネットワーク不具合時は、②ローカル環境でhtmlファイルを起動して各自対応してください。



## Rで塩基配列解析：ゲノム解析からトランスクリプトーム解析まで

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

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/>

# 自己紹介

## 学歴および職歴

- 2002年3月 東京大学・大学院農学生命科学研究科 博士課程修了
- 2002年4月 産業技術総合研究所・CBRC
- 2003年11月 放射線医学総合研究所・先端遺伝子発現研究センター
- 2005年2月～ 東京大学・大学院農学生命科学研究科  
 アグリバイオインフォマティクス人材養成プログラム(科学技術振興調整費: 2004/10-2009/3)  
 アグリバイオインフォマティクス教育研究プログラム(特別教育研究経費: 2009/4~2014/3)

## アグリバイオインフォマティクス教育研究プログラム

- 他大学の学生や社会人も受講できる、希少なバイオインフォ教育プログラム

年度	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
修士課程	12	65	73	83	68	72	107	100	121	124	108	151
博士課程	3	7	11	13	6	8	12	21	16	19	24	25
社会人	5	3	8	4	1	0	11	19	32	26	55	34
<b>合計</b>	<b>20</b>	<b>75</b>	<b>92</b>	<b>100</b>	<b>75</b>	<b>80</b>	<b>130</b>	<b>140</b>	<b>169</b>	<b>169</b>	<b>187</b>	<b>210</b>
開講科目数	9	15	15	15	15	12	15	15	14	15	13	13
常勤教員数	6	6	7	7	7	3	4	4	3	2	2	2
ポスドク数	>2	>2	>2	>2	>2	1	1	1	1	1	1	0
門田担当コマ数	3	?	?	8	8	5	5	11	13	14	18	20

1科目以上の合格者数



少数のスタッフで行っているアグリバイオの活動のみで基本的に手一杯。ここ数年でさらに「研究 << 教育」のヒトに…。現在、研究は片手間以下。  
 ①限界以下のスタッフ数でアグリバイオの本務を行っているため、精神状態をなるべく平静に保つべく、優先順位の低い活動には関与しません。

基本スタンスは、優先順位とエフォート。基本独裁、一匹狼、ロビー活動なし、門田教への勧誘なし、信者になっても(オールフリー派なのでw)メリットゼロ。受益者が金と時間をかけずに効率的に学べる教材整備が最優先。

# 主な活動

- 東大アグリバイオの大学院講義(バイオインフォ全般)
  - Rを中心としたハンズオン講義(平成16年度～)
    - 受講人数が多い(最大130名)ので、クラウド(ウェブツール)系実習は実質的に不可能
    - 講義補助員(TA)が数名のみなので、Linux系実習も困難
- NBDC/東大アグリバイオ/HPCIのNGSハンズオン講義(NGSに特化)
  - Linuxを中心としたハンズオン講義(平成26年度～)
    - 受講人数は多い(最大71名;おそらくアグリバイオ本体に次ぐ規模)が、受講生の意識レベルが高く(きっちり予習をやるヒトが多数派)、環境構築済みノートPC数、TA数が充実しているため、本格的なLinux実習が成立しうる。
- 日本乳酸菌学会誌のNGS連載
  - Linuxを中心とした自習用教材(平成26年度～)
    - バクテリア(乳酸菌)データを、主にBio-Linux上で解析するノウハウを提供。
    - 第6回(2016年3月予定)分以降は、DDBJ Pipeline(ウェブツール)の利用法も紹介。
    - データ取得・インストール・実行に時間がかかるものも、自習なので時間を気にせずにできる。ハンズオン講義よりも心穏やか。
- その他
  - 研究(発現変動解析精度向上のためのアルゴリズム開発や評価)
  - HPCI講習会・バイオインフォマティクス実習コースの講師
    - 丸2日だが、上記の主要3項目に比べれば心穏やか

# Contents1

## ■ イン트로ダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# (Rで)塩基配列解析

①2013年秋以降の講義資料や連載原稿のPDFを簡単な解説つきで公開。講義資料系は、1年以上昔のものは参考程度。ウェブサイトが見つらいとか見栄えに関する要望は無視(優先順位が閾値以下)

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2016/01/29, since 2011)

### 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 ([Windows2015.04.04版](#)と[Macintosh2015.04.03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)
- 多群間比較用の推奨ガイドライン提唱論文([Tang et al., BMC Bioinformatics, 2015](#))がpublishされました。論文概要については[門田](#)のページでも紹介しています。講習会でよく述べている「サンプル間クラスタリング結果からDEG検出結果のおおよその見積もりが可能である」という主張の根拠となる原著論文がこれになります。推奨ガイドライン周辺の関連項目もアップデートしました。(2015/11/05) **NEW**
- [日本乳酸菌学会誌](#)の[NGS関連連載](#)の第5回ウェブ資料を更新しました。2015年12月下旬に一気に全てやり直したので、若干プログラムのバージョンが上がっています。(2015/12/22)
- [解析 | 一般 | アライメント || について](#)を追加しました。(2015/12/16)
- [日本乳酸菌学会誌](#)の[NGS関連連載](#)の第4回ウェブ資料を更新しました。2015年12月初旬に一気に全てやり直したので、若干プログラムのバージョンが上がっています。各回終了時点のovaファイル(約6GB)も提供可能です。(権利関係上無条件公開はできませんので...)欲しい方は、メールのタイトルを「乳酸菌連載第x回終了時点のovaファイル希望」として私宛にメールしてください(本文は空でOK)。URLをお知らせします。(2015/12/11)

- [はじめに](#) (last modified 2015/03/31)
- [参考資料\(講義、講習会、本など\)](#) **①** modified 2015/11/17
- [過去のお知らせ](#) (last modified 2015/12/22)
- [インストール || について](#) (last modified 2015/11/12)
- [インストール || について](#) (last modified 2015/03/22) 推奨

[トップページへ](#)

# (Rで)塩基配列解析

Linux系の教材。日本乳酸菌学会誌のNGS連載。①第4回、②第5回。第6回は2016年3-4月ごろ公開予定

## 参考資料(講義、講習会、本など)

基本的に私門田の個人ページに記載してあるものです。かなり古い講演資料などの情報をもとに勉強されている方もいらっしゃるようですので、ここでは2013年秋以降の情報を載せておくとともに、大まかな内容についても述べておきます。講演予定のものについては、資料のアップは講演当日が基本です。50-100MB程度ありますがオリジナルのPowerPointファイルがほしい方はお気軽にリクエストしてください。講義資料としての利用などは事前連絡や私個人への謝辞も気にせずご自由にお使いください。

## 書籍、学会誌

- 孫建強, 清水謙多郎, 門田幸二, 「次世代シーケンサーデータの解析手法: 第5回アセンブル、マッピング、そしてQC」, [日本乳酸菌学会誌](#), 26(3):193-201, 2015.  
内容: 日本乳酸菌学会誌のNGS関連連載の第5回分です。ゲノムアセンブリ周辺の概説。FaQCs (ver. 1.34)によるQC。トランスクリプトームアセンブリ周辺の概説。バクテリア専用アセンブラRockhopper 2のインストールと利用。(Bio-Linux上での)Rの基本的な利用法とパッケージのインストール。第1回で述べた「Rでゲノム解析」のLinux版を説明。QuasR (ver. 1.8.4)での乳酸菌RNA-seqデータの乳酸菌リファレンスゲノムへのマッピング。ほとんどマップされなかった原因の発見とその解決策。問題部分のトリム、およびマッピングとアセンブルの改善例を示す。(2015年11月17日時点で、Trinityのほうがアセンブル結果がよさそうであることがわかってきたので、第6回原稿中でそのあたりを述べる予定です。) 書籍中のリンク先やウェブ資料などは「書籍 | 日本乳酸菌学会誌 | [第5回アセンブル、マッピング、そしてQC](#)」の項目をご覧ください。
- 孫建強, 湯敏, 清水謙多郎, 門田幸二, 「次世代シーケンサーデータの解析手法: 第4回クオリティコントロールとプログラムのインストール」, [日本乳酸菌学会誌](#), 26(2):124-132, 2015.  
内容: 日本乳酸菌学会誌のNGS関連連載の第4回分です。p131 右 第2段落で間違いを発見しました。「FastQC (ver. 0.11.1)」は「FastQC (ver. 0.11.3)」です。2015年7-8月に開催されたNGSハンズオン講習会の後半ごろから共有フォルダ設定がリセットされるという不具合が生じたため、p.128 左 16行目付近の共有フォルダ設定周辺のウェブ資料内容を2015年8月23日にアップデートしました。書籍中のリンク先やウェブ資料などは「書籍 | 日本乳酸菌学会誌 | [第4回クオリティコントロールとプログラムのインストール](#)」の項目をご覧ください。
- 孫建強, 三浦文, 清水謙多郎, 門田幸二, 「次世代シーケンサーデータの解析手法: 第3回Linux環境構築からNGSデータ取得まで」, [日本乳酸菌学会誌](#), 26(1):32-41, 2015.  
内容: 日本乳酸菌学会誌のNGS関連連載の第3回分です。p.32 右 1行目に「WinとMacのLinux環境の違い」とありますが、「WinとMacの"VM上"の"Linux環境の違い」としたほうが正確だったかもしれず、書籍中のリンク先やウェブ資料などは「書籍 | 日本乳酸菌学会誌 | [第3回Linux環境構築からNGSデータ取得ま](#)



# (Rで)塩基配列解析

①2014年4月刊行のR本。トランスクリプトーム解析全般の基礎知識的なところは、この本の第1章をご覧ください。

困り果てた配列を読み込んでGC含量をコピーで得られることなど。書籍中のリンク先は「書籍 | 日本乳酸菌学会誌 | 第1回イントロダクション」の項目をご覧ください。

- ・ 門田幸二著(金明哲 編), シリーズ Useful R 第7巻トランスクリプトーム解析, 共立出版, 2014. ISBN: 978-4-320-12370-0  
内容: マイクロアレイとRNA-seq解析を例としてRを用いてトランスクリプトーム解析を行うための体系的な本としてまとめました。数式が苦手なヒト向けに、重みつき平均の具体的な計算例などを挙げてオプションの意味などがわかるような中身の理解に重点を置いた構成にしています。書籍中のRコードは「書籍 | トランスクリプトーム解析 | ...」をご覧ください。
- ・ 門田幸二, 「トランスクリプトミクスの推奨データ解析ガイドライン」, ニュートリゲノミクスを基盤としたバイオマーカーの開発, シーエムシー出版, 45-52, 2013. ISBN: 978-4-7813-0820-3  
内容: マイクロアレイ解析の話がメインです。実験デザインの重要性を述べています。Affymetrix GeneChipデータの数値化と発現変動遺伝子(DEG)検出法の組合せの重要性の話や、サンプル間クラスターリングである程度DEGに関する情報がわかることを述べています。MAS5データを用いる場合は特に倍率変化で議論することも無意味であること、RMAのようなマルチアレイ正規化法を用いて得られたマイクロアレイデータの場合にはなぜ倍率変化でうまくいく傾向にあるかなどの理由をM-A plotを用いて説明しています。

講習会、講義、講演資料

- ・ 門田幸二, 寺田透, 三浦文, 清水謙多郎, 「ノートPCを用いたバイオインフォマティクス分野におけるハンズオン講義」, MBI研究会・第18回MBI研究発表会, 明治薬科大学(東京), 2015.11.06  
内容: アグリバイオインフォマティクス教育研究プログラムの紹介。カリキュラムや講義概要。特徴はフリーソフトウェアRを多くの講義で利用しているところ。保有PCの劣化は深刻。スタッフ数を増やすか、4年で完全にリリース可能な予算規模を確保しておく必要があるだろう。次世代シーケンサ(NGS)に特化したバイオインフォマティクス人材育成カリキュラムがNBDCIによって2014年に策定された。そのカリキュラムに基づいた講習会を平成26年9月に10日間かけて試行実施。好評?だったらしく、平成27年度も7-8月に計14日間かけて実施。特徴はハンズオンのみしたこと、講師の数を大幅に減らして連携を強化したこと、かなり厳しい予習を課したこと。相当な労力をかけて行っているため、受講生からの評価も高かったが、その分研究にかけられる時間は減る。どこまで研究者としての自分を犠牲にして"公共事業"にエフォートを費やすかは難しいところ。。40 min分。
- ・ 門田幸二, 「ビッグデータ解析の一例としてのトランスクリプトーム解析とその周辺 (2015.10.22版)」, 日本臨床麻酔学会・第35回大会, パシフィコ横浜(神奈川), 2015.10.21  
内容: ビッグデータといえばNGS, DDBJの利用を推奨。DBCLS SRAでNGSデータのトレンドを概観。Youtube



# (Rで)塩基配列解析

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

関与する配列を読み込んでGC含量をコピーで得られることなど。書籍中のリンク先やRコードは「書籍 | 日本乳酸菌学会誌 | 第1回イントロダクション」の項目をご覧ください。

- ・ 門田幸二著(金明哲 編), シリーズ Useful R 第7巻トランスクリプトーム解析, 共立出版, 2014. ISBN: 978-4-320-12370-0  
**内容:** マイクロアレイとRNA-seq解析を例としてRを用いてトランスクリプトーム解析を行うための体系的な本としてまとめました。数式が苦手なヒト向けに、重みつき平均の具体的な計算例などを挙げてオプションの意味などがわかるような中身の理解に重点を置いた構成にしています。書籍中のRコードは「書籍 | トランスクリプトーム解析 | ...」をご覧ください。
- ・ 門田幸二, 「トランスクリプトミクスの推奨データ解析ガイドライン」, ニュートリゲノミクスを基盤としたバイオマーカーの開発, シーエムシー出版, 45-52, 2013. ISBN: 978-4-7813-0820-3  
**内容:** マイクロアレイ解析の話がメインです。実験デザインの重要性を述べています。Affymetrix GeneChipデータの数値化と発現変動遺伝子(DEG)検出法の組合せの重要性の話や、サンプル間クラスタリングである程度DEGに関する情報がわかることを述べています。MAS5データを用いる場合は特に倍率変化で議論することも無意味であること、RMAのようなマルチアレイ正規化法を用いて得られたマイクロアレイデータの場合にはなぜ倍率変化でうまくいく傾向にあるかなどの理由をM-A plotを用いて説明しています。

講習会、講義、講演資料 ①

- ・ 門田幸二, 寺田透, 三浦文, 清水謙多郎, 「ノートPCを用いたバイオインフォマティクス分野におけるハンズオン講義」, MBI研究会・第18回MBI研究発表会, 明治薬科大学(東京), 2015.11.06 ←  
**内容:** アグリバイオインフォマティクス教育研究プログラムの紹介。カリキュラムや講義概要。特徴はフリーソフトウェアRを多くの講義で利用しているところ。保有PCの劣化は深刻。スタッフ数を増やすか、4年で完全にリリース可能な予算規模を確保しておく必要があるだろう。次世代シーケンサ(NGS)に特化したバイオインフォマティクス人材育成カリキュラムがNBDCIによって2014年に策定された。そのカリキュラムに基づいた講習会を平成26年9月に10日間かけて試行実施。好評?だったらしく、平成27年度も7-8月に計14日間かけて実施。特徴はハンズオンのみにしたこと、講師の数を大幅に減らして連携を強化したこと、かなり厳しい予習を課したこと。相当な労力をかけて行っているため、受講生からの評価も高かったが、その分研究にかけられる時間は減る。どこまで研究者としての自分を犠牲にして"公共事業"にエフォートを費やすかは難しいところ。。40 min分。
- ・ 門田幸二, 「ビッグデータ解析の一例としてのトランスクリプトーム解析とその周辺 (2015.10.22版)」, 日本臨床麻酔学会・第35回大会, パシフィコ横浜(神奈川), 2015.10.21 ← [トップページへ](#)  
**内容:** ビッグデータといえばNGS。DBJの利用を推奨。DBCLS SRAでNGSデータのトレンドを概観。Youtube



# アグリバイオ

東京大学大学院農学生命科学研究科

## アグリバイオインフォマティクス教育研究ユニット

Agricultural Bioinformatics Research Unit

+ サイトマップ + English



受講生の方へ



研究者の方へ

ホーム > 教育プログラム > 各講義のページ



### 各講義のページ

(科目名をクリックすると各講義のページに移動します)

先端トピックス セミナー・討論形式 研究指導	農学生命情報科学特別演習			
	農学生命情報科学特論 I	農学生命情報科学特論 II	農学生命情報科学特論 III	農学生命情報科学特論 IV
方法論 講義・実習を一体化	生物配列統計学	システム生物学概論	知識情報処理論	
	オーム情報解析	機能ゲノム学	分子モデリングと分子シミュレーション	
基礎 講義・実習を一体化	ゲノム情報解析基礎		構造バイオインフォマティクス基礎	
	生物配列解析基礎		バイオスタティスティクス基礎論	

科目名: 農学生命情報科学特論I  
 内容: 公共DB、チェックサム、QC、前処理、アセンブリ、マッピング、RPKM、発現変動など。  
 実施日: 2015.06.16、2015.06.23、2015.06.30、2015.07.07



科目名: 機能ゲノム学  
 内容: データ取得、正規化、クラスターリング、発現変動解析、多重比較問題、機能解析など。  
 実施日: 2015.05.12、2015.05.19、2015.05.26、2015.06.09



科目名: ゲノム情報解析基礎  
 内容: Rの基礎。GC含量計算やCpG解析、上流配列解析、Rのバージョンの違いなど。  
 実施日: 2015.04.07、2015.04.14、2015.04.21

# アグリバイオ

例えば、特論Iの第4回講義資料は、  
①講義資料をクリックすればよいが...

科目名: 農学生命情報科学特論I  
内容: 公共DB、チェックサム、QC、前処理、アセンブリ、マッピング、RPKM、発現変動など。  
実施日: 2015.06.16、2015.06.23、2015.06.30、2015.07.07

科目名: 機能ゲノム学  
内容: データ取得、正規化、クラスタリング、発現変動解析、多重比較問題、機能解析など。  
実施日: 2015.05.12、2015.05.19、2015.05.26、2015.06.09

科目名: ゲノム情報解析基礎  
内容: Rの基礎。GC含量計算やCpG解析、上流配列解析、Rのバージョンの違いなど。  
実施日: 2015.04.07、2015.04.14、2015.04.21

- 門田幸二、「[R: Bioconductorの利用法\(7/30分\)](#) (2015.07.31版)」、[バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.30  
内容: パッケージの説明。CRANとBioconductor。定期的なバージョンアップの重要性。インストール手順。Bioconductor概観。ゲノム配列パッケージ(BSgenome)。プロモーター配列取得(sessionInfo,バージョンの違い)FASTA形式ファイルとGFF3形式ファイル。データの型。FASTQファイルの各種解析(LinuxとRで活用)。FastQCとRパッケージ(ShortReadやQuasR)の比較。FastQC実行結果の各種項目(Overrepresented sequence、Per base sequence content、Kmer Content)の動作確認でRを利用など。4コマ(4×90 min)分。Youtubeと統合TV。
- 門田幸二、「[R: 基礎\(7/29分\)](#) (2015.07.27版)」、[バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.29  
内容: Rの基礎復習。「(Rで)塩基配列解析」の基本的な利用法。アノテーションファイルからの情報抽出。ド内部の説明。上下左右の矢印キー、タブ補完、二重クォーテーション問題、ありがちなミスなどの各種問題の解決。Blekhman et al., Genome Res., 2010の実データの読み込みから各種整形の基本。サンプル間クラスタリング。FASTA形式ファイルの解析やコード内部の説明。任意の領域の切り出し。関数の利用法、GC含量計算の説明。read.table, dim, table, sort, list.files, file.info, head, colnames, length, cbind, colSums, range, apply, min, max, colMeans, summary関数など。4コマ(4×90 min)分。Youtubeと統合TV。
- 門田幸二、「[Linux基礎](#) (2015.07.21版)」、[バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.23  
内容: Linux基本コマンド(pwd, cd, ls, rm, )のおさらい。wget, ドラッグ & ドロップ、共有フォルダ経由の各種データ取得手段。Integrative Genomics Viewer (IGV)のインストール。解凍(unzip)とパスを通す作業。Bio-LinuxにプレインストールされているFastQC (ver. 0.10.1)の実行。FastQC (ver. 0.11.3)のインストール。「ls -dj」や「rm -rf」などのオプションを徐々に追加説明。シェルスクリプトの基本。chmodでの権限変更。FastQCのバージョンの違いによる結果の違いを見る。正規表現。Genome Analysis Toolkit (GATK)の取得と解凍。whoami, sudo, bzip2, tar。4コマ(4×90 min)分。Youtubeと統合TV。
- 門田幸二、「[講義資料](#)」、[アグリバイオインフォマティクス教育研究プログラム](#)の大学院講義科目: [農学生命情報科学特論I第4回](#), 東京大学(東京), 2015.07.07  
内容: 教科書の3.3節周辺。FastQC (ver. 0.11.3)の--nogroupオプションの有無とKmer Contentの項目の違い。様々な角度で動作確認および検証することの重要性。adapters/primers除去後の乳酸菌paired-end RNA-seqデータのマッピング。アノテーション情報がないときとアノテーションファイル(GFF3)を利用したカウント情報取得実例。RPKMの基本的な考え方。配列長とカウント数の関係。原著論文(Blekhman et al., 2010)の公共カウントデータを利用した各種解析。エクセルファイルでRで読み込めること、サブセットのテクニック。サンプル間クラスタリング結果での実験デザインの説明や発現変動解析結果の予想。



# NGSハンズオン講習会

「NGSハンズオン講習会」のほうが、Rについては基本的なところをきっちり抑えているので、①や②も自習してください

http://www.iu.a.u-tokyo.ac.jp/~ka...seq.html#refe iu.a.u-tokyo.ac.jp の...

- 門田幸二、「R:Bioconductorの利用法(7/30分) (2015.07.31版)」, [バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.30  
内容: パッケージの説明, [CRANとBioconductor](#). 定期的なバージョンアップの重要性. インストール手順おさらい. [Bioconductor](#)概観. ゲノム配列パッケージ(BSgenome). プロモーター配列取得(sessionInfo, バージョンの違い) FASTA形式ファイルとGFF3形式ファイル. データの型. FASTQファイルの各種解析(LinuxとRを相補的に活用). FastQCとRパッケージ(ShortReadやQuasR)の比較. FastQC実行結果の各種項目(Overrepresented sequences, Per base sequence content, Kmer Content)の動作確認でRを利用など. 4コマ(4×90 min)分. [Youtube](#)と[統合TV](#). **①**
- 門田幸二、「R:基礎(7/29分) (2015.07.27版)」, [バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.29  
内容: Rの基礎復習. 「(Rで)塩基配列解析」の基本的な利用法. アンテーションファイルからの情報抽出. コード内部の説明. 上下左右の矢印キー, タブ補完, 二重クォーテーション問題, ありがちなミスなどの各種Tips. [Blekhman et al., Genome Res., 2010](#)の実データの読み込みから各種整形の基本. サンプル間クラスターリング. FASTA形式ファイルの解析やコード内部の説明. 任意の領域の切り出し. 関数の利用法. GC含量計算部分の説明. read.table, dim, table, sort, list.files, file.info, head, colnames, length, cbind, colSums, range, apply, min, max, colMeans, summary関数など. 4コマ(4×90 min)分. [Youtube](#)と[統合TV](#).
- 門田幸二、「Linux基礎 (2015.07.21版)」, [バイオインフォマティクス人材育成カリキュラム\(次世代シーケンサ\)NGSハンズオン講習会](#), 東京大学(東京), 2015.07.23  
内容: Linux基本コマンド(pwd, cd, ls, rm, )のおさらい. wget, ドラッグ & ドロップ, 共有フォルダ経由の各種データ取得手段. Integrative Genomics Viewer (IGV)のインストール. 解凍( unzip )とパスを通す作業. Bio-Linux 8 にプレインストールされているFastQC (ver. 0.10.1)の実行. FastQC (ver. 0.11.3)のインストール. 「ls -dj」や「rm -f」などのオプションを徐々に追加説明. シェルスクリプトの基本. chmodでの権限変更. FastQCのバージョンの違いによる実行結果の違いを見る. 正規表現. Genome Analysis Toolkit (GATK)の取得と解凍. whoami, sudo, bzip2, tarなど. 4コマ(4×90 min)分. [Youtube](#)と[統合TV](#).
- 門田幸二、「[講義資料](#)」, [アグリバイオインフォマティクス教育研究プログラム](#)の大学院講義科目: [農学生命情報科学特論1](#)第4回, 東京大学(東京), 2015.07.07  
内容: [教科書](#)の3.3節周辺. FastQC (ver. 0.11.3)の--nogroupオプションの有無とKmer Contentの項目の挙動の違い. 様々な角度で動作確認および検証することの重要性. adapters/primers除去後の乳酸菌paired-end RNA-seqデータのマッピング. アンテーション情報がないときとアンテーションファイル(GFF3)を利用したときのカウント情報取得実例. RPKMの基本的な考え方. 配列長とカウント数の関係. 原著論文([Blekhman et al 2010](#))の公共カウントデータを利用した各種解析. エクセルファイルでRで読み込めること, サブセットアップのテクニック. サンプル間クラスターリング結果での実験デザインの説明や発現変動解析結果の予想. TCC

# NGSハンズオン講習会

## (Rで)塩基配列解析

～NGS, RNA-seq, ゲノム, トランスクリプトーム, 正規化, 発現変動, 統計, モデル, バイオインフォマティクス～  
(last modified 2016/01/29, since 2011)

### 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 (Windows2015.04.03版とMacintosh2015.04.03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)
- 多群間比較用の推奨ガイドライン提唱論文 (Tang et al., BMC Bioinformatics, 2015)がpublishされました。論文概要  
については門田のページでも紹介しています。講習会でよく述べている「サンプル間クラスティング結果からDEG検  
出結果のおおよその見積もりが可能である」という主張の根拠となる原著論文がこれになります。推奨ガイドライン  
周辺の関連項目もアップデートしました。(2015/11/13)

- 日本乳酸菌学会誌のNGS関連連載の第5回ウェブ  
ので、若干プログラムのバージョンが上がっていま  
解析|一般|アライメント|についてを追加しまし  
日本乳酸菌学会誌のNGS関連連載の第4回ウェブ  
ので、若干プログラムのバージョンが上がっていま  
利関係上無条件公開はできませんので...欲しいブ  
ル希望」として私宛にメールしてください(本文は空

- はじめに (last modified 2015/03/31)
- 参考資料(講義, 講習会, 本など) (last modified 2015/03/31)
- 過去のお知らせ (last modified 2015/12/22)
- インストール|について (last modified 2015/11/12)
- インストール|R本体|最新版|新|用| (last modified 2015/11/12)

- インストール | R | パッケージ | **必要最小限(数GB?!)** (last modified 2015/05/25)
- インストール | R | パッケージ | **個別** (last modified 2015/06/10)
- (削除予定)Rのインストールと起動 (last modified 2015/04/02)
- (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)
- 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ハンズオン講習会2015** (last modified 2015/11/13)
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | **速習コース2014** (last modified 2015/02/11)
- 書籍 | トランスクリプトーム解析 | について** (last modified 2014/05/12)
- 書籍 | トランスクリプトーム解析 | **2.3.1 RNA-seqデータ(FASTQファイル)** (last modified 2014/04/15)
- 書籍 | トランスクリプトーム解析 | **2.3.2 リファレンス配列** (last modified 2014/04/16)
- 書籍 | トランスクリプトーム解析 | **2.3.3 アノテーション情報** (last modified 2014/04/17)
- 書籍 | トランスクリプトーム解析 | **2.3.4 マッピング(準備)** (last modified 2014/06/20)
- 書籍 | トランスクリプトーム解析 | **2.3.5 マッピング(本番)** (last modified 2014/06/21)
- 書籍 | トランスクリプトーム解析 | **2.3.6 カウントデータ取得** (last modified 2015/09/12)
- 書籍 | トランスクリプトーム解析 | **3.3.1 解析目的別留意点** (last modified 2014/04/20)
- 書籍 | トランスクリプトーム解析 | **3.3.2 データの正規化(基礎編)** (last modified 2014/06/23)
- 書籍 | トランスクリプトーム解析 | **3.3.3 クラスティング** (last modified 2014/04/20)
- 書籍 | トランスクリプトーム解析 | **3.3.4 各種プロット** (last modified 2014/04/27)



# NGSハンズオン講習会

①大元(NBDC)のサイトへのリンク。NBDCのサイトのほうが見やすいのは、全日程終了から数か月後に整形して公開するから当然です。  
②動画(統合TVとYouTube)も公開されている

## バイオインフォマティクス人材育成カリキュラム 2015

NGSハンズオン講習会を2015年7月22日-8月6日の11日間(エフ様分(服部先生と山口先生)の講義資料を差し替えました(2015/11/13)。

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のところに列挙した項目の予習は必須でしておきましょう。
- 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
  - 報告プレゼン資料(20150126\_kadota.pdf, 約1MB)  
報告書PDFの短縮版のようなものです。Twitterで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, 2015/07/22)の環境構築済みのovaファイルを用いた失敗、あるいは変なことになっちゃっても、10教にも合わせ150GB版と50GB版の2種類を用意しております。Macでは動作確認できています。Macよりも圧倒的に楽です。安心して沢山失敗してください。(2015/07/28)

http://biosciencedbc.jp/human/human-resources/workshop/h27

NBDC National Bioscience Database Center

English サイトマップ サイト内検索

Home ▶ 人材支援 ▶ 支援 ▶ 講習会 ▶ 平成27年度NGSハンズオン講習会

### 平成27年度NGSハンズオン講習会

平成27年度は、平成26年度の実績を踏まえ、講義内容の改善等を行い、ハンズオンに特化した、より効果的なNGS講習会を開催しました。

- H27年度概要
- H27年度講義日程・参考資料
- H26年度講習会の情報についてはこちらをご覧ください。
- H27年度講義資料・動画 \*講義資料一覧のファイル名をクリックすると資料ファイル(PDF等)がダウンロードできます。

実施日	実施時間	大項目	項目	レベル	習得技術	担当講師(敬称略)	講義資料・動画(統合TV)		
7月22日 (水)	10:30-12:00	PC環境の構築	Bio-Linux8とRのインストール状況確認		・Linux導入 ・R導入 ・NGS解析に必要な環境構築技術	門田 幸二 (東京大学)	事前予習資料一覧 (PDF:52KB)		
	13:15-14:45						寺田 透 (東京大学)	講義資料一覧 (PDF:108KB)	
	15:00-16:30							門田 幸二 (東京大学)	講義資料一覧 (PDF:32KB)
	16:45-18:15								統合TV
7月23日 (木)	10:30-12:00	UNIX/Linuxとスクリプト言語	Linux基礎	初級	UNIXの基礎的理解	門田 幸二 (東京大学)	講義資料一覧 (PDF:32KB)		
	13:15-14:45			中級			統合TV		
	15:00-16:30								
	16:45-18:15								

# NGSハンズオン講習会

①Rは2015年7月29-30日に開催。②動画はこちら

## バイオインフォマティクス人材育成カリキュラム(2015年度) NGSハンズオン講習会2015

NGSハンズオン講習会を2015年7月22日-8月6日の11日間(エフ様分(服部先生と山口先生)の講義資料を差し替えました(2015/11/13)。

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のところに列挙した項目の予習は必須でしておきましょう。
- 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
  - 報告プレゼン資料(20150126\_kadota.pdf, 約1MB)  
報告書PDFの短縮版のようなものです。Twitterで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, 2015/7/22の環境構築済みのovaファイルを用いた失敗、あるいは変なことになっちゃっても、10教にも合わせ150GB版と50GB版の2種類を用意しております。Macでは動作確認できています。Macよりも圧倒的に楽です。安心して沢山失敗してください。(2015/07/28)

開催日	時間	講義名	レベル	講師	講義資料		
7月28日 (火)	15:00-16:30				統合TV		
	16:45-18:15				統合TV		
	10:30-12:00		中級	Python	講義資料一覧 (PDF:52KB)		
	13:15-14:45				統合TV		
7月29日 (水)	15:00-16:30				統合TV		
	16:45-18:15				統合TV		
	10:30-12:00	データ解析環境R	R基礎 1	初級	R言語の基礎 (インストールから利用まで)	門田 幸二 (東京大学)	講義資料一覧 (PDF:37KB)
	13:15-14:45		R基礎 2	初級	・ファイルの読み込み ・行列演算の基本		統合TV
7月30日 (木)	15:00-16:30				統合TV		
	16:45-18:15				統合TV		
	10:30-12:00		Bioconductorの利用法 1	中級	データの型やバージョンの違い		講義資料一覧 (PDF:53KB)
	13:15-14:45		Bioconductorの利用法 2	中級	FASTA/FASTQファイルの各種解析		統合TV
8月3日 (月)	15:00-16:30				統合TV		
	16:45-18:15				統合TV		
	10:30-12:00	NGS解析[A日程]	NGS解析基礎	初級	・ファイル形式 ・可視化 (IGV) ・quality check ・マッピング ・アセンブル	山口 昌雄 (アメリエフ)	講義資料一覧 (PDF:79KB)
	13:15-14:45						統合TV
8月4日 (火)	15:00-16:30				統合TV		
	16:45-18:15				統合TV		
	10:30-12:00		ゲノムReseq、変異解析	初級	代表的なパイプラインについての実習:ゲノムReseq、変異解析	山口 昌雄 (アメリエフ)	講義資料一覧 (PDF:82KB)
	13:15-14:45						統合TV
	15:00-16:30				統合TV		

# 乳酸菌NGS連載

NGS連載関連。①主に全体像、原稿およびウェブ資料関係。②赤枠の個別の回のところで、原稿中のプログラムへのリンクや、コピー用Linuxコマンドなどを利用可能

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化  
(last modified 2016/01/29, since 2011)

### 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、ソフトウェアと必要なパッケージをインストール済み  
用法(Windows2015.04.03版とMacintosh2015.04.03版)も  
あります。(2015/04/03)
- 多群間比較用の推奨ガイドライン提唱論文(Tang et al.)  
については門田のページでも紹介しています。講習会  
出結果のおおよその見積もりが可能である」という主  
周辺の関連項目もアップデートしました。(2015/11/05)
- 日本乳酸菌学会誌のNGS関連連載の第5回ウェブ資料  
なので、若干プログラムのバージョンが上がっています
- 解析|一般|アライメント|についてを追加しました
- 日本乳酸菌学会誌のNGS関連連載の第4回ウェブ資料  
なので、若干プログラムのバージョンが上がっています  
利用関係上無条件公開はできませんので...欲しい方は  
ご希望として私宛にメールしてください(本文は空で  
OK)
- はじめに (last modified 2015/03/31)
- 参考資料(講義、講習会、本など) (last modified 2015/03/31)
- 過去のお知らせ (last modified 2015/12/22)
- インストール|について (last modified 2015/11/12)
- インストール|R本体|最新版|用法(Windows2015.04.03版とMacintosh2015.04.03版) (last modified 2015/04/03)

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

- 書籍 | トランスクリプトーム解析 | [3.3.3 クラスターリング](#) (last modified 2014/04/20)
- 書籍 | トランスクリプトーム解析 | [3.3.4 各種プロット](#) (last modified 2014/04/27)
- 書籍 | トランスクリプトーム解析 | [4.3.1 シミュレーションデータ\(負の二項分布\)](#) (last modified 2014/04/27)
- 書籍 | トランスクリプトーム解析 | [4.3.2 データの正規化\(応用編\)](#) (last modified 2014/04/27)
- 書籍 | トランスクリプトーム解析 | [4.3.3 2群間比較](#) (last modified 2014/04/28)
- 書籍 | トランスクリプトーム解析 | [4.3.4 他の実験デザイン\(3群間\)](#) (last modified 2014/04/28)
- 書籍 | 日本乳酸菌学会誌 | [|について](#) (last modified 2015/12/28)
- 書籍 | 日本乳酸菌学会誌 | [第1回イントロダクション](#) (last modified 2015/09/11)
- 書籍 | 日本乳酸菌学会誌 | [第2回GUI環境からコマンドライン環境へ](#) (last modified 2015/11/26)
- 書籍 | 日本乳酸菌学会誌 | [第3回Linux環境構築からNGSデータ取得まで](#) (last modified 2015/12/07)
- 書籍 | 日本乳酸菌学会誌 | [第4回クオリティコントロールとプログラムのインストール](#) (last modified 2015/12/11)
- 書籍 | 日本乳酸菌学会誌 | [第5回アセンブル、マッピング、そしてQC](#) (last modified 2015/12/22)
- 書籍 | 日本乳酸菌学会誌 | [第6回ゲノムアセンブリ](#) (last modified 2016/01/26) **NEW**
- 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 2015/07/26)
- イントロ | 一般 | [ランダムな塩基配列を生成](#) (last modified 2014/06/16)
- イントロ | 一般 | [任意の長さの可能な全ての塩基配列を作成](#) (last modified 2015/02/19)
- イントロ | 一般 | [任意の位置の塩基を置換](#) (last modified 2013/09/12)
- イントロ | 一般 | [指定した範囲の配列を取得](#) (last modified 2015/04/06)
- イントロ | 一般 | [指定したID\(染色体やdescription\)の配列を取得](#) (last modified 2014/03/10)
- イントロ | 一般 | [翻訳配列\(translate\)を取得\(基礎\)](#) | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | [翻訳配列\(translate\)を取得\(応用\)](#) | [seqinr\(Charif 2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2013/06/14)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2013/06/14)

[トップページへ](#)

# HPCI講習会のPC環境

実習用PC環境は、①の手順に従って「Rおよび必要なパッケージ」のインストールが完了している状態です。自分のPCで復習したい場合は①を参考にして自力で環境構築してください。

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2016/01/29, since 2011)

### 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 ([Windows2015.04.04版](#)と[Macintosh2015.04.03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)
- 多群間比較用の推奨ガイドライン提唱論文([Tang et al., BMC Bioinformatics, 2015](#))がpublishされました。論文概要については門田のページでも紹介しています。講習会でよく述べている「サンプル間クラスタリング結果からDEG検出結果のおおよその見積もりが可能である」という主張の根拠となる原著論文がこれになります。推奨ガイドライン周辺の関連項目もアップデートしました。(2015/11/05) **NEW**
- [日本乳酸菌学会誌](#)の[NGS関連連載](#)の第5回ウェブ資料を更新しました。2015年12月下旬に一気に全てやり直したので、若干プログラムのバージョンが上がっています。(2015/12/22)
- [解析 | 一般 | アライメント | | について](#)を追加しました。(2015/12/16)
- [日本乳酸菌学会誌](#)の[NGS関連連載](#)の第4回ウェブ資料を更新しました。2015年12月初旬に一気に全てやり直したので、若干プログラムのバージョンが上がっています。各回終了時点のovaファイル(約6GB)も提供可能です。(権利関係上無条件公開はできませんので...)欲しい方は、メールのタイトルを「乳酸菌連載第x回終了時点のovaファイル希望」として私宛にメールしてください(本文は空でOK)。URLをお知らせします。(2015/12/11)

- [はじめに](#) (last modified 2015/03/31)
- [参考資料\(講義、講習会、本など\)](#) (last modified 2015/11/17)
- [過去のお知らせ](#) (last modified 2015/12/22)
- [インストール | | について](#) (last modified 2015/11/12)

[トップページへ](#)

インストール | | 本ページ | 最新版 | [Windows](#) (last modified: 1 2015/03/31) 推奨



# HPCI講習会のPC環境

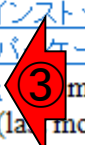
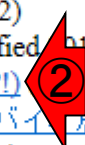
具体的な順番は、①R本体のインストール、  
②各種Rパッケージのインストールです。  
③の「基本的な利用法」の習得は、HPCI講習会の枠組みでは必須ではありません

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2016/01/29, since 2011)

### 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 (Windows2015.04.04版とMacintosh2015.04.03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)
  - 多群間比較用の推奨ガイドライン提唱論文(Tang et al.)については門田のページでも紹介しています。講習会結果のおおよその見積もりが可能である」という点と周辺の関連項目もアップデートしました。(2015/11/05)
  - 日本乳酸菌学会誌のNGS関連連載の第5回ウェブページなので、若干プログラムのバージョンが上がっています
  - 解析|一般|アラインメント|についてを追加しました
  - 日本乳酸菌学会誌のNGS関連連載の第4回ウェブページなので、若干プログラムのバージョンが上がっています(関係上無条件公開はできませんので...)欲しい方「希望」として私宛にメールしてください(本文は空で)
- はじめに (last modified 2015/03/31)
  - 参考資料(講義、講習会、本など) (last modified 2015/11/17)
  - 過去のお知らせ (last modified 2015/12/22)
  - インストール|について (last modified 2015/11/12)
  - インストール | R本体 | 最新版 | Win用 (last modified 2015/03/22)推奨
  - インストール | R本体 | 最新版 | Mac用 (last modified 2015/04/22)推奨
  - インストール | R本体 | 過去版 | Win用 (last modified 2015/03/22)
  - インストール | R本体 | 過去版 | Mac用 (last modified 2015/03/22)
  - インストール | Rパッケージ | ほぼ全て(20GB以上?!)(last modified 2015/05/25)
  - インストール | Rパッケージ | 必要最小限プラスアルファ(数GB?!)(last modified 2015/12/18)推奨
  - インストール | Rパッケージ | 必要最小限プラスアルファ(アグリバイ(居室のみ))(last modified 2015/06/16)
  - インストール | Rパッケージ | 必要最小限(数GB?!)(last modified 2015/05/25)
  - インストール | Rパッケージ | 個別 (last modified 2015/06/10)
  - (削除予定)Rのインストールと起動 (last modified 2015/04/02)
  - (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)
  - 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ハンズオン講習会2015 (last modified 2015/06/15)



# HPCI講習会

①HPCI講習会の②バイオインフォマティクス実習コース、の中の一部が門田担当。

The screenshot shows a web browser window with the URL <https://hpci.cbrc.jp/modules/tutorial/index.html>. The page title is "HPCI 戦略プログラム 分野1 代表機関 国立研究開発法人 理化学研究所 「予測する生命科学・医療および創薬基盤」 人材養成プログラム". The main content area is titled "講習会" (Workshop) and "2015年度 HPCI 講習会". It contains a paragraph of text and a circular image of a computer lab. A sidebar on the left has a menu with "tutorial" highlighted by a red arrow with the number 1. Below the text, there is a table with two columns: "開設コース" (Established Course) and "受講状況" (Enrollment Status). The first row lists "バイオインフォマティクス実習コース" (Bioinformatics Practical Course) and "HPCI チュートリアルセミナー" (HPCI Tutorial Seminar). The second row lists "創薬インフォマティクス実習コース" (Drug Discovery Informatics Practical Course). A red arrow with the number 2 points to the "受講状況" column, which shows "受講中" (In Progress).

HPCI 戦略プログラム 分野1 代表機関 国立研究開発法人 理化学研究所  
「予測する生命科学・医療および創薬基盤」  
人材養成プログラム  
実施機関 国立研究開発法人 産業技術総合研究所 創薬基盤研究部門

top  
outline  
seminar  
workshop  
tutorial ①  
e-learning  
links  
contact  
faq

講習会  
2015年度 HPCI 講習会  
2015年度 講習会

産総研 生命工学領域 創薬基盤研究部門では、2011年度より生命情報工学研究センター、ゲノム情報研究センターが実施してきたHPCI人材養成事業を継続してまいります。2001年設立の生命情報科学研究センター以来、研究活動だけでなく人材養成にも注力し、この間に培ってきたノウハウを活かして毎年カリキュラム構成を見直し、初心者にもわかりやすく丁寧な実習指導付きの講習会を開催してまいりました。実習には1人1台の備付けPC(Windows7)を使用します。講師は産総研の研究員をはじめ、第一線の研究者が務めます。

開設コース

バイオインフォマティクス実習コース	② 受講中
HPCI チュートリアルセミナー	
創薬インフォマティクス実習コース	

ユーザー名:  
パスワード:

# HPCI講習会

具体的には、①「Rを使ったNGS解析を基礎から学ぶ」のうちの、②塩基配列解析(特にゲノム解析とトランスクリプトーム解析部分)が門田の担当

## ● バイオインフォマティクス実習コース

- バイオインフォマティクスの基礎知識・実践技術を短期間に習得 -

第一線の研究者が講師として、バイオ情報取り扱いの基礎理論から実際の解析研究までをテーマごと

に指導します。計算機実習は1人1台の混ぜたカリキュラムで、バイオインフで講義・実習を行い、受講を希望する知識や技術を設定した受講要件がある本コースのカリキュラムは、これまでイオインフォマティクス速習コースIIクス実習コースのカリキュラムを元に

大量の配列データも怖くない!! Windows上の Linux 環境で高速・簡単に配列データ解析

A-1	Linux, Perl基礎		2015年9月17日(木) / 9月18日(金)
A-2	配列解析1	ChIP-seqデータ解析およびENCODEプロジェクトなどによる既存のデータの活用	2015年9月30日(水)午後 -10月1日(木)
		LASTIによるさまざまな配列解析	2015年10月2日(金)
A-3	配列解析2	配列モチーフ探索	2015年10月15日(木)



① フリーウェア 統計解析パッケージ R を使った NGS データ解析を基礎から学ぶ

B-1	R基礎*		2016年3月2日(水)
B-2	Rで塩基配列解析: ゲノム解析からトランスクリプトーム解析まで*	②	2016年3月3-4日(木,金)
B-3	多変量データ解析 / 遺伝子ネットワーク解析*		2016年3月10-11日(木,金)

# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境


## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# NGSデータ解析戦略

自分の置かれている環境・予算・ポリシーによっても異なる。どの選択肢でも基本正解。Rは、主に統計解析部分で使われている。

- 解析受託企業に外注：Linuxコマンドを知らなくてもよい
  -  ngs 受託解析
- クラウド（ウェブツール）：Linuxコマンドを知らなくてもよい
  - DDBJ Pipeline (Nagasaki et al., *DNA Res.*, 20: 383–390, 2013)
  - Illumina BaseSpace
  - Galaxy (Goecks et al., *Genome Biol.*, 11: R86, 2010)
  - ...
- Linuxコマンドを駆使（旧来型）
  - なるべく自力で解析
  - LinuxコマンドやNGS解析用プログラムのインストールなどを練習し、スパコンを使いこなす
  - NBDC/東大アグリバイオ/HPCIの「NGSハンズオン講習会」の方向性



# DDBJ Pipeline

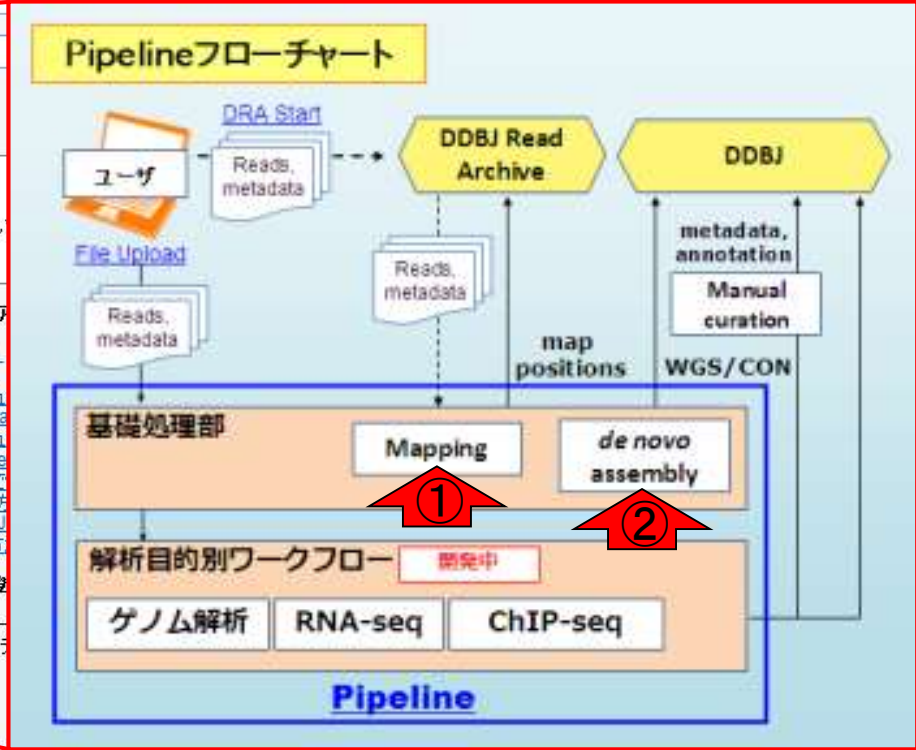
DDBJ Pipelineでは、主に①マッピングや② *de novo*アセンブリができる。特に後者ができるのは非常に有難い。③新規アカウント作成から*de novo*アセンブリまでの詳細については、乳酸菌連載第6回ウェブ資料を参照

The screenshot shows the DDBJ Pipeline website interface. At the top, there's a navigation bar with the DDBJ logo and the title "DDBJ Read Annotation Pipeline". Below this, there are language selection buttons for "English" and "Japanese". A description states: "DDBJ Read Annotation Pipelineは、次世代シーケンサ配列のクラウド型データ解析プラットフォームです。" (DDBJ Read Annotation Pipeline is a cloud-based data analysis platform for next-generation sequencing data).

The main content area is titled "LOGIN" and includes a "新規アカウント作成" (New Account Creation) link. There are input fields for "User ID:" and "Password:" with a "Login" button. Below the login form, there's a section for "動作中JOBの確認" (Check running jobs) with a note: "PipelineのIDをお持ちでないことができます。" (You can check jobs without the Pipeline ID).

On the left side, there's a "Pipelineフローチャート" (Pipeline Flowchart) diagram. It shows the flow from "ユーザ" (User) through "DRA Start" (Reads, metadata) to "DDBJ Read Archive" (Reads, metadata) and finally to "DDBJ" (metadata, annotation, Manual curation). The "基礎処理部" (Basic Processing Unit) includes "Mapping" and "de novo assembly". Below this, there's a "解析目的別ワークフロー" (Workflow by analysis purpose) section with options for "ゲノム解析" (Genome analysis), "RNA-seq", and "ChIP-seq".

At the bottom left, there's a "Tweets" section with a tweet from @pipeline\_info dated 22 Dec, mentioning that email support will not be open from Dec 25 to Jan 3.



# NGSデータ

乳酸菌(*Lactobacillus hokkaidonensis* LOOC260<sup>T</sup>)ゲノム解読論文(PMID: 25879859)。Illumina MiSeqデータ(DRR24501)とPacBioデータ(DRR024500)を併用することでcomplete genomeを得ることができた、という内容。尚、DRR024500は登録内容の誤りが判明し、2016年1月末に削除されDRR054113-054116に差し替えられている。

The screenshot shows a PubMed abstract page. At the top, the URL is <http://www.ncbi.nlm.nih.gov/pubmed/25879859>. The page title is "Complete genome sequence and analysis of *Lactobacillus hokkaidonensis* LOOC260(T), a psychrotrophic lactic acid bacterium isolated from silage." The authors listed are Tanizawa Y<sup>1,2</sup>, Tohno M<sup>3</sup>, Kaminuma E<sup>4</sup>, Nakamura Y<sup>5</sup>, and Arita M<sup>6,7</sup>. The abstract text includes a background section stating the bacterium is isolated from Timothy grass silage in Hokkaido, Japan, and a results section describing the genome structure (one circular chromosome and two plasmids). The conclusions mention that this is the first complete genome in the *L. vaccinostercus* group. On the right side of the page, there are sections for "Full text links" (BioMed Central, PMC Full text), "Save items" (Add to Favorites), "Similar articles" (listing related papers), and "Related information".

Tanizawa et al., *BMC Genomics*, 16: 240, 2015



# NGSデータ

① Full textリンク先で全文を見られる。② Availability of supporting dataという項目をよく眺めると、生データがDDBJ Sequence Read Archive (DDBJ SRA; 略してDRA)にDRR024500とDRR24501というIDで登録されていることがわかる

NCBI Resources How To

PubMed

Search

Advanced Help

Abstract

BMC Genomics. 2015 Mar 25;16:240. doi: 10.1186/s12864-015-1435-2.

**Complete genome sequence and analysis of *Lactobacillus hokkaidonensis* LOOC260(T), a psychrotrophic lactic acid bacterium isolated from silage.**

Tanizawa Y<sup>1,2</sup>, Tohno M<sup>3</sup>, Kaminuma E<sup>4</sup>, Nakamura Y<sup>5</sup>, Arita M<sup>6,7</sup>.

Author information

**Abstract**

**BACKGROUND:** *Lactobacillus hokkaidonensis* is an obligate heterofermentative lactic acid bacterium, which is isolated from Timothy grass silage in Hokkaido, a subarctic region of Japan. This bacterium is expected to be useful as a silage starter culture in cold regions because of its remarkable psychrotolerance; it can grow at temperatures as low as 4°C. To elucidate its genetic background particularly in relation to the source of psychrotolerance, we constructed the complete genome sequence of *L. hokkaidonensis* LOOC260(T) using PacBio single-molecule real-time sequencing technology.

**RESULTS:** The genome of LOOC260(T) comprises one circular chromosome (2.28 Mbp) and two circular plasmids: pLOOC260-1 (81.6 kbp) and pLOOC260-2 (41.0 kbp). We identified diverse genetic elements, such as prophages, integrated and conjugative elements, and conjugative elements which may reflect adaptation to plant-associated niches. Comparative genome analysis also revealed unique genomic features, such as genes involved in pentose assimilation and NADPH generation.

**CONCLUSIONS:** This is the first complete genome in the *L. vaccinostercus* group, which is characterized, so the genomic information obtained in this study provides insight into the genome evolution of this group. We also found several factors that may contribute to the ability of *L. hokkaidonensis* to grow at cold temperatures. The results of this study will facilitate further research for the cold-tolerance mechanism of *L. hokkaidonensis*.

PMID: 25879859 [PubMed - in process] PMID: PMC4377027 Free PMC Article

Full text links

Read free full text at BioMed Central

PMC Full text

Save items

Add to Favorites

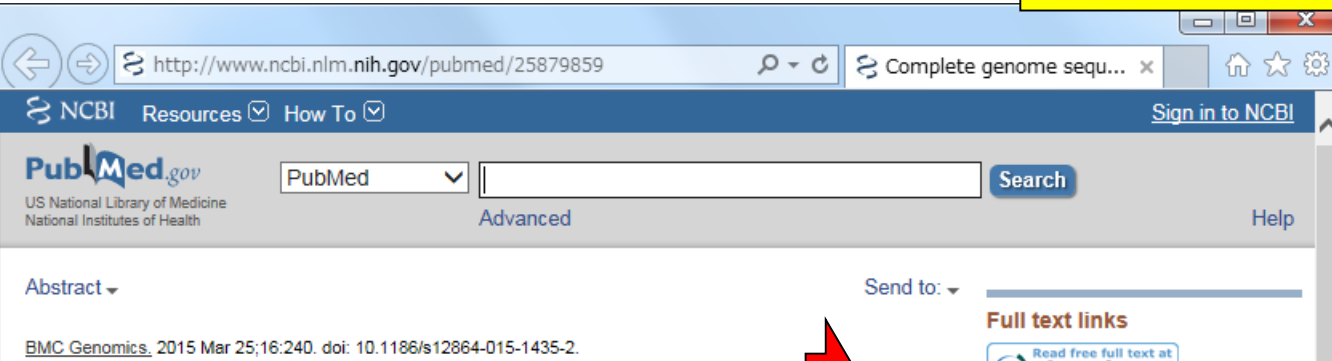
Similar articles

**Availability of supporting data**

The complete genome sequence of *L. hokkaidonensis* LOOC260<sup>T</sup> and its annotations were deposited at DDBJ/ENA/GenBank under accession numbers AP014680 (chromosome), AP014681 (plasmid pLOOC260-1), and AP014682 (plasmid pLOOC260-2). All of the sequencing data were deposited in the DDBJ Sequence Read Archive under accession numbers DRR024500 and DRR024501. The phylogenetic tree and associated data matrix for in Additional file 1: Figure S2 are available in TreeBASE database (Accession URL: <http://purl.org/phylo/treebase/phylovs/study/TB2:S17206>).

# NGSデータ

① Genome sequencing and *de novo* assemblyという項目を見ると、②paired-endで5,942,620リードと書いてある。一応公共DB(DRA)上で確認する。



① **Genome sequencing and *de novo* assembly**

The cells of *L. hokkaidonensis* LOOC260<sup>T</sup> were cultured in MRS (de Man, Rogosa, and Sharpe) broth (Difco) and were harvested in the mid-logarithmic phase. The genomic DNA was extracted and purified using Qiagen Genomic-tip 500/G and Qiagen Genomic DNA Buffer Set with lysozyme (Sigma) and proteinase K (Qiagen) according to the manufacturer's instruction. PacBio SMRT whole-genome sequencing was performed using a PacBio RSII sequencer with P4-C2 chemistry. Four SMRT cells were used for sequencing, thereby yielding 163,376 adapter-trimmed reads (subreads) with an average read length of approximately 4 kbp, which corresponded to approximately 250-fold coverage. *De novo* assembly was conducted using the HGAP method based on the SMRT Analysis package 2.0, which yielded seven contigs. Independent genome sequencing using the 250-bp paired-end Illumina MiSeq system generated 5,942,620 reads, which were assembled into contigs using Platanus assembler v1.2.2 with the default settings [40]. The initial contigs



# NGSデータ

① Genome sequencing and *de novo* assemblyという項目を見ると、② paired-endで5,942,620リードと書いてある。③ DRR024501というIDのほうは、④ 2,971,310リードと書いてある。5,942,620 / 2 = 2,971,310である。ウェブサイト上の数値は、single-endとしてのリード数と考えれば妥当

https://trace.ddbj.nig.ac.jp/DRAsearch/run?acc=DRR024501 DRR024501 -  
DRASearch Send Feedback Search

DRR024501 FASTQ SRA

Run Detail	
Alias	DRR024501
Instrument model	
Date of run	
Run center	
Number of spots	2,971,310
Number of bases	1,491,597,620

Navigation  
Submission DRA002643 FTP  
Study DRP002401

READS (joined) quality show 10 rows << < 1

```
>DRR024501.1
ATGNATCGAAACAGTATTTACAAGATTTGCATACTGAAATTTGAAGCTGATCAACACGAAACCATTC
AATCTAAACACCCATTAGCTGTTATTGAAGCTTTGCAGCAACGAGTTGATGATAAAATGACCGTTT
GAGCCATTATATTTGGATGGCCCGGCACCTCCGAAGTTATGAGCCTCGCCATTTATTGTTTAGTAA
TTGGAGTGGCGATGAACCGTATTAAGCCCTAAACGAACGGCTGTCTCCAGTCTTGTCCAGTAAAT
CCAGAAACAGAGACTGATTAGCATTGGGCCGAACTAACCGCAGCCGAAATTTGACCAAGGTAGCGCC
GCATCCCACTTAACAATAAATGGCGAGGCTCATAACTTCGGAAGTGCCGGGCCATCCAAATATA
TCAAACGCAACGTTTCATATTAT

>DRR024501.2
GTCNGAACACATGAATGGTGAACGGCGCTGAACTTTTACGGACGCGGCACGAGGATCCACAGGGC
AAACACGTCACAGACTTGTATCACCGCATTATTACGTGAGTGGATTTCTGATAAAACAGTGTAA
AATTCGATTTCTAGCCAATCAAAGACAGATTTAACGAAATCACAGATGACCAAAATTTGGCATTGAGT
TAAAAAAGTGATGGACCCCTCTTTAACCCTAAGTTGTCCCGAATAACATTCGAAACTCTCTGCTTTT
ACCTCAAATGATTTGCCCAATCAATTTGACTTGTCTTGTGTCATGATCTGATTTCACTGTTTAT
ACTCAATGCCAATTTGGTCATCTGTGATTTCTGTTAACTCTGTCTTTGATTGGCTAGACATCGAATT
TTCACACTGTTTTATACGAAAT

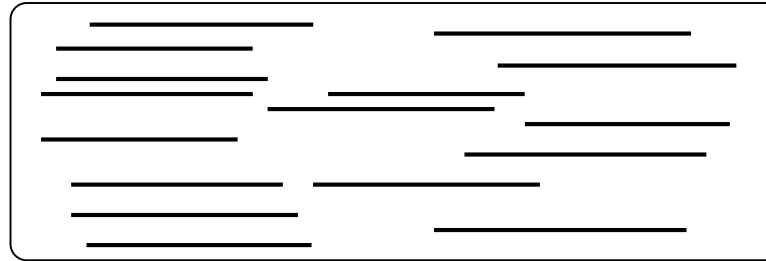
>DRR024501.3
CAANGATACAATCATTATCATGAATCTAATGCCGGTTCTGGTATGCACTTCTACTGTTTGGCTTGAAGCC
CTGGTGACACGCACTCAGTTCTTAACAAACTAGGCGATTAT
```

**Genome sequencing and *de novo* assembly**  
The cells of *L. hokkaidonensis* LOOC260<sup>T</sup> were cultured in MRS (de Man, Rogosa, and Sharpe) broth (Difco) and were harvested in the mid-logarithmic phase. The genomic DNA was extracted and purified using Qiagen Genomic-tip 500/G and Qiagen Genomic DNA Buffer Set with lysozyme (Sigma) and proteinase K (Qiagen) according to the manufacturer's instruction. PacBio SMRT whole-genome sequencing was performed using a PacBio RSII sequencer with P4-C2 chemistry. Four SMRT cells were used for sequencing, thereby yielding 163,376 adapter-trimmed reads (subreads) with an average read length of approximately 4 kbp, which corresponded to approximately 250-fold coverage. *De novo* assembly was conducted using the HGAP method based on the SMRT Analysis package 2.0, which yielded seven contigs. Independent genome sequencing using the 250-bp paired-end Illumina MiSeq system generated 5,942,620 reads, which were assembled into contigs using Platanus assembler v1.2.2 with the default settings [40]. The initial contigs

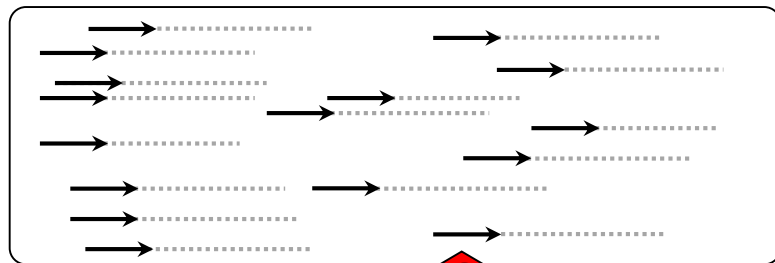
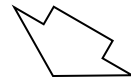
# 用語：リード

リードとは、Sequencerで読んだ塩基配列のこと。①黒矢印の一本一本がリードに相当する。

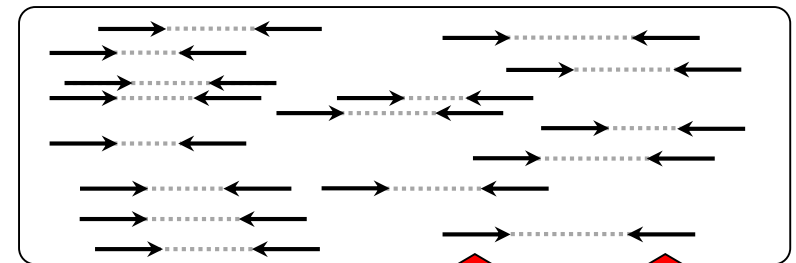
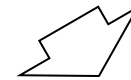
断片化されたゲノム配列



single-endの場合



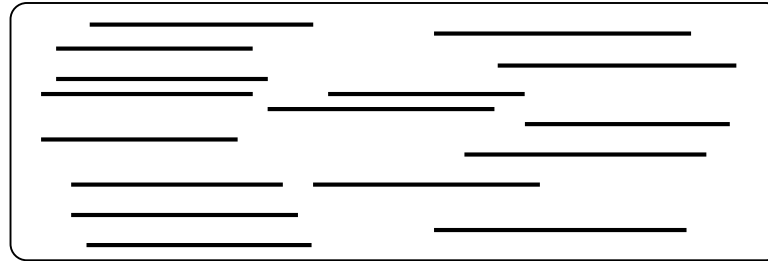
paired-endの場合



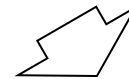
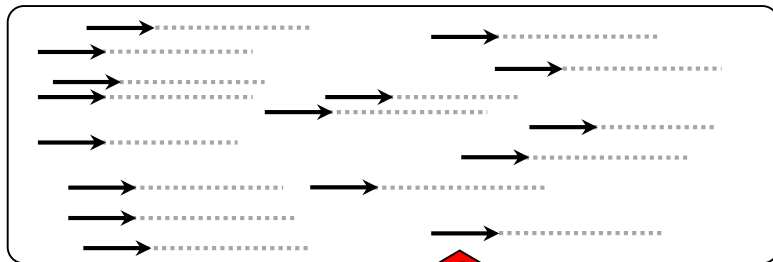
# 用語 : single-end

①断片化された配列の片側のみを読む場合を single-end という。この場合は右向き矢印のみ

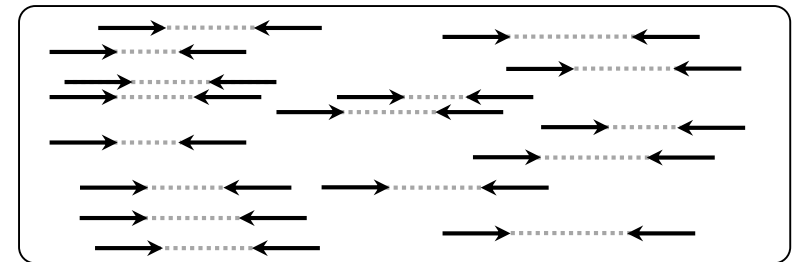
断片化されたゲノム配列



single-endの場合



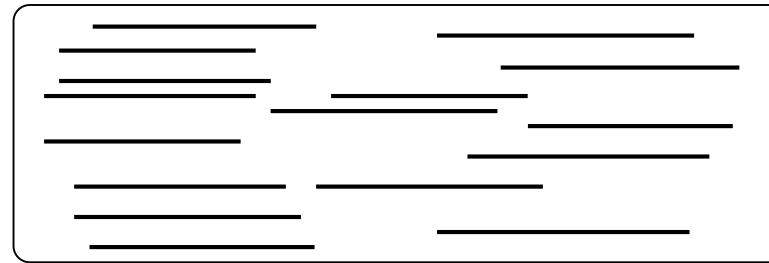
paired-endの場合



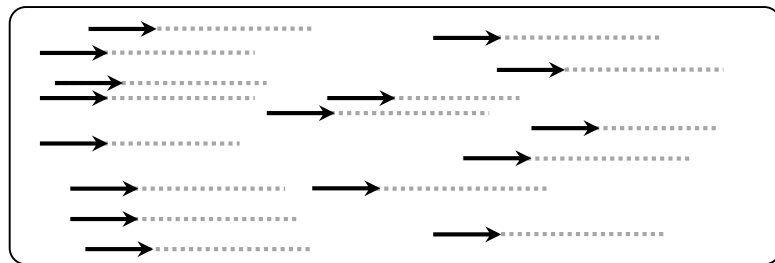
# 用語: paired-end

①断片化された配列の両側から読む場合を paired-end という。②右向き矢印と③左向き矢印のリードが読まれることになる。それぞれを forward側リード、reverse側リードなどと呼ぶ。

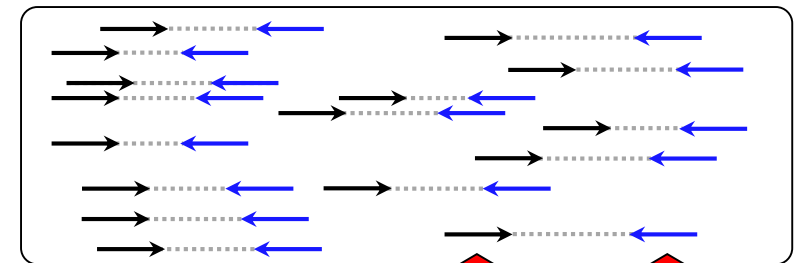
断片化されたゲノム配列



single-endの場合



paired-endの場合



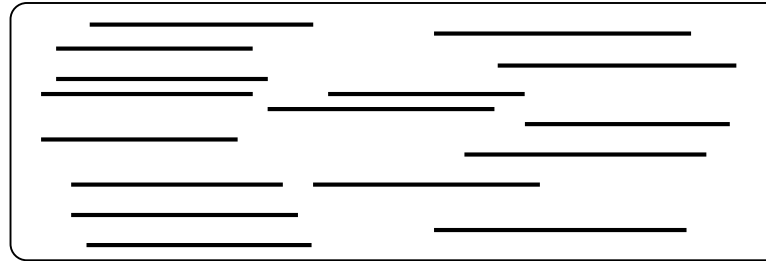
forward側

reverse側

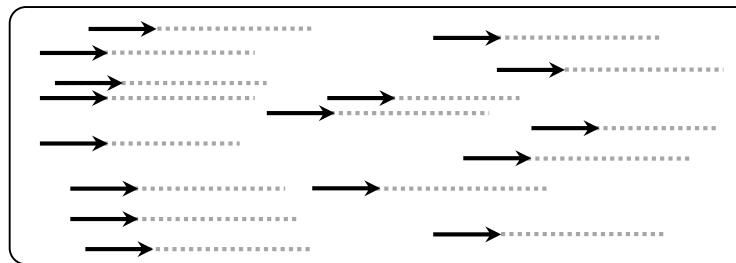
# 用語: paired-end

ILLUMINA MiSeqデータ(DRR24501)の場合、① forward側、② reverse側ともに、矢印の長さが250 bp、矢印の本数(リード数)が計5,942,620個(約594万;片側のみで約297万)に相当。

断片化されたゲノム配列

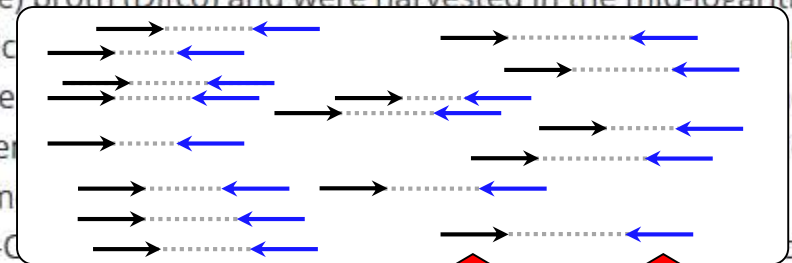


single-endの場合



## Genome sequencing and *de novo* assembly

The cells of *L. hokkaidonensis* LOOC260<sup>T</sup> were cultured in MRS (de Man, Rogosa, and Sharpe) broth (Difco) and were harvested in the mid-logarithmic phase. The genomic DNA was extracted using the DNeasy spin kit 500/G and Qiagen proteinase K (Qiagen). The genome was sequenced using SMRT whole-genome sequencing with P4-C1000 sequencer with P4-C1000, thereby yielding 163,376 adapter-trimmed reads with an average read length of approximately 4 kbp, which corresponded to approximately 250-fold coverage. *De novo* assembly was conducted using the HGAP method based on the SMRT Analysis package 2.0, which yielded seven contigs. Independent genome sequencing using the 250-bp paired-end Illumina MiSeq system generated 5,942,620 reads, which were assembled into contigs using Platanus assembler ver 1.2 with the default settings [40]. The initial contigs



①

②





# DDBJ SRA (DRA)

①forward側 : DRR024501\_1.fastq.bz2、  
②reverse側 : DRR024501\_2.fastq.bz2、  
のような感じ。DRAの場合は、bzip2圧縮FASTQファイルをダウンロード可能。乳酸菌ゲノム配列決定論文では、このデータを入力としてde novoアセンブリが行われた。

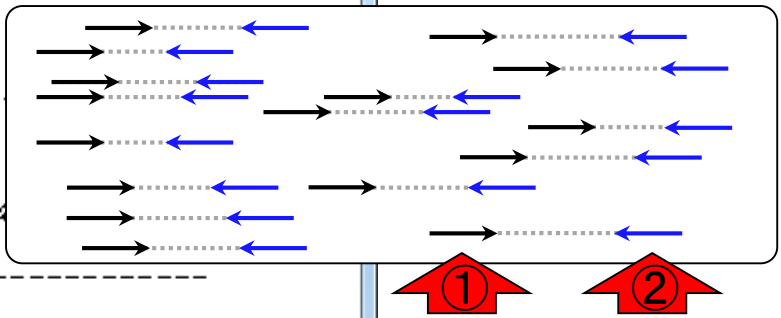
FTP ディレクトリ /ddbj\_database/dra/fastq/DRA002/DRA002643/D  
ftp.ddbj.nig.ac.jp

エクスプローラーでこの FTP サイトを表示するには、Alt キーを押して、[表示]をクリックし、[エクスプローラーで FTP サイトを開く]をクリックしてください。

-Welcome to DDBJ FTP Archive, running on ftp.ddbj.nig.ac.jp!  
-Please contact ddbj@ddbj.nig.ac.jp when you have any problem access to this archive, downloading the data, and etc.  
-For details on the directory structure and file contents, please refer to the README.TXT placed in the "/ddbj\_database".

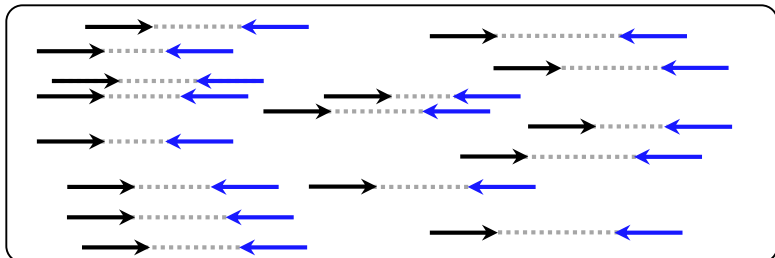
[1階層上のディレクトリへ](#)

11/12/2014 12:00午前	470,724,676	<a href="#">DRR024501_1.fastq.bz2</a>
11/12/2014 12:00午前	528,430,059	<a href="#">DRR024501_2.fastq.bz2</a>

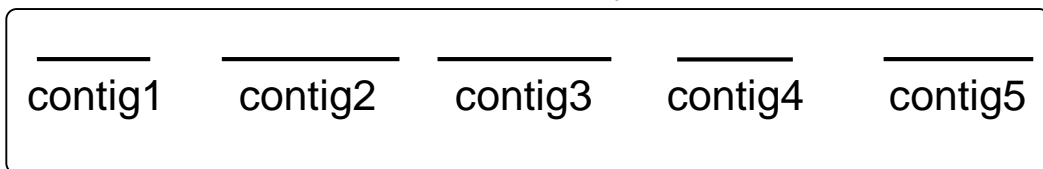


# 用語: コンティグ

入力: paired-end FASTQファイル



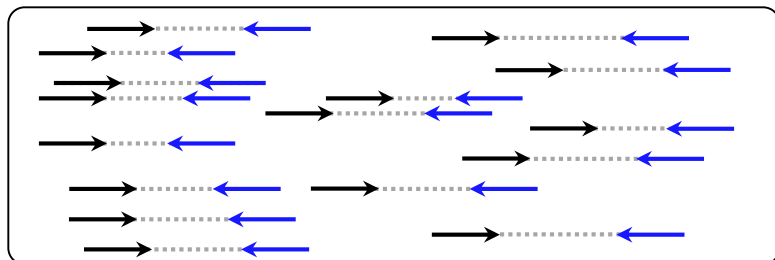
Assembly (コンティグの作成)



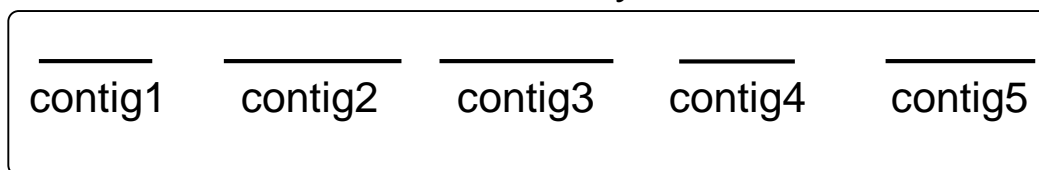
(通常は) paired-endのリードファイルを入力として、*de novo*アセンブリプログラムを実行した結果として得られる、異なる複数のリードが(ACGTの切れ目なく)つなげられたもの。  
contiguous sequence (連続的な配列) という意味。通常、元のリード長よりも長くなる。

# 用語 : scaffold

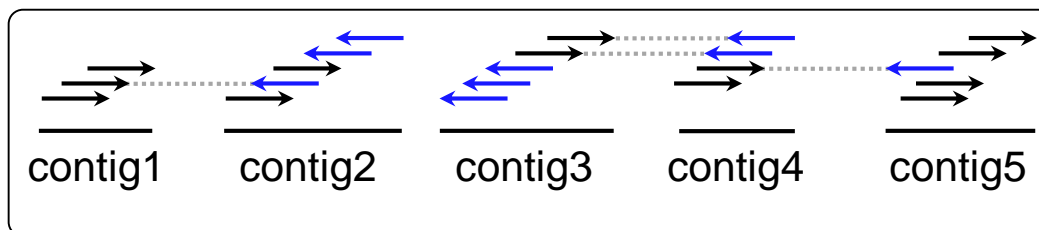
入力 : paired-end FASTQファイル



↓ Assembly (コンティグの作成)

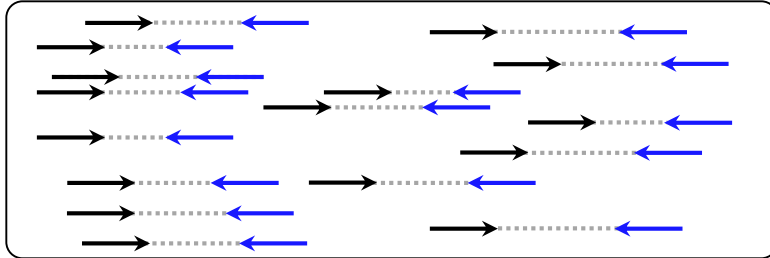


↓ Scaffold

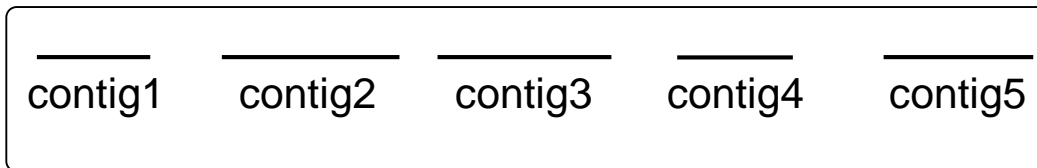


# 用語: scaffold

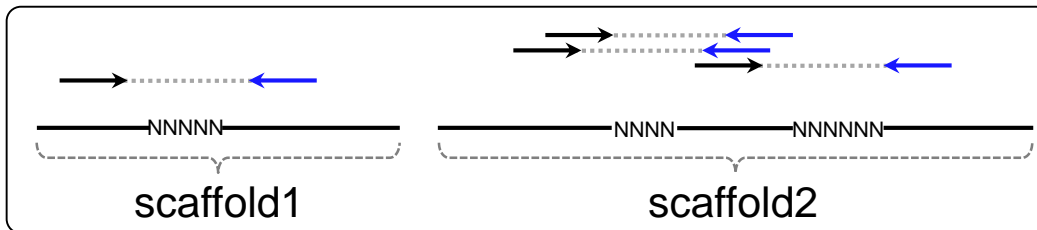
入力: paired-end FASTQファイル



Assembly (コンティグの作成)



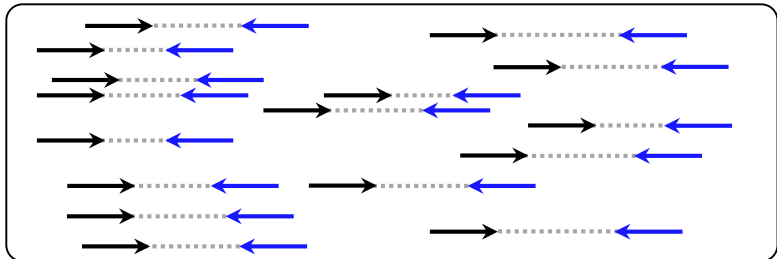
Scaffold



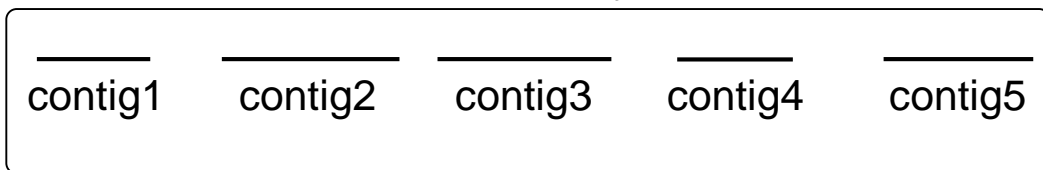
得られたコンティグにリードをマップし…ペアの情報を頼りにコンティグ間にNを入れて連結したもの。supercontigともいう。scaffoldの数はcontigの数よりも少なくなる。尚、Nを入れた部分をgapという

# 用語: gap close

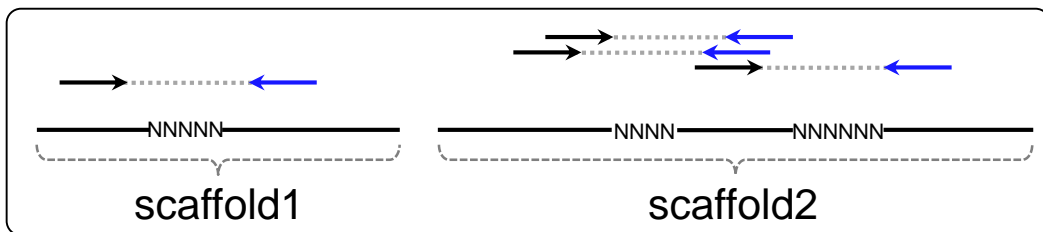
入力: paired-end FASTQファイル



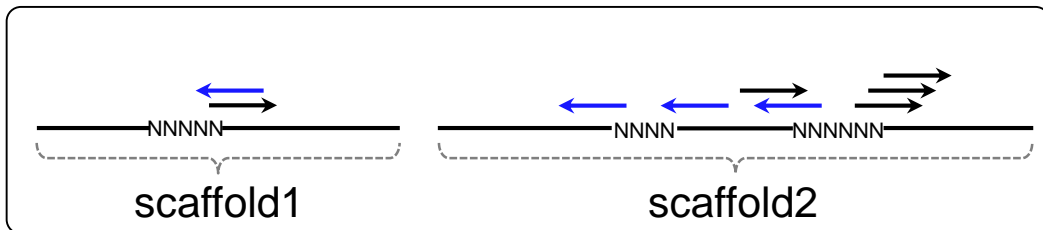
Assembly (コンティグの作成)



Scaffold

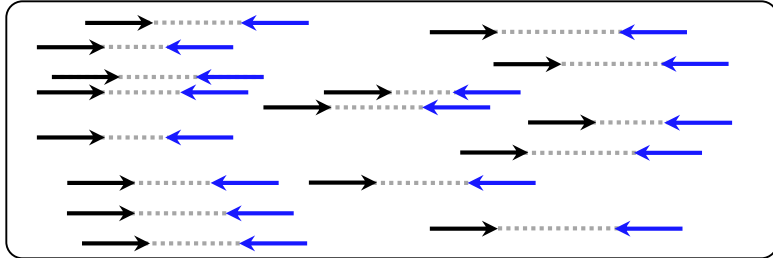


Gap close

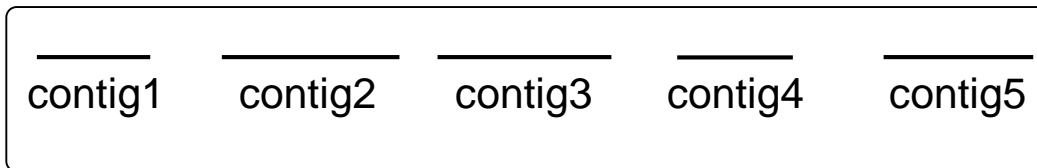


# 用語: gap close

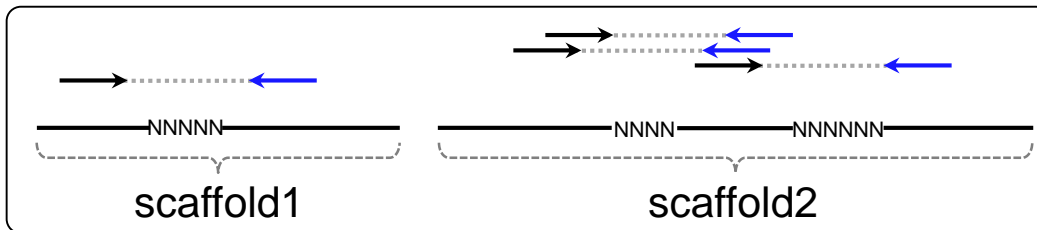
入力: paired-end FASTQファイル



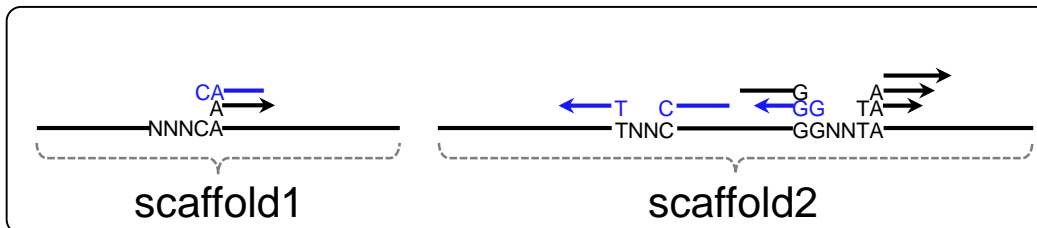
Assembly (コンティグの作成)



Scaffold



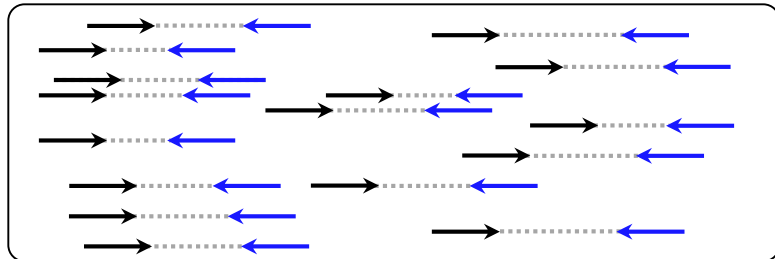
Gap close



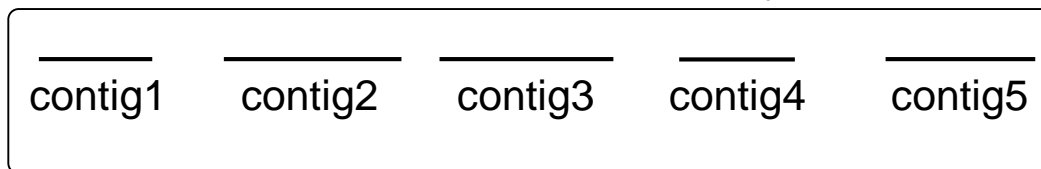
得られたscaffoldsにリードをマップし...①  
gap周辺にマップされたリードの塩基でNを  
置換。gapのNがなくなり、閉じていく(close)  
のでgap closeという(おそらく)。

# de novoアセンブリ

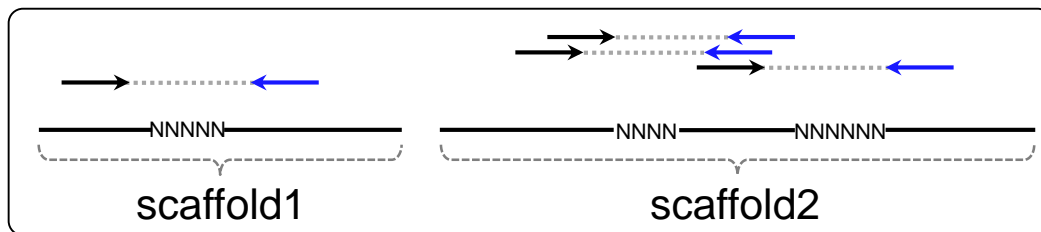
入力: paired-end FASTQファイル



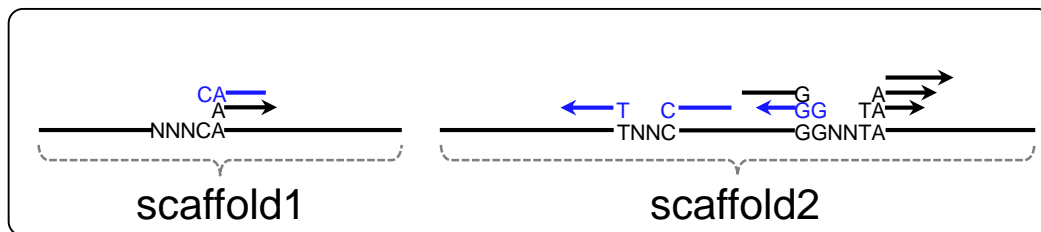
Step1: Assembly



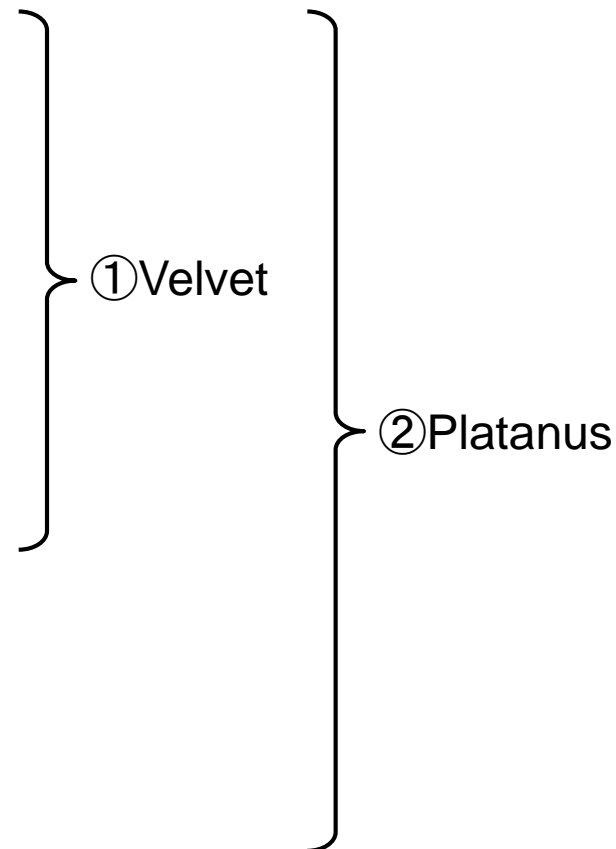
Step2: Scaffold



Step3: Gap close



①最も有名なNGSデータ用 *de novo* ゲノムアセンブリプログラムであるVelvet (Zerbino and Birney, Genome Res., 2008) は、Step2までを実行。②比較的最近開発されたPlatanus (Kajitani et al., Genome Res., 2014) は、Step3までを実行してくれる



# 乳酸菌論文は...

乳酸菌(*Lactobacillus hokkaidonensis* LOOC260<sup>T</sup>)ゲノム解読論文では、Illumina MiSeqデータ(DRR24501)の *de novo* アセンブリプログラムとして①Platanus (ver. 1.2)を利用している。

DRASearch

Send Feedback Search Home DRA Home

DRR024501 FASTQ SRA

Run Detail	
Alias	DRR024501
Instrument model	
Date of run	
Run center	
Number of spots	2,971,310
Number of bases	1,491,597,620

Navigation

- Submission DRA002643 FTP
- Study DRP002401

READS (joined) quality show 10 rows << < 1

```
>DRR024501.1
ATGNATCGAAACAGTATTTACAAGATTTGCATACTGAAATTTGAAGCTGATCAACACGAAACCATTC
AATCTAAACACCCATTAGCTGTTATTGAAGCTTTGCAGCAACGAGTTGATGATAAAATGACCGTTT
GAGCCATTATATTTGGATGGCCCGGCACCTCCGAAGTTATGAGCCTCGCCATTTATTGTTTAGTAA
TTGGAGTGGCGATGAAACCGTATTAAGCCCTAAACGAACGGCTGCTCCAGTTCTTGTCCAGTAAAT
CCAGAAACAGAGACTGATTTAGCTATTGGGCCGAACTAACCGCAGCCGAAATTTGACCAAGGTAGCG
GCATCCCACTTAACAATAAATGGCGAGGCTCATAACTTCGGAAGTGCCGGGCCATCCAAATATA
TCAAACGCAACGTTTCATATTAT

>DRR024501.2
GTCNGAACACATGAATGGTGAACGGCGCTGAACTTTTACGGACGCGGCACGAGGATCCACAGGGC
AAACACGTCACAGACTTGTATCACCGCATTATTACGTGAGTGGATTTCTGATAAAACAGTGTAA
AATTGATTTCTAGCCAATCAAAGACAGATTTAACGAAATCACAGATGACCAAAATTTGGCATTGAGT
TAAAAAAGTGATGGACCCCTCTTTAACCCTAAGTTGTCCCGAATAACATTCGAAACTCTCTGCTTT
ACCTCAAATGATTTGCCCAATCAATTTGACTTGTTCCTTGTGCATGATCTGATTTCACTGTTTAT
ACTCAATGCCAATTTGGTCATCTGTGATTTCTGTTAACTCTGTCTTTGATTGGCTAGACATCGAA
TTCACACTGTTTTATACGAAAT

>DRR024501.3
CAANGATACAATCATTATCATGAACTCTAATGCCGGTTCTGCTGATGCACTGCTACTGTTGCGTGA
CTGGTGACACGCACTCAGTTCTTAAACAACTAGGCGATTAT
```

**Genome sequencing and *de novo* assembly**

The cells of *L. hokkaidonensis* LOOC260<sup>T</sup> were cultured in MRS (de Man, Rogosa, and Sharpe) broth (Difco) and were harvested in the mid-logarithmic phase. The genomic DNA was extracted and purified using Qiagen Genomic-tip 500/G and Qiagen Genomic DNA Buffer Set with lysozyme (Sigma) and proteinase K (Qiagen) according to the manufacturer's instruction. PacBio SMRT whole-genome sequencing was performed using a PacBio RSII sequencer with P4-C2 chemistry. Four SMRT cells were used for sequencing, thereby yielding 163,376 adapter-trimmed reads (subreads) with an average read length of approximately 4 kbp, which corresponded to approximately 250-fold coverage. *De novo* assembly was conducted using the HGAP method based on the SMRT Analysis package 2.0, which yielded seven contigs. Independent genome sequencing using the 250-bp paired-end Illumina MiSeq system generated 5,942,620 reads, which were assembled into contigs using Platanus assembler ver 1.2 with the default settings [40]. The initial contigs





# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# DDBJ Pipeline

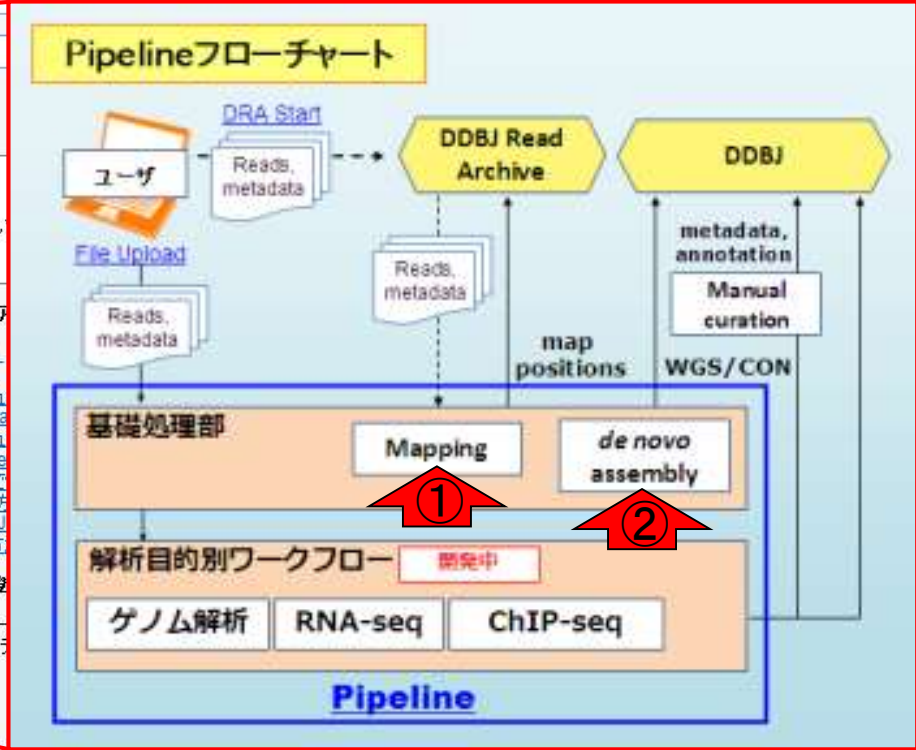
DDBJ Pipelineでは、主に①マッピングや② *de novo*アセンブリができる。特に後者ができるのは非常に有難い。③新規アカウント作成から*de novo*アセンブリまでの詳細については、乳酸菌連載第6回ウェブ資料を参照。ここでは、説明は必要最小限にして、Rのハンズオンへと移行する。

The screenshot shows the DDBJ Pipeline website interface. At the top, there is a navigation bar with the DDBJ logo and the title "DDBJ Read Annotation Pipeline". Below this, a description states: "DDBJ Read Annotation Pipelineは、次世代シーケンサ配列のクラウド型データ解析プラットフォームです。" (DDBJ Read Annotation Pipeline is a cloud-based data analysis platform for next-generation sequencing data).

The main content area is titled "LOGIN" and includes a "新規アカウント作成" (New Account Creation) link. There are input fields for "User ID:" and "Password:" with a "Login" button. Below the login fields, there is a section for "動作中JOBの確認" (Check running jobs) with a note: "PipelineのIDをお持ちでないことができます。" (You can check jobs without the Pipeline ID).

On the left side, there is a "Pipelineフローチャート" (Pipeline Flowchart) diagram. It shows the flow from "ユーザ" (User) through "DRA Start" (Reads, metadata) to "DDBJ Read Archive" (Reads, metadata) and finally to "DDBJ" (metadata, annotation, Manual curation). The "基礎処理部" (Basic Processing Unit) includes "Mapping" and "de novo assembly". Below this, there is a "解析目的別ワークフロー" (Workflow by analysis purpose) section with options for "ゲノム解析" (Genome analysis), "RNA-seq", and "ChIP-seq".

At the bottom left, there is a "Tweets" section with a tweet from @pipeline\_info: "DDBJ Pipeline e-mail support will not open from Dec 25 - Jan 3. Thank you for your cooperation."



# DDBJ PipelineでPlatanus

DDBJ Pipelineのプログラム選択画面。①Velvetや②Platanusを選択可能

Preprocessing Start

step-1  
Preprocessing

Mapping / de novo Assembly

step-2

**Workflow**

Genome (SNP/Short Indel)  
RNA-seq (Tag count)  
ChIP-seq

**JOB STATUS**

step1. Preprocessing

step1. Mapping

step1. de novo Assembly

step2-All status

**HELP**

HELP [?](#)

TUTORIAL

Contact Us.  
DDBJ Read Annotation Pipeline.  
Development Team.

Tool	Help	Version	Base space	Color space	Paired-end	MSS (WGS)	Comment
<input type="checkbox"/> <a href="#">BLAT</a>	<a href="#">?</a>	34	✓				Single-end analysis only
<input type="checkbox"/> <a href="#">bwa</a>	<a href="#">?</a>	0.6.1	✓	✓	✓	✓	
<input type="checkbox"/> <a href="#">Bowtie</a>	<a href="#">?</a>	0.12.7	✓	✓	✓	✓	
<input type="checkbox"/> <a href="#">TopHat</a>	<a href="#">?</a>	1.0.11	✓	✓	✓	✓	
<input type="checkbox"/> <a href="#">Bowtie2</a>	<a href="#">?</a>	2.0.0	✓	✓	✓	✓	For reads longer than about 50 bp, Bowtie2 is generally faster, more sensitive, and uses less memory than Bowtie1.
<input type="checkbox"/> <a href="#">TopHat2</a>	<a href="#">?</a>	2.0.9	✓	✓	✓	✓	

**de novo Assembly**  
Total limit = 22 Gbp

Tool	Help	Version	Base space	Color space	Paired-end	MSS (WGS)	Comment
<input type="checkbox"/> <a href="#">SOAPdenovo</a>	<a href="#">?</a>	1.05	✓		✓		
<input type="checkbox"/> <a href="#">ABySS</a>	<a href="#">?</a>	1.3.2	✓		✓		Maximum K-mer value is 64.
<input type="checkbox"/> <a href="#">Velvet</a>	<a href="#">?</a>	1.2.10	✓		✓	✓	We severe recommend when performing Velvet, total length of those reads is up to 22G bp. Maximum K-mer value is 64.
<input type="checkbox"/> <a href="#">Trinity</a>	<a href="#">?</a>	r2013-02-25	✓		✓		RNA-Seq De novo Assembly
<input checked="" type="checkbox"/> <a href="#">Platanus</a>	<a href="#">?</a>	1.2.2	✓		✓		
<input type="checkbox"/> <a href="#">HGAP</a>	<a href="#">?</a>	Protocol3 (v 2.2.0)					HGAP Pipeline for PacBio Sequence based on SMRT Analysis v2.2.0. For bax.h5 file only. (Beta version)

**Mapping Contigs by de novo Assemble to Reference Sequences.**  
The contigs will be aligned to reference genome.

Tool	Comment
<input checked="" type="radio"/> BLAT	Single-end analysis only

DDBJ Pipeline: Nagasaki et al., *DNA Res.*, **20**: 383-390, 2013

Platanus: Kajitani et al., *Genome Res.*, **24**: 1384-1395, 2014

# DDBJ PipelineでPlatanus

De novoアセンブリの一般的な手順がわかっているならば、赤枠内のStep1-3の説明の意味がなんとなくわかる。①DDBJ Pipelineは基本的にボタンをポチポチ押していくだけ

ACCOUNT  
login ID [agribio]  
Logout  
Change password

ANALYSIS  
Data setup  
DRA Start  
FTP upload  
HTTP upload  
DRA Import  
Preprocessing Start  
step-1  
Preprocessing  
Mapping / de novo Assembly  
step-2  
Workflow  
Genome (SNP/Short Indel)  
RNA-seq (Tag count)  
ChIP-seq

JOB STATUS  
step1. Preprocessing  
step1. Mapping  
step1. de novo Assembly  
step2-All status

HELP  
HELP  
TUTORIAL  
Contact Us.  
DDBJ Read Annotation

Setting for De Novo Assembly

platanus

Set optional parameters of the paired-end analysis

Memory Usage :  Low (recommended)  High

If you request "High" memory usage during the time Nig super computer system is congested, you might be kept waiting long before job starts running,

Step1) Assembly : Construct contigs using the algorithm based on the de Bruijn graph.

platanus assemble -t 15 -m 120 -o out [ ] -f PE1.fastq PE2.fastq

Step2) Scaffold : Map paired-end (mate-pair) reads on contigs and construct scaffolds

platanus scaffold -t 8 -o out [ ]  
-c out\_contig.fa -b out\_contigBubble.fa -IP1 PE1.fastq PE2.fastq (-OP2 MP1.fastq MP2.fastq)

Step3) Gap Close : Map paired-end (mate-pair) reads on scaffolds and assemble reads on gaps and close gaps

platanus gap\_close -t 8 -o out [ ]  
-c out\_scaffold.fa -IP1 PE1.fastq PE2.fastq (-OP2 MP1.fastq MP2.fastq)

Step5) Create assembled sequences in FASTA file from pileupped reads to [submit WGS division of DDBJ.](#)

Set filtered length for contigs  
 perl lengthfilter.pl pileupFile [100] out\_WGS.txt

BACK NEXT

DDBJ Pipeline: Nagasaki et al., *DNA Res.*, **20**: 383-390, 2013

Platanus: Kajitani et al., *Genome Res.*, **24**: 1384-1395, 2014

# DDBJ PipelineでPlatanus

アセンブリ終了後の画面。①  
Platanus実行結果ファイル  
(platanusResult.zip)をダウンロ  
ードして解凍したのが...

**ANALYSIS**

Data setup

- DRA Start
- FTP upload
- HTTP upload
- DRA Import
- Preprocessing Start

step-1

- Preprocessing
- Mapping / *de novo* Assembly

step-2

**Workflow**

- Genome (SNP/Short Indel)
- RNA-seq (Tag count)
- ChIP-seq

**JOB STATUS**

- step1. Preprocessing
- step1. Mapping
- step1. *de novo* Assembly
- step2-All status

**HELP**

- HELP
- TUTORIAL
- Contact Us. DDBJ Read Annotation Pipeline. Development Team.

**Job info**

ID: 21211

Tool (Version): Platanus (1.2.2)

RunAccession or Filename	Download	Read length	Alias
QC.1.trimmed.fastq.gz	<a href="#">QC.1.trimmed.fastq.gz</a>	N.A. bp	L.hokkaidonensis_MiSeq_denovo

**Download modified queries**

- [QC.1.trimmed.fastq.gz](#) (Original size 189.4 MB)
- [QC.2.trimmed.fastq.gz](#) (Original size 189.6 MB)

**Download wgs file**

- [out\\_WGS.fasta.gz](#) (Original size 2.3 MB)

**Assembly statistics**

Contig # : 117  
Total contig size : 2,356,019  
Maximum contig size : 257,728  
Minimum contig size : 101  
N50 contig size : 92,304

**Time**

Wait time	Start time	End time
0: 0:9	2016-01-20 18:33:36	2016-01-21 10:10:06

Command	Start time	End time	Log1	Log2	Result	MD5
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>

BACK

Top of page

# DDBJ PipelineでPlatanus

アセンブリ終了後の画面。①  
Platanus実行結果ファイル  
(platanusResult.zip)をダウンロ  
ードして解凍したのが…②hoge  
フォルダ中のplatanusResult。

The screenshot shows the 'Detail view' of a Platanus assembly job. The left sidebar contains navigation menus for 'ANALYSIS', 'JOB STATUS', and 'HELP'. The main content area is divided into several sections:

- Job info:** ID 21211, Tool (Version) Platanus (1.2.2).
- RunAccession or Filename:** QC.1.trimmed.fastq.gz, Download [QC.1.trimmed.fastq.gz](#), Read length N.A. bp, Alias L.hokkaidonensis\_MiSeq\_denovo.
- Download modified queries:**
  - [QC.1.trimmed.fastq.gz \(Original size 189.4 MB\)](#)
  - [QC.2.trimmed.fastq.gz \(Original size 189.6 MB\)](#)
- Download wgs file:**
  - [out\\_WGS.fasta.gz \(Original size 2.3 MB\)](#)
- Assembly statistics:** Contig #, Total contig size : 2,356, Maximum contig size : 257, Minimum contig size, N50 contig size : 92.
- Time:** Wait time 0: 0:9, Start time 2016-01-20 18:33:36, End time 2016-01-21 10:10:06.
- Command Log:**

Command	Start time	End time	Log1	Log2	Result	MD5
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>

At the bottom right, there is a 'BACK' button and a 'Top of page' dropdown menu. A red arrow labeled '1' points to the 'Download(2.2 MB)' link in the Command Log table.

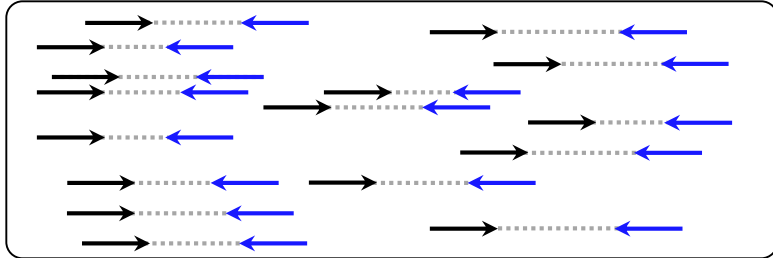
The screenshot shows a file explorer window with the address bar set to 'hoge > platanusResult'. The window title is 'platanusResult' with a red arrow labeled '2' pointing to it. The file list is as follows:

名前	サイズ
out_32merFrq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubble.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubble.fa	0 KB
out_scaffoldComponent.tsv	5 KB

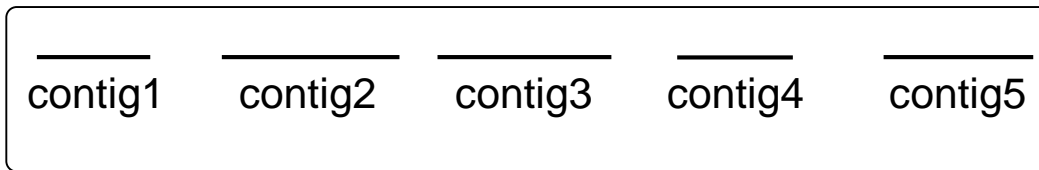
At the bottom right of the file explorer, there is a red arrow labeled '1' pointing to the 'out\_scaffoldComponent.tsv' file.

# DDBJ PipelineでPlatanus

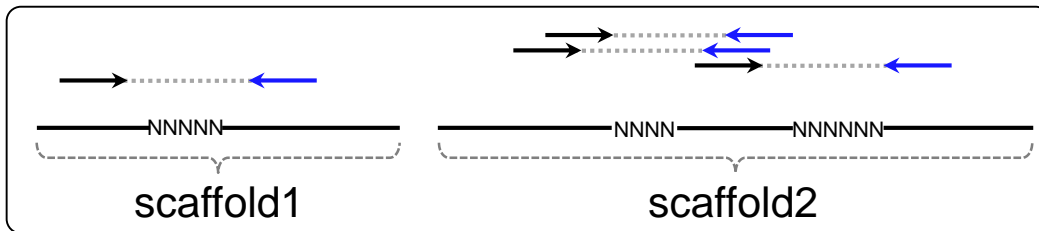
入力: paired-end FASTQファイル



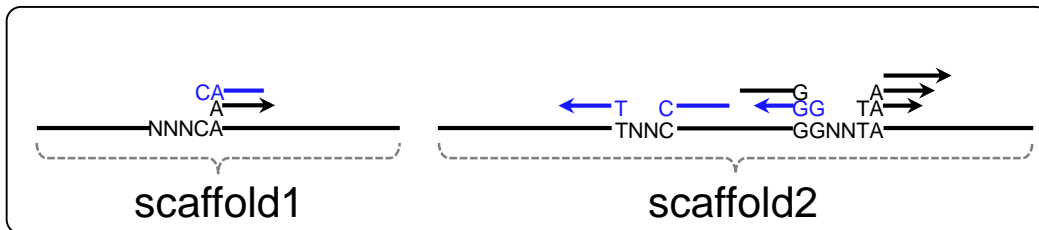
Step1: Assembly



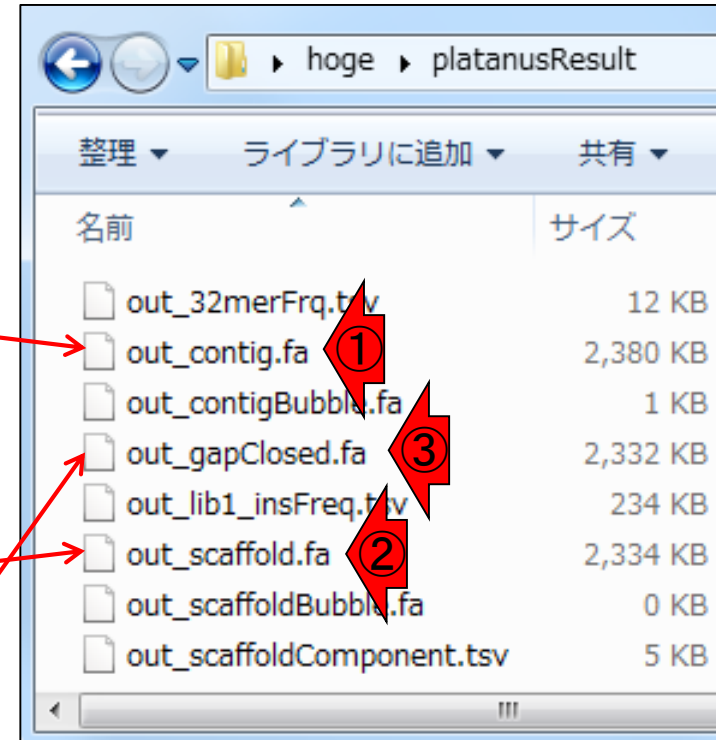
Step2: Scaffold



Step3: Gap close



一般的な *de novo* アセンブリの手順を知っておけば、ファイル名から最終的な結果が③out\_gapClosed.faだと認識できる。



# DDBJ PipelineとR

## ■ 解析受託企業に外注: Linuxコマンドを

-  ngs 受託解析

## ■ クラウド(ウェブツール): Linuxコマンドを

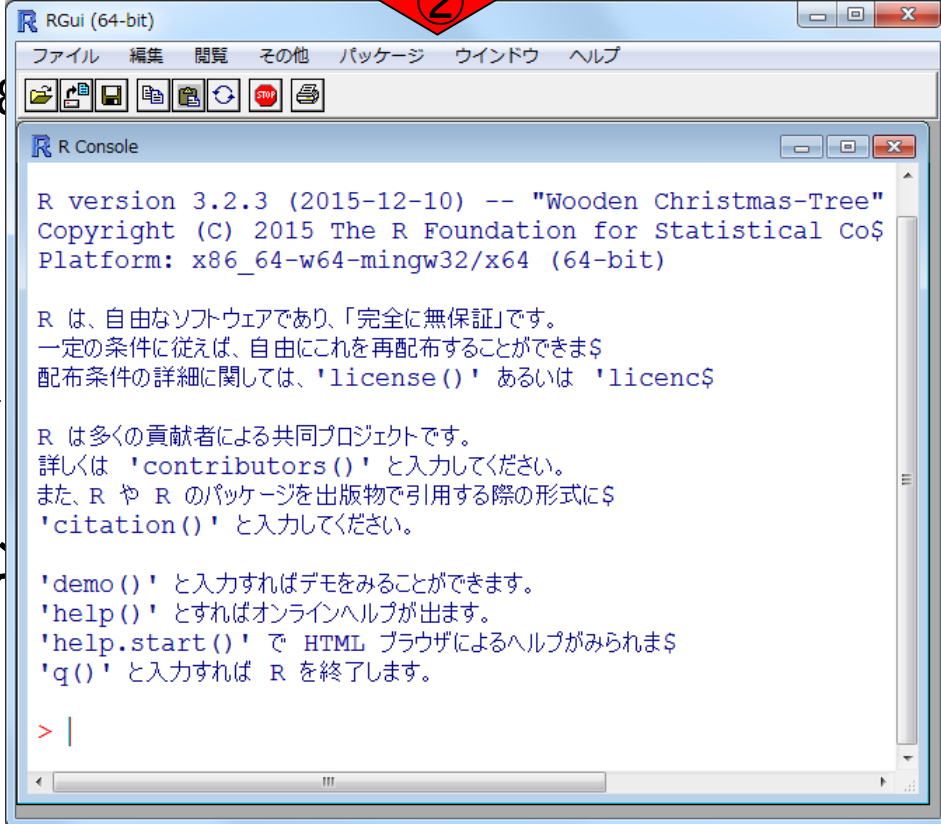
- ① □ DDBJ Pipeline (Nagasaki et al., *DNA Res.*, **20**: 383–390, 2013)

- Illumina BaseSpace
- Galaxy (Goecks et al., *Genome Biol.*, 11: R85–R90, 2010)
- ...

## ■ Linuxコマンドを駆使(旧来型)

- なるべく自力で解析
- LinuxコマンドやNGS解析用プログラムのインストール/アップデート/バグ修正/トラブルシューティング/スクリプトの作成/スクリプトの修正/スクリプトのデバッグ/スクリプトの最適化/スクリプトの共有/スクリプトの再利用/スクリプトの自動化/スクリプトの保守/スクリプトのドキュメント作成/スクリプトのテスト/スクリプトのデバッグ/スクリプトの最適化/スクリプトの共有/スクリプトの再利用/スクリプトの自動化/スクリプトの保守/スクリプトのドキュメント作成/スクリプトのテスト
- NBDC/東大アグリバイオ/HPCIの「NGSハ

①DDBJ Pipelineだけで全てのNGS解析ができるわけではない。②Rもまた然り。特にRでは、(門田の知る限り) *de novo*アセンブリは不可能。現実を知り、うまく使い分けるべし。DDBJ Pipeline上で*de novo*アセンブリを行った結果の解釈や確認をRで行い、塩基配列解析基礎のスキルがあつてよかつたと思つた実例を紹介。



```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

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

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

> |
```



# Contents1

## ■ イン트로ダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

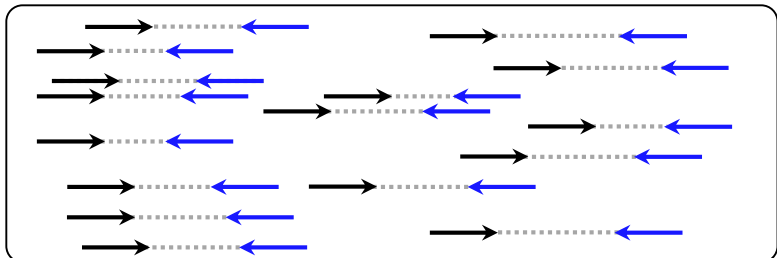
## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得

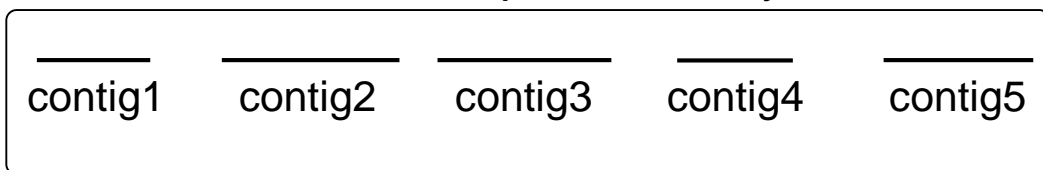


# 塩基配列解析基礎1

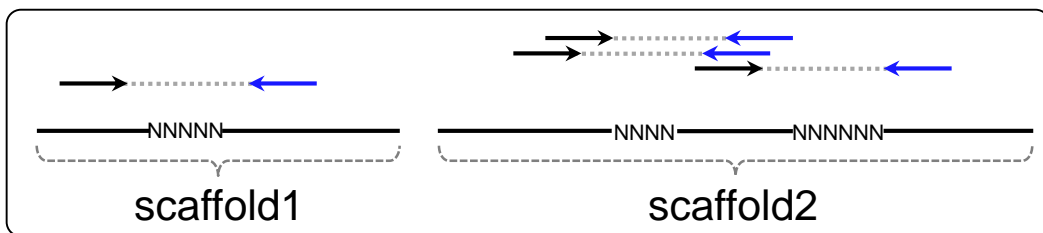
入力: paired-end FASTQファイル



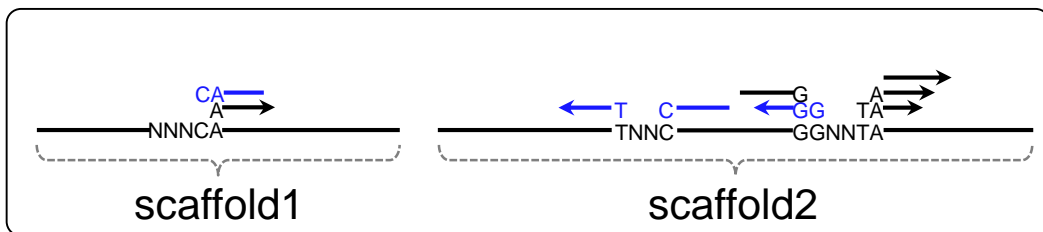
Step1: Assembly



Step2: Scaffold



Step3: Gap close



(アセンブリ実行結果の) multi-FASTAファイルを読み込んで、塩基ごとの出現頻度解析ができる。①Step1実行後 (out\_contig.fa) はNがなく、②Step2実行後 (out\_scaffold.fa) にNができて、③Step3実行後 (out\_gapClosed.fa) にNが減るのだろうと妄想できる。それを自力で確認することで、アルゴリズムの理解を深めることができる。

名前	サイズ
out_32merFreq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubbles.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubbles.fa	0 KB
out_scaffoldComponent.tsv	5 KB

# 塩基ごとの出現頻度解析

① (アセンブリ実行結果の) multi-FASTAファイルを読み込んで、塩基ごとの出現頻度解析を行う項目

## (Rで)塩基配列解析

～NGS, RNA-seq, ゲノム, トランスクリプトーム, 正規化, 発現変動, 統計, モデル, バイオインフォマティクス～  
(last modified 2016/02/03, since 2011)

### What's new?

- このウェブページはフリーソフト Rと必要 [利用法\(Windows20書籍\)](#)もあります。(2016/02/03)
- 多群間比較用の推定については [門田](#)のページに検出結果のおおよそをイン周辺の関連項目
- Erratum.** 2014.06.22には、kの値を小さくはいはいいです。(見てきたm(\_)\_m(2016/02/03))

- ・ [イントロ](#) | [一般](#) | [翻訳配列\(translate\)を取得\(応用\)](#) | [seqinr\(Charif 2005\)](#) (last modified 2015/03/09)
- ・ [イントロ](#) | [一般](#) | [相補鎖\(complement\)を取得](#) (last modified 2013/06/14)
- ・ [イントロ](#) | [一般](#) | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2013/06/14)
- ・ [イントロ](#) | [一般](#) | [逆鎖\(reverse\)を取得](#) (last modified 2013/06/14)
- ・ [イントロ](#) | [一般](#) | [k-mer解析 | k=1\(塩基ごとの出現頻度解析\)](#) | [Biostrings](#) (last modified 2016/02/03) **NEW**
- ・ [イントロ](#) | [一般](#) | [k-mer解析 | k=2\(2連続塩基の出現頻度解析\)](#) | [Biostrings](#) (last modified 2016/01/28) **NEW**
- ・ [イントロ](#) | [一般](#) | [k-mer解析 | k=3\(3連続塩基の出現頻度解析\)](#) | [Biostrings](#) (last modified 2016/01/28) **NEW**
- ・ [イントロ](#) | [一般](#) | [k-mer解析 | k=n\(n連続塩基の出現頻度解析\)](#) | [Biostrings](#) (last modified 2016/01/28) **NEW**
- ・ (削除予定) [イントロ](#) | [一般](#) | [2連続塩基の出現頻度情報を取得](#) (last modified 2015/04/20)
- ・ (削除予定) [イントロ](#) | [一般](#) | [3連続塩基の出現頻度情報を取得](#) (last modified 2015/02/19)
- ・ (削除予定) [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 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | Biostrings **NEW**

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基ごとの出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

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

1. [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: 配列ごとに出現頻度をカウントした結果を返すやり方です。

```

in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "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
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をoutに格納
#outの中身を表示
    
```

# 塩基ごとの出現頻度解析

①例題7が、PlatanusのStep3実行後のファイル(②out\_gapClosed.fa)を入力とするものなので、そのままコピペできて便利。これを実行します。

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基ごとの出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

「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:

配列ごとに出現頻度をカウント



```
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)

#本番
out <- alphabetFrequency(fasta)

#ファイルに保存
tmp <- cbind(names(fasta), out)
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)
```

2. FASTA形式ファイル(out\_gapClosed.fa)の場合:

[DDBJ Pipeline \(Nagasaki et al., DNA Res., 2013\)](#)上で de novoゲノムアセンブリプログラム [Platanus \(Kajitani et al., Genome Res., 2014\)](#) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。



```
in_f <- "out_gapClosed.fa"
out_f <- "hoge7.txt"
param_base <- c("A", "C", "G", "T", "N")

#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)
obj <- is.element(colnames(hoge), param_base)
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), 2, sum)

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

# 塩基ごとの出現頻度解析

イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | Biostrings

つまり、Platanus実行結果ファイル (platanusResult.zip)をダウンロードし、解凍して得られた①platanusResultフォルダ中の②out\_gapClosed.faを入力として、塩基ごとの出現頻度解析を行う

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基の出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

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

1. 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:

配列ごとに出現頻度をカウントした

```
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)

#本番
out <- alphabetFrequency(fasta)

#ファイルに保存
tmp <- cbind(names(fasta), out)
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)
```

7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

[DDBJ Pipeline \(Nagasaki et al., DNA Res., 2013\)](#)上で de novoゲノムアセンブリを実行して得られたmulti-FASTA形式ファイル(hoge7.txt)を入力として、塩基ごとの出現頻度を調べるやり方を示します。

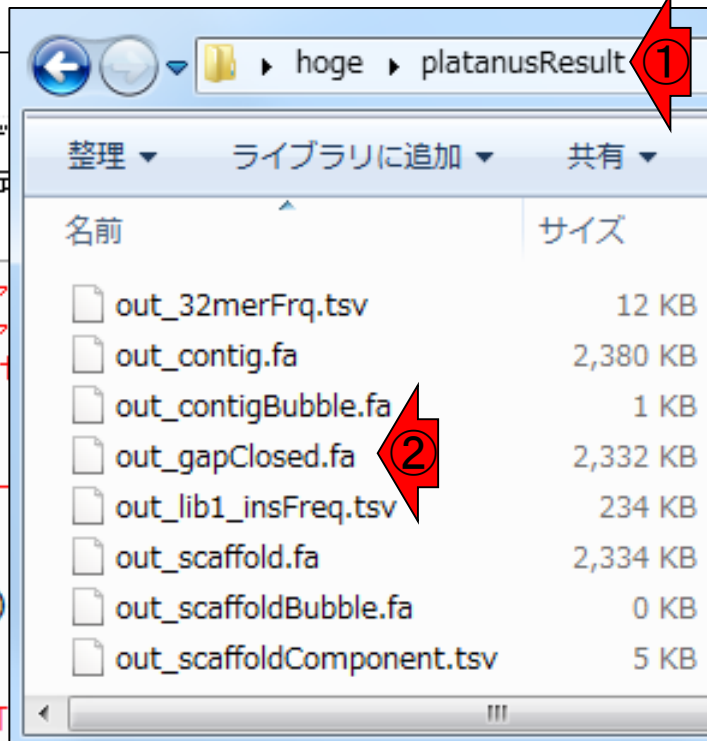
```
in_f <- "out_gapClosed.fa" #入力ファイル
out_f <- "hoge7.txt" #出力ファイル
param_base <- c("A", "C", "G", "T", "N") #出力する塩基

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

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

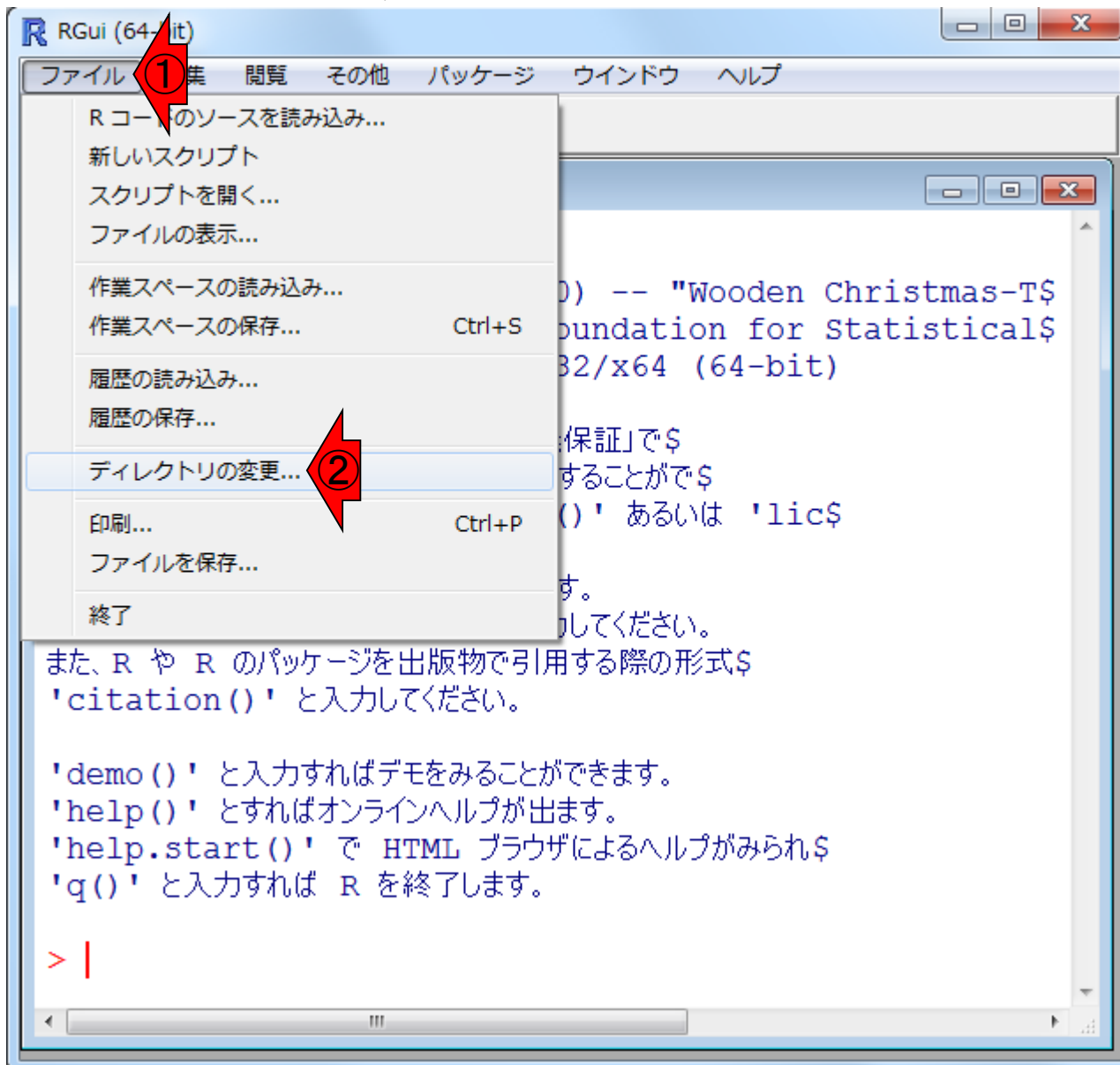
#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,Nの出現頻度を調べる
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納

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



# Rの起動と作業ディレクトリ変更

①ファイル、②ディレクトリの変更。③「デスクトップ - hoge - platanusResult」を指定する。



The screenshot shows the R GUI window titled "RGui (64-bit)". The "ファイル" (File) menu is open, and the option "ディレクトリの変更..." (Change Directory...) is highlighted with a red arrow labeled "2". A red arrow labeled "1" points to the "ファイル" menu itself. The console window shows the following text:

```
R コーのソースを読み込み...
新しいスクリプト
スクリプトを開く...
ファイルの表示...

作業スペースの読み込み...
作業スペースの保存... Ctrl+S
履歴の読み込み...
履歴の保存...

ディレクトリの変更...
印刷... Ctrl+P
ファイルを保存...
終了
```

また、R や R のパッケージを出版物で引用する際の形式 '\$citation()' と入力してください。

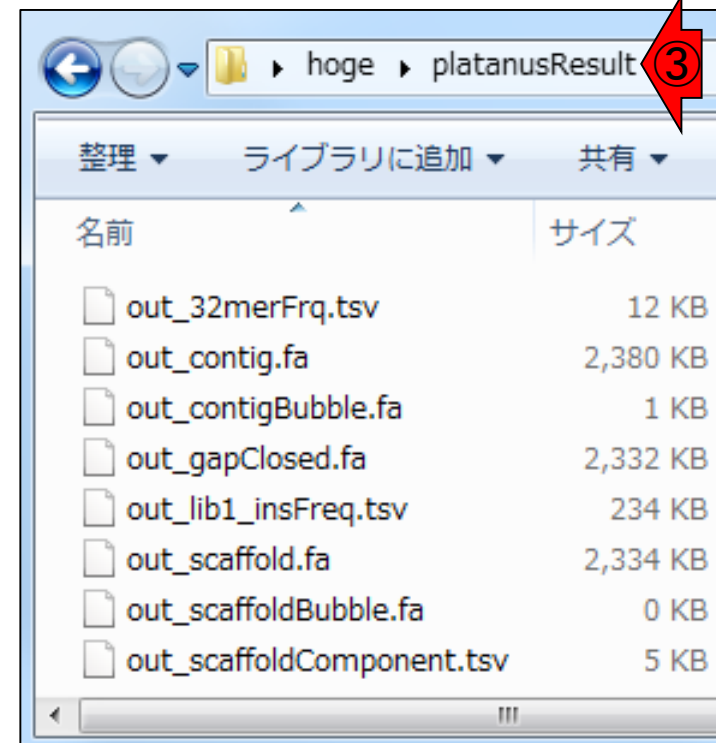
'demo()' と入力すればデモをみることができます。

'help()' とすればオンラインヘルプが出ます。

'help.start()' で HTML ブラウザによるヘルプがみられ\$

'q()' と入力すれば R を終了します。

> |

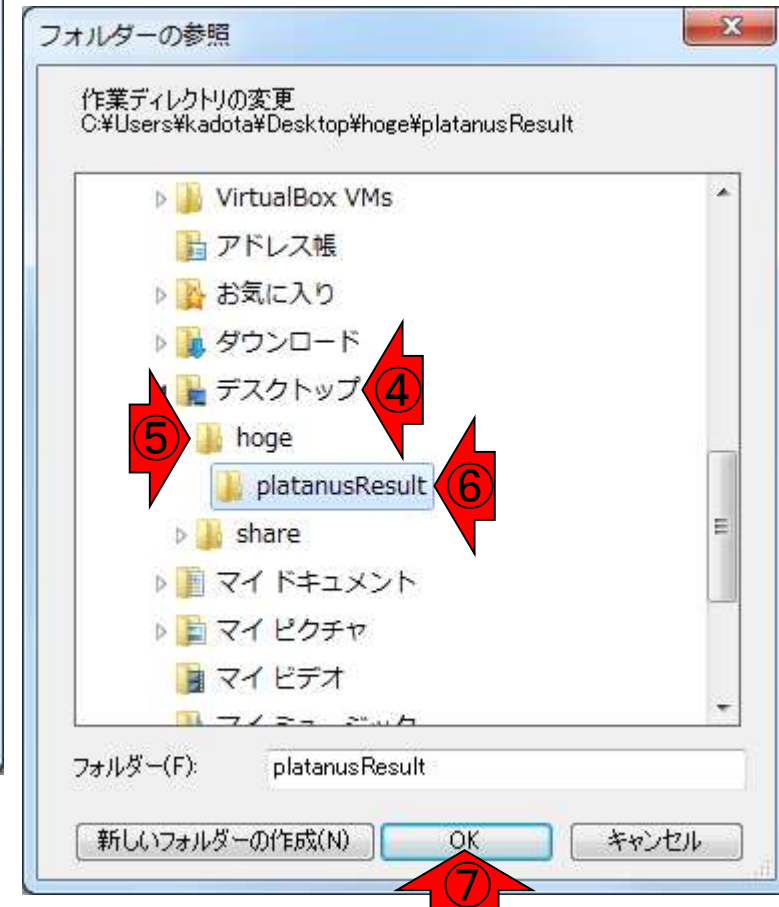
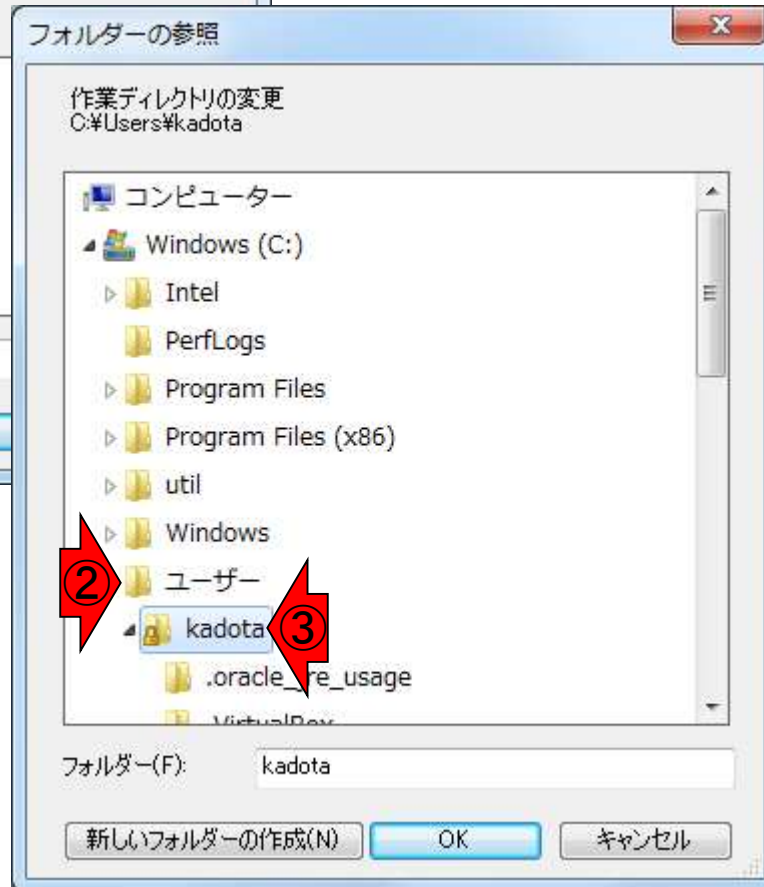
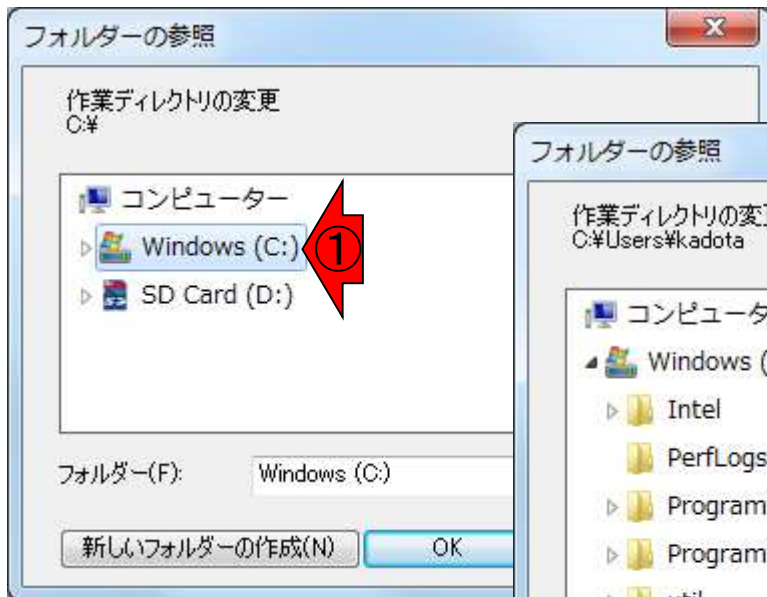


The screenshot shows a Windows Explorer window with the address bar displaying "hoge > platanusResult", indicated by a red arrow labeled "3". The window shows a list of files in the "platanusResult" directory:

名前	サイズ
out_32merFrq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubble.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubble.fa	0 KB
out_scaffoldComponent.tsv	5 KB

# 作業ディレクトリ変更

ヒトによって若干見栄えは違うだろうが、⑤-⑦が同じになればよい。



# getwd()

作業ディレクトリ変更の確認です。  
①getwd()と打ち込んで確認。②の  
のように見えていればOK

```
R Console

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

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

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

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

名前	サイズ
out_32merFrq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubble.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubble.fa	0 KB
out_scaffoldComponent.tsv	5 KB



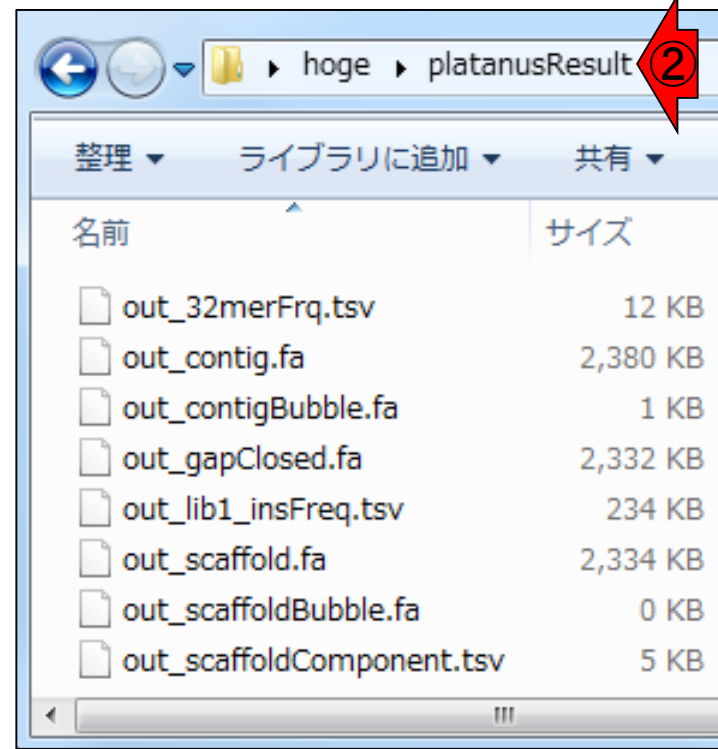
# list.files()

R上で、現在の作業ディレクトリ中の  
ファイルを眺めるのが①list.files()。  
②GUI画面上で眺めている  
platanusResultフォルダ中のものと同じ  
ものが見えていることがわかる。

```
R Console

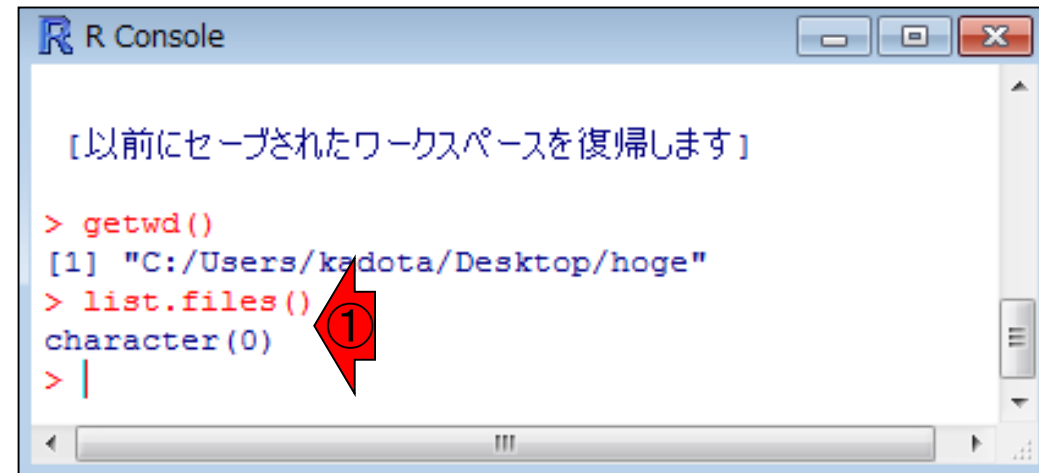
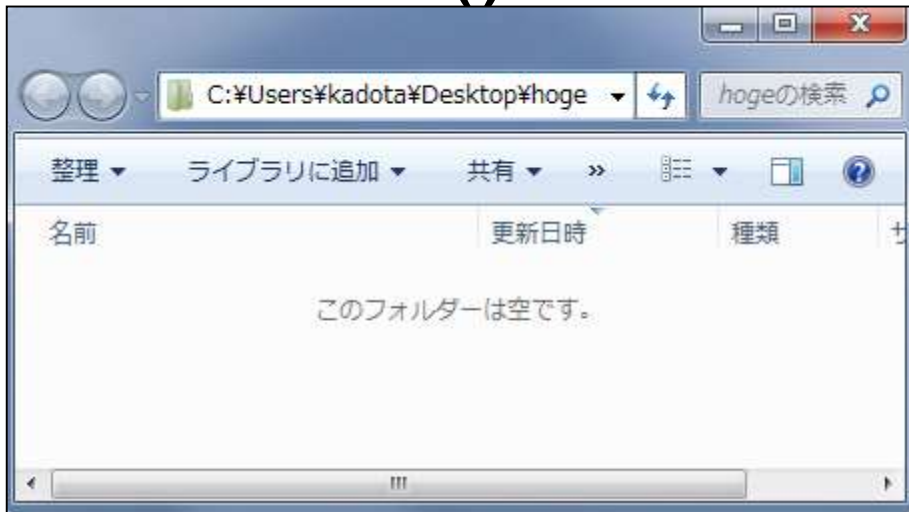
'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみら$
'q()' と入力すれば R を終了します。

> getwd()
[1] "C:/Users/①dota/Desktop/hoge/platanusResult"
> list.files()
[1] "out_32merFrq.tsv"
[2] "out_contig.fa"
[3] "out_contigBubble.fa"
[4] "out_gapClosed.fa"
[5] "out_lib1_insFreq.tsv"
[6] "out_scaffold.fa"
[7] "out_scaffoldBubble.fa"
[8] "out_scaffoldComponent.tsv"
> |
```



# list.files()

①ファイルが存在しないフォルダ上で、list.files()とやると、character(0)という結果になる。



```
R Console

[以前にセーブされたワークスペースを復帰します]

> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
character(0)
```

A screenshot of an R Console window. The text "[以前にセーブされたワークスペースを復帰します]" is displayed at the top. The console shows the following commands and output:

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
character(0)
```

A red lightning bolt icon with the number "1" inside is positioned next to the output "character(0)".

# 塩基ごとの出現頻度解析

当たり前ですが、解析したいディレクトリ(またはフォルダ)を正しく指定できなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます。今は①解析したい入力ファイル(out\_gapClosed.fa)が、②R Console画面上でも③見えているのでエラーなく動くはずですよ。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4M)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,...
obj <- is.element(colnames(hoge), param_base) #条件
#out <- colSums(hoge[, obj]) #列ごとの総和
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごと

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

```
R Console
'demo()' と入力すればデモをみるすることができます。
'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$
'q()' と入力すれば R を終了します。

> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files()
[1] "out_32merFrq.tsv"
[2] "out_contig.fa"
[3] "out_contigBubble.f"
[4] "out_gapClosed.fa"
[5] "out_lib1_insFreq.tsv"
[6] "out_scaffold.fa"
[7] "out_scaffoldBubble.fa"
[8] "out_scaffoldComponent.tsv"
> |
```

# 基本はコピペ

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

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)をダウンロードし、Rで読み込みます。

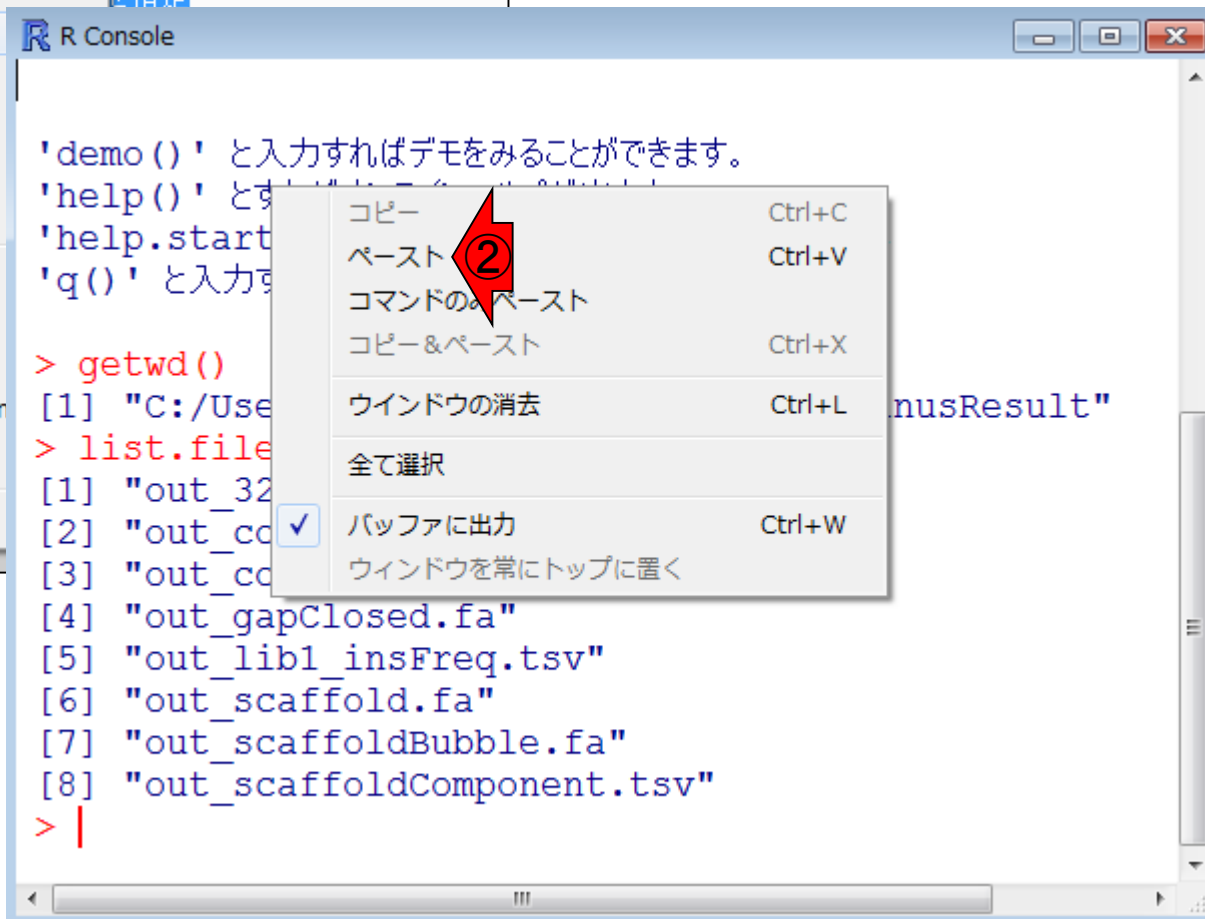
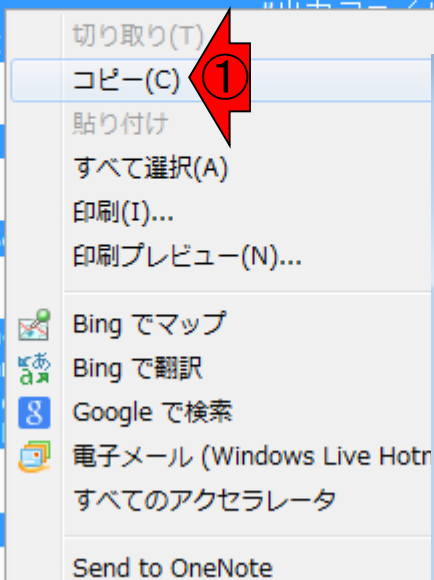
```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T") #塩基を指定
```

```
#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)
```

```
#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), param_base)
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), MARGIN=2, FUN=function(x) sum(x))
```

```
#ファイルに保存
write.table(out, out_f, row.names=FALSE, col.names=FALSE, as.is=TRUE)
```



# 途中経過と終了後

①コピー直後と②実行後の状態。エラーなく実行できたときはこんな感じになります。一見何も変化がないように見えますが…

```
R Console
> list.files()
[1] "out_32merFrq.tsv"
[2] "out_contig.fa"
[3] "out_contigBubble.fa"
[4] "out_gapClosed.fa"
[5] "out_lib1_insFreq.tsv"
[6] "out_scaffold.fa"
[7] "out_scaffoldBubble.fa"
[8] "out_scaffoldComponent.tsv"
> in_f <- "out_gapClosed.fa"
> out_f <- "hoge7.txt"
> param_base <- c("A", "C", "G", "T",
>
>
> #必要なパッケージをロード
> library(Biostrings)
要求されたパッケージ BiocGenerics をロード中です
要求されたパッケージ parallel をロード中です
```



```
R Console
要求されたパッケージ S4Vectors をロード中です
要求されたパッケージ stats4 をロード中です
要求されたパッケージ IRanges をロード中です
要求されたパッケージ XVector をロード中です
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")#$
>
> #本番
> hoge <- alphabetFrequency(fasta) #A,C,G,T,$
> obj <- is.element(colnames(hoge), param_base) #条$
> #out <- colSums(hoge[, obj]) #列ごとの$
> out <- apply(as.matrix(hoge[, obj]), 2, sum) #列$
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quot$
> |
```



# 結果の解説

①解析結果(塩基ごとの出現頻度情報)はhoge7.txtというファイルに保存されている。②list.files()とやると、確かに自分が出カファイル名として指定した③hoge7.txtが存在することがわかる

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPl et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力

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

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

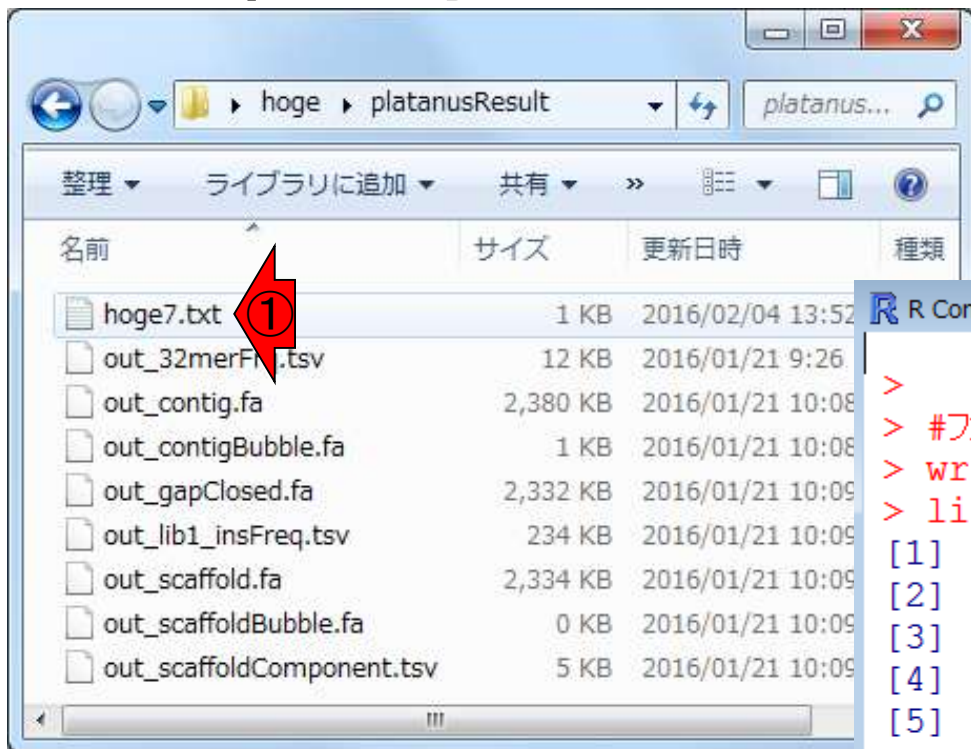
#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,N
obj <- is.element(colnames(hoge), param_base) #条ごとの$
#out <- colSums(hoge[, obj]) #列ごとの$
out <- apply(as.matrix(hoge[, obj]), 2, sum)

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

```
R Console
> hoge <- alphabetFrequency(fasta) #A,C,G,T,N
> obj <- is.element(colnames(hoge), param_base) #条ごとの$
> #out <- colSums(hoge[, obj]) #列ごとの$
> out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの$
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F)
> list.files()
[1] "hoge7.txt"
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> |
```

# 結果の解説

①もちろん出力ファイル(hoge7.txt)は手の届く場所(つまり作業ディレクトリ内)にある。②getwd()や、③現在時刻を表示するdate()はただの確認用。④エクセルで眺めるとこんな感じ。



```
R Console
>
> #ファイルに保存
> write.table(out, out_f, sep="\n")
> list.files()
[1] "hoge7.txt"
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> date()
[1] "Thu Feb 04 14:27:38 2016"
> |
```

Excel spreadsheet showing the contents of 'hoge7.txt'.

	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0



# R上で眺める

①赤枠程度の情報量なら、エクセルなどをわざわざ開くまでもなく、R上で眺めればよい。例えば、ここでは②出力ファイル名をout\_fというオブジェクト名で取り扱っている。③out\_fと打てば対応関係がわかる。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラム (et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)を眺めます。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力
```

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

```
#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
obj <- is.element(colnames(hoge), param_base) #列ごと
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), 2, sum)#
```

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

```
R Console
> write.table(out, out_f, sep="\t")
> list.files()
[1] "hoge7.txt"
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> date()
[1] "Thu Feb 04 14:27:38 2016"
> out_f
[1] "hoge7.txt"
> |
```

hoge7.txt - Excel

	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0





# R上で眺める

Rコードの最後の部分が、ファイルに保存するところ。①out\_fに書き込んでいるのは、②outというオブジェクトの情報。③outの中身を見ればhoge7.txtと同じ情報を得られる

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力
```

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

```
#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
obj <- is.element(colnames(hoge), param_base) #列ごと
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), 2, sum)#
```

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



```
R Console
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> date()
[1] "Thu Feb 04 14:27:38 2016"
> out_f
[1] "hoge7.txt"
> out
      A      C      G      T      N
729772 458211 440585 727451      0
> |
```

	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0

# sumで総塩基数を得る

①outオブジェクトは、数値ベクトル。②sumは、数値ベクトルの総和を計算する関数。outに対して実行した結果(2,356,019)は、入力ファイル(out\_gapClosed.fa)の総塩基数を調べていることと同義。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラム (et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)を処理する。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力
```

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

```
#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
obj <- is.element(colnames(hoge), param_base) #列ごと
out <- apply(as.matrix(hoge[, obj]), 2, sum) #
```

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

```
R Console
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> getwd()
[1] "C:/Users/kadota/Desktop/hog
> date()
[1] "Thu Feb 04 14:27:38 2016"
> out_f
[1] "hoge7.txt"
> out
      A      C      G      T      N
729772 458211 440585 727451      0
> sum(out)
[1] 2356019
> |
```

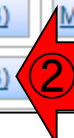
	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0



# sumで総塩基数を得る

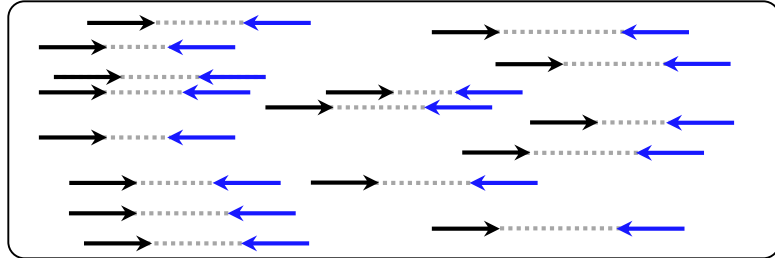
①DDBJ Pipeline実行結果画面上の数値と同じ。②入力ファイル(out\_gapClosed.fa)は、DDBJ Pipeline上でPlatanusという*de novo*アセンブリプログラムを実行した結果だったことを思い出そう。

ID		21211				
Tool (Version)		Platanus (1.2.2)				
RunAccession or Filename	Download	Read length	Alias			
QC.1.trimmed.fastq.gz	<a href="#">QC.1.trimmed.fastq.gz</a>	N.A. bp	L.hokkaidonensis_MiSeq_denovo			
Download modified queries						
<ul style="list-style-type: none"> <li><a href="#">QC.1.trimmed.fastq.gz (Original size 189.4 MB)</a></li> <li><a href="#">QC.2.trimmed.fastq.gz (Original size 189.6 MB)</a></li> </ul>						
Download wgs file						
<ul style="list-style-type: none"> <li><a href="#">out_WGS.fasta.gz (Original size 2.3 MB)</a></li> </ul>						
Assembly statistics						
		Contig # : 117 Total contig size : 2,356,019 Maximum contig size : 257,728 Minimum contig size : 101 N50 contig size : 92,304				
Time						
Wait time	Start time	End time				
0: 0:9	2016-01-20 18:33:36	2016-01-21 10:10:06				
Command	Start time	End time	Log1	Log2	Result	MD5
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>

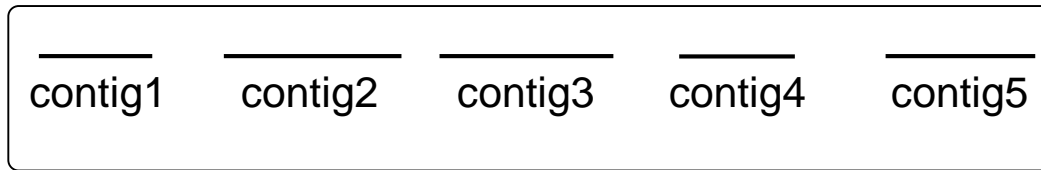


# 目的をおさらい

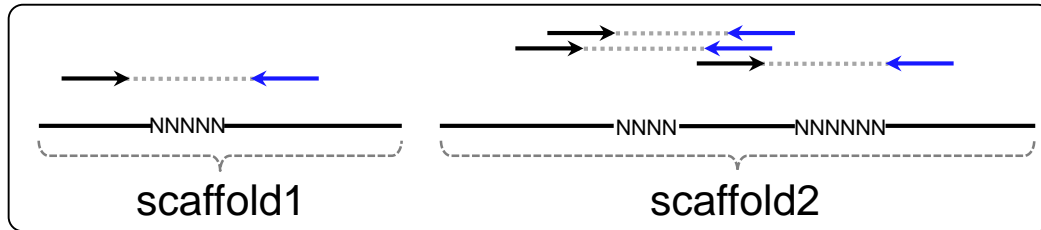
入力: paired-end FASTQファイル



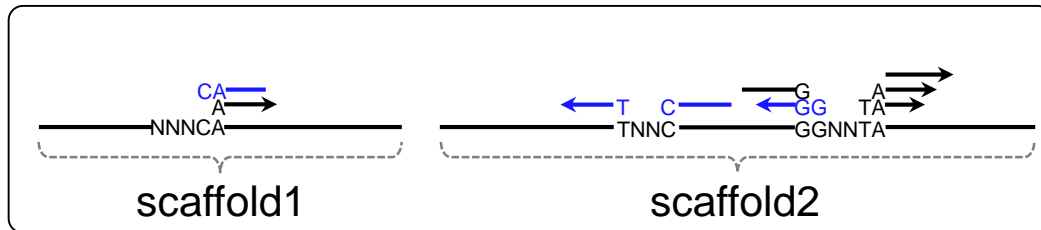
Step1: Assembly



Step2: Scaffold



Step3: Gap close

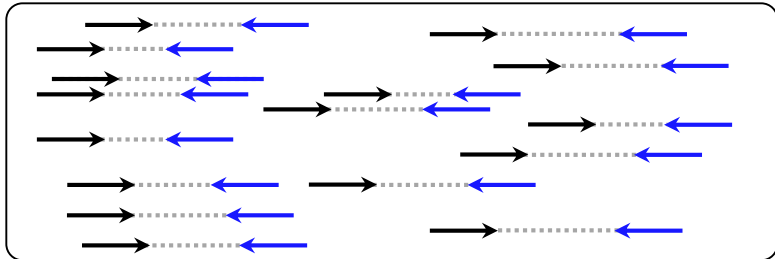


(アセンブリ実行結果の) multi-FASTAファイルを読み込んで、塩基ごとの出現頻度解析ができる。①Step1実行後 (out\_contig.fa) はNがなく、②Step2実行後 (out\_scaffold.fa) にNができて、③Step3実行後 (out\_gapClosed.fa) にNが減るので、それを自力で確認することで、アルゴリズムの理解を深めることができる。

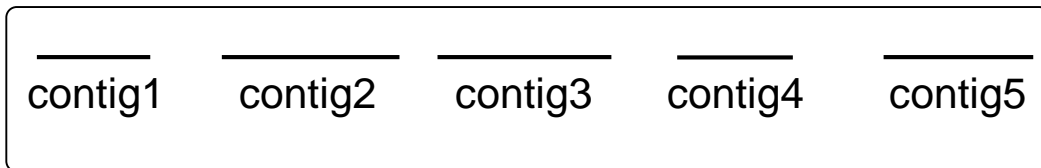
名前	サイズ
out_32merFreq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubbles.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubbles.fa	0 KB
out_scaffoldComponent.tsv	5 KB

# 目的をおさらい

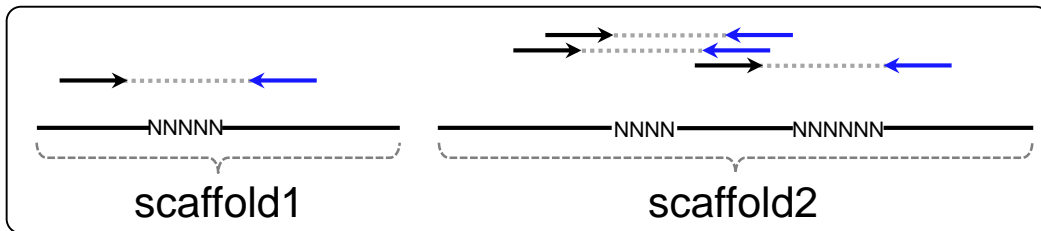
入力: paired-end FASTQファイル



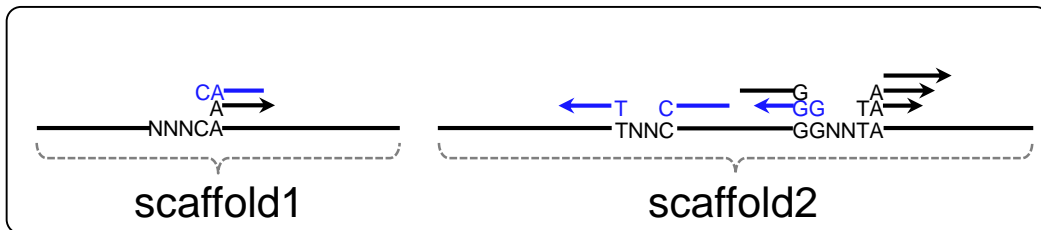
Step1: Assembly



Step2: Scaffold



Step3: Gap close



(アセンブリ実行結果の) multi-FASTAファイルを読み込んで、塩基ごとの出現頻度解析ができる。①Step1実行後(out\_contig.fa)はNがなく、②Step2実行後(out\_scaffold.fa)にNができて、③Step3実行後(out\_gapClosed.fa)にNが減るのだろうと妄想できる。それを自力で確認することで、アルゴリズムの理解を深めることができる。を調べるにはどうすればいいか？

名前	サイズ
out_32merFrq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubble.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubble.fa	0 KB
out_scaffoldComponent.tsv	5 KB

②

# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# 入力ファイルを変更

- ①テンプレートのout\_gapClosed.faを、
- ②out\_scaffold.faに変更すればよい

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

[DDBJ Pipeline \(Nagasaki et al., DNA Res., 2013\)](#)上で de novoゲノムアセンブリプログラム [Platanus \(Kajitani et al., Genome Res., 2014\)](#) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウント
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納

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

名前	サイズ
out_32merFreq.tsv	12 KB
out_contig.fa	2,380 KB
out_contigBubble.fa	1 KB
out_gapClosed.fa	2,332 KB
out_lib1_insFreq.tsv	234 KB
out_scaffold.fa	2,334 KB
out_scaffoldBubble.fa	0 KB
out_scaffoldComponent.tsv	5 KB

# 入力ファイルを変更

適当なテキストエディタ(ここではEmEditor)に例題をコピーし、①必要最小限の変更を施したところ

```
無題 * - EmEditor
ファイル(F) 編集(E) 検索(S) 表示(V) ツール(T) ウィンドウ(W) ヘルプ(H)
無題 * x
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納↓
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定↓
↓
#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
↓
#本番↓
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントし
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納↓
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納↓
↓
#ファイルに保存↓
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=T, col.names=F)
←
Text 1行, 14桁 日本語 (シフト JIS)
```

	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0



# 変更後のコードをコピー

無題 \* - EmEditor

ファイル(F) 編集(E) 検索(S) 表示(V) ツール(T) ウィンドウ(W) ヘルプ(H)

無題 \* x

```
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"
param_base <- c("A",
#必要なパッケージをロ
library(Biostings)
#入力ファイルの読み込
fasta <- readDNASTri
#本番
hoge <- alphabetFreq
obj <- is.element(co
#out <- colSums(hoge
out <- apply(as.matr
#ファイルに保存
write.table(out, out
```

次の文字列を選択に追加(X) Ctrl+R  
 すべての文字列を選択 Ctrl+Shift+A

元に戻す(U)  
 やり直し(R)

切り取り(T)  
**コピー(C) ①**  
 貼り付け(P)  
 引用付きコピー(Q)  
 削除(L)

すべて選択(A)  
 リンクをコピー  
 リンクを開く

選択範囲の変換(S)  
 高度な操作(N)

選択範囲または現在行をコピーしてクリップボードに保存します。

hoge7.txt - Excel

D7	A	B
1	A	729772

R Console

'q()' と入力すれば R を終了します。

```
> getwd()
[1] "C:/Users
> list.files(
[1] "hoge7.tx
[2] "out_32me
[3] "out_cont
[4] "out_cont
[5] "out_gapC
[6] "out_lib1
[7] "out_scaffold.Ia"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
> |
```

コピー Ctrl+C  
**ペースト ②** Ctrl+V  
 コマンドのペースト  
 コピー&ペースト Ctrl+X  
 ウィンドウの消去 Ctrl+L  
 全て選択  
 バッファに出力 Ctrl+W  
 ウィンドウを常にトップに置く

sResult"

# ありがちなミス1

①これはエラーメッセージですw。②エラーの理由は、出力予定ファイル(hoge7.txt)を開くことができない、というもの。Permission denied(権限が与えられていない)は、「アク禁」みたいなものです。Tips:「ワードパッド」や「メモ帳」で開く分にはエラーは出ないようです。

The screenshot shows an R script editor window with a context menu open over a line of code. The R console below shows the execution of the script, resulting in an error message: "file('hoge7.txt') を開くことができません: Permission denied".

```
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"
param_base <- c("A",

#必要なパッケージをロ
library(Biostings)

#入力ファイルの読み込
fasta <- readDNASTri

#本番
hoge <- alphabetFreq
obj <- is.element(co
#out <- colSums(hoge
out <- apply(as.matr

#ファイルに保存
write.table(out, out

<

```

Context menu options:

- 次の文字列を選択に追加(X) Ctrl+R
- すべての文字列を選択 Ctrl+Shift+A
- 元に戻す(U)
- やり直し(R)
- 切り取り(T)
- コピー(C)
- 貼り付け(P)
- 引用付きコピー(Q)
- 削除(L)
- すべて選択(A)
- リンクをコピー
- リンクを開く
- 選択範囲の変換(S)
- 高度な操作(N)

R Console output:

```
>
> #本番
> hoge <- alphabetFrequency(fasta) #A,C,G,T,...の
> obj <- is.element(colnames(hoge), param_base) #条件を注
> #out <- colSums(hoge[, obj]) #列ごとの総和をou
> out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F
file(file, ifelse(append, "a", "w")) でエラー: ①
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:
file(file, ifelse(append, "a", "w")) で:
ファイル 'hoge7.txt' を開くことができません: Permission denied
> |

```

# ありがちなミス1

①エラーの原因は、エクセルで hoge7.txtを開いているから。閉じて再実行すればエラーは出なくなる。

The screenshot shows an R script in EmEditor with a context menu open over the `write.table` function. The R Console displays the following error message:

```
> |  
> #本番  
> hoge <- alphabetFrequency(fasta) #入力ファイル名を指定してin_fに格納  
> obj <- is.element(colnames(hoge), param_base) #必要なパッケージをロードし、library(Biostrings)に格納  
> #out <- colSums(hoge[, obj])  
> out <- apply(as.matrix(hoge[, obj]), MARGIN=2, FUN=function(x) sum(x))  
> #ファイルに保存  
> write.table(out, out_f, sep="\t", append=F, quote=F, file(file, ifelse(append, "a", "w"))) でエラー:  
  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:  
file(file, ifelse(append, "a", "w")) で:  
  ファイル 'hoge7.txt' を開くことができません: Permission denied  
> |
```

The Excel spreadsheet shows the following data:

	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0

# 再実行

エクセルを閉じて再実行した結果。エラーは出ていないことがわかる。①outオブジェクトの中身を見ると、②確かにNがある!

The screenshot shows the EmEditor window with R code. A context menu is open over the code, with the 'Copy (C)' option highlighted. A red arrow labeled '1' points to the 'Copy (C)' option. The code in the background includes:

```
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"
param_base <- c("A",
#必要なパッケージをロ
library(Biostrings)
#入力ファイルの読み込
fasta <- readDNASTri
#本番
hoge <- alphabetFreq
obj <- is.element(co
#out <- colSums(hoge
out <- apply(as.matr
#ファイルに保存
write.table(out, out
```

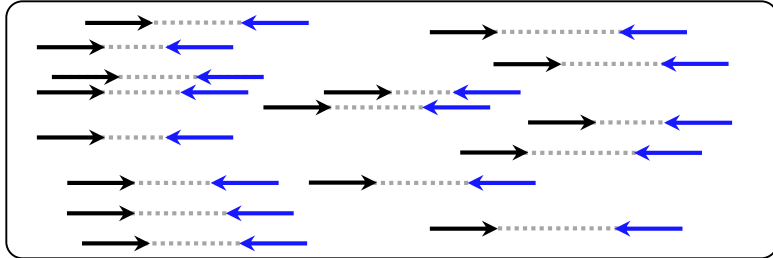
The screenshot shows the R Console window with the following output:

```
>
> #本番
> hoge <- alphabetFrequency(fasta) #A,C,G,T,..の
> obj <- is.element(colnames(hoge), param_base) #条件を満
> #out <- colSums(hoge[, obj]) #列ごとの総和をou
> out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F)
> out
      A      C      G      T      N
729635 458119 440510 727306 491
> sum(out)
[1] 2356061
> |
```

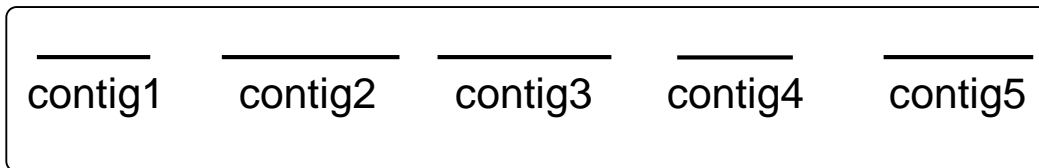
A red arrow labeled '2' points to the value '491' in the 'N' column of the output table. A small cartoon mouse is visible in the bottom right corner of the R Console window.

# 納得できる結果

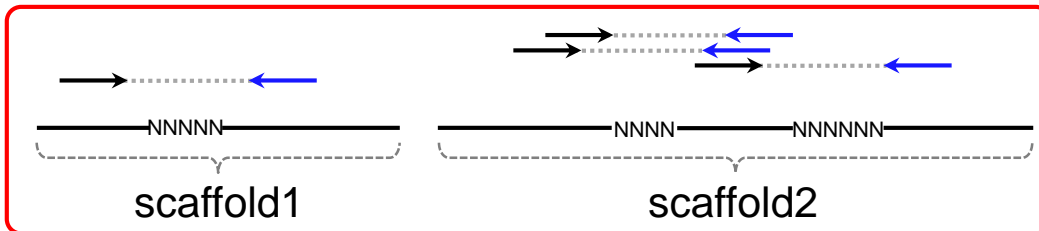
入力: paired-end FASTQファイル



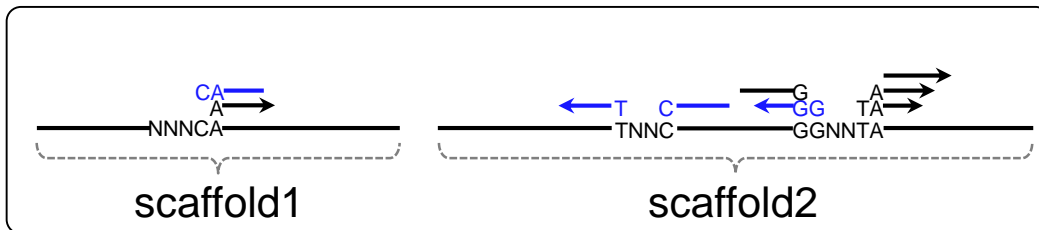
Step1: Assembly



Step2: Scaffold



Step3: Gap close



入力ファイル(out\_scaffold.fa)のイメージは①のような感じなので、②Nが491個あったという結果は合理的

```
AlphabetFrequency(fasta) #A,C,G,T,..の  
.element(colnames(hoge), param_base) #条件を注  
colSums(hoge[, obj]) #列ごとの総和をou  
ply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総  
le(out, out_f, sep="\t", append=F, quote=F
```

C	G	T	N
19	440510	727306	491

The screenshot shows a terminal window with a script output. A table displays the results of a script, with the value '491' under the 'N' column highlighted by a red box and a red arrow labeled '2'. A red circle with the number '1' is also present. A small cartoon mouse icon is visible in the bottom right corner of the terminal window.

# ありがちなミス2

①最終行の部分で、改行をキチンと含めないとハマる

```
無題 * - EmEditor
ファイル(F) 編集(E) 検索(S) 表示(V) ツール(T) ウィンドウ(W) ヘルプ(H)
無題 * x
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定
#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
#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
#ファイルに保存
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=T, col.names=F)#tmpの中身を指定したファイル名で保存↓
<
Text 1行, 1桁 日本語 (シフト JIS)
```



# ありがちなミス2

①最終行の部分で、改行をキチンと含めないと、最後のwrite.table関数部分が実行されない。つまりファイルが作成されません。

```
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,
obj <- is.element(colnames(hoge), param_base)#条
#out <- colSums(hoge[, obj]) #列ごとの
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ご

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

```
>
> #必要なパッケージをロード
> library(Biostrings) #パッケージ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")$
>
> #本番
> hoge <- alphabetFrequency(fasta) #A,C,G,T$
> obj <- is.element(colnames(hoge), param_base)#$
> #out <- colSums(hoge[, obj]) #列ごと$
> out <- apply(as.matrix(hoge[, obj]), 2, sum)#列$
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quo$
```

# 実際の利用時は...

hogeフォルダ直下にある、①rcode1.txtのような、無駄なコメントを除いてスリムにした一連のスクリプトを作成しておき、一気にコピー

```
rcode1.txt
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定↓
library(Biostrings) #パッケージの読み込み↓
↓
#####↓
### Step 1↓
#####↓
in_f <- "out_contig.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "result_step1.txt" #出力ファイル名を指定してout_fに格納↓
↓
fasta <- readDNASTringSet(in_f, format="fasta") #in_fで指定したファイルの読み込み
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納↓
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=T, col.names=F) #t
↓
#####↓
### Step 2↓
#####↓
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "result_step2.txt" #出力ファイル名を指定してout_fに格納↓
↓
fasta <- readDNASTringSet(in_f, format="fasta") #in_fで指定したファイルの読み込み
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納↓
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=T, col.names=F) #t
↓
#####↓
### Step 3↓
#####↓
```



# 一気に結果を得る

hogeフォルダ直下にある、①rcode1.txtのような、無駄なコメントを除いてスリムにした一連のスクリプトを作成しておき、一気にコピペ。  
②コピペ後に自分が指定した出力ファイルができていることを確認

```
rcode1.txt
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定↓
library(Biostrings) #パッケージの読み込み↓

#####↓
### Step 1↓
#####↓
in_f <- "out_contig.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "result_step1.txt" #出力ファイル名を指定してout_fに格納↓

fasta <- readDNASTringSet(in_f, format="fasta") #in
hoge <- alphabetFrequency(fasta) #A,C,G,T,...
obj <- is.element(colnames(hoge), param_base) #条件
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごと
write.table(out, out_f, sep="\t", append=F, quote=

#####↓
### Step 2↓
#####↓
in_f <- "out_scaffold.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "result_step2.txt" #出力ファイル名を指定してout_fに格納↓

fasta <- readDNASTringSet(in_f, format="fasta") #in
hoge <- alphabetFrequency(fasta) #A,C,G,T,...
obj <- is.element(colnames(hoge), param_base) #条件
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごと
write.table(out, out_f, sep="\t", append=F, quote=

#####↓
### Step 3↓
#####↓
```

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files()
[1] "hoge7.txt"
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble.fa"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq.tsv"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubble.fa"
[9] "out_scaffoldComponent.tsv"
[10] "result_step1.txt"
[11] "result_step2.txt"
[12] "result_step3.txt"
```

# 結果のまとめ

```

rcode1.txt x
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定↓
library(Biostrings) #パッケージの読み込み↓
#####↓
### Step 1↓
#####↓
in_f <- "out_contig.fa" #入力ファイル名を指定してin_fに格納↓
out_f <- "result_step1.txt" #出力ファイル名を指定してout_fに格納↓
↓
fasta <- readDNASTringSet(in_f, format="fasta")#in
hoge <- alphabetFrequency(fasta) #A,C,G,T,...
obj <- is.element(colnames(hoge), param_base)#条件
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごと
write.table(out, out_f, sep="¥t", append=F, quote=
↓
#####↓
### Step 2↓
#####↓
in_f <- "out_scaffold.fa" #入力ファイ
out_f <- "result_step2.txt" #出力ファイ
↓
fasta <- readDNASTringSet(in_f, format="fasta")#in
hoge <- alphabetFrequency(fasta) #A,C,G,T,...
obj <- is.element(colnames(hoge), param_base)#条件
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごと
write.table(out, out_f, sep="¥t", append=F, quote=
↓
#####↓
### Step 3↓
#####↓

```

```

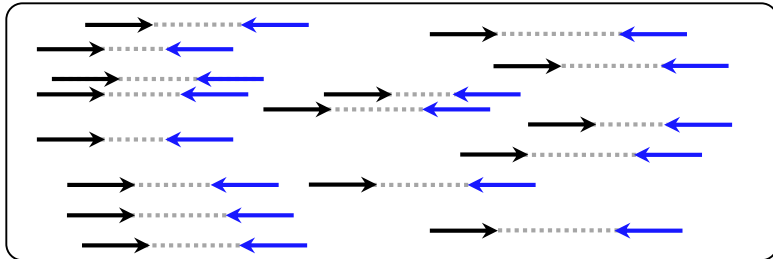
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files()
[1] "hoge7.txt"
[2] "out_32merFrq.tsv"
[3] "out_contig.fa"
[4] "out_contigBubble"
[5] "out_gapClosed.fa"
[6] "out_lib1_insFreq"
[7] "out_scaffold.fa"
[8] "out_scaffoldBubb"
[9] "out_scaffoldComponent.tsv"
[10] "result_step1.txt"
[11] "result_step2.txt"
[12] "result_step3.txt"
> |

```

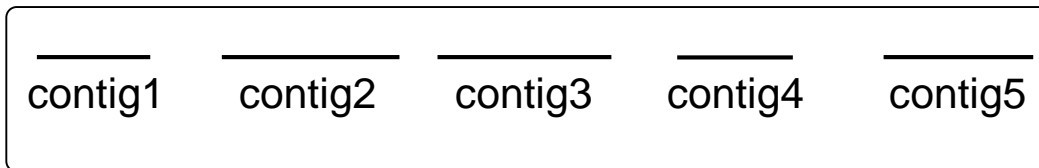
base	Step1	Step2	Step3
A	739836	729635	729772
C	469377	458119	458211
G	446806	440510	440585
T	742714	727306	727451
N	0	491	0

# 結果の解釈

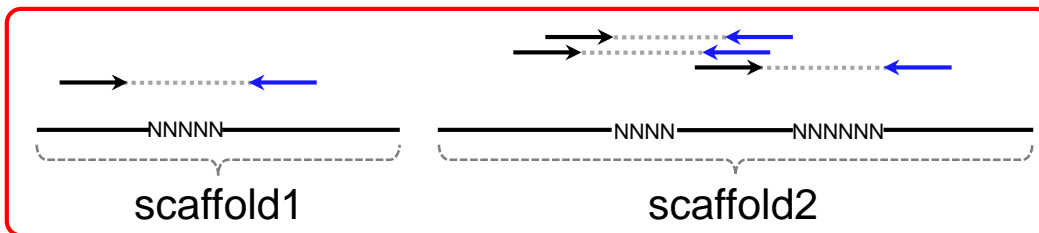
入力: paired-end FASTQファイル



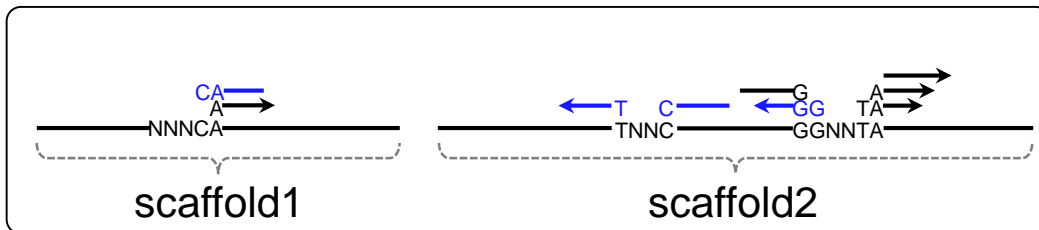
Step1: Assembly



Step2: Scaffold



Step3: Gap close



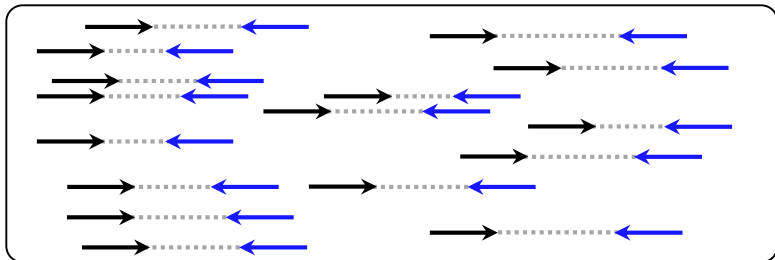
①Step1実行後はNが0。②Step2実行後にNが491個生成されたということは、いくつかのcontigsがまとめられてscaffoldsになったのだろう。③Step3でNが0個になったのはおそらくたまたまうまくいっただけ。491個よりも減ったということが重要で、gap closeがうまく機能したと判断できる。



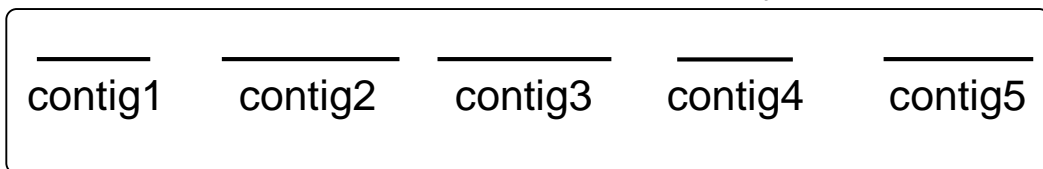
base	Step1	Step2	Step3
A	739836	729635	729772
C	469377	458119	458211
G	446806	440510	440585
T	742714	727306	727451
N	0	491	0

# コード内部の理解は重要

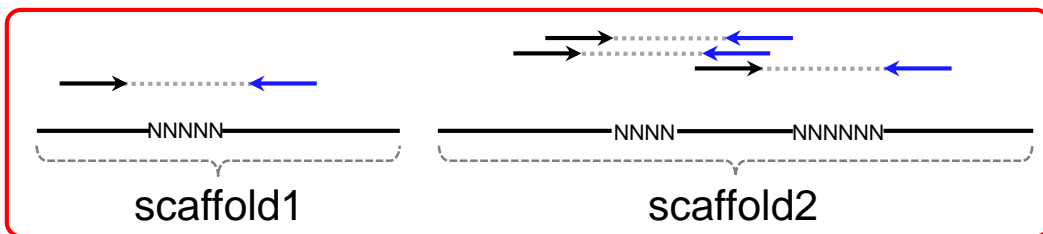
入力: paired-end FASTQファイル



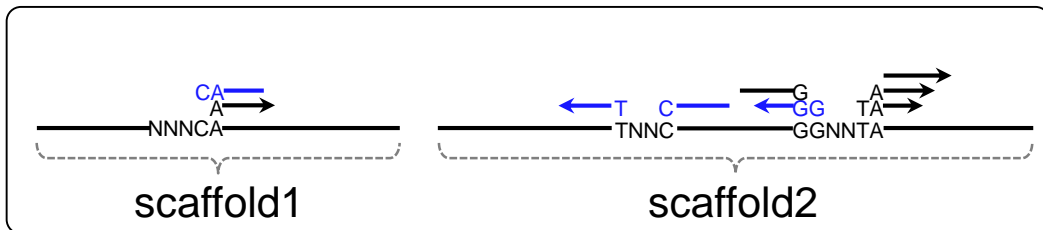
Step1: Assembly



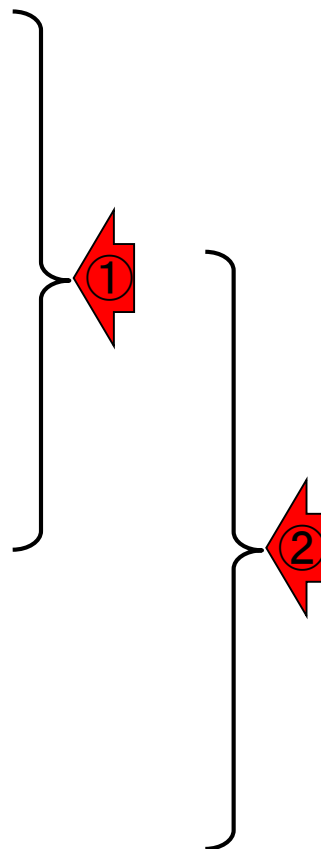
Step2: Scaffold



Step3: Gap close



Rを使うことで、アセンブリプログラムの内部挙動の把握や理解ができる。他の例は、配列数 (contig数やscaffold数と書くと説明しづらいので配列数に統一)。配列数は、①Step1 → Step2で減り、②Step2 → Step3では不変だろうと予想。



# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# Rコードの解説

配列数の把握の仕方、の前にTips。①list.files() 実行時にpatternオプションをつけて任意の文字列を含むもののみ表示させることが可能。ここでは"out\_"という文字列を含むもの(ファイル)のみ表示させている。②入力ファイルの存在確認

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリー (et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)を処理する。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

```

```

#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, fromFile=TRUE)

```

```

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), param_base)
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), MARGIN=2, FUN=function(x) sum(x[obj]))

```

```

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

```

```

R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files()
 [1] "hoge7.txt"           "out_32merFrq.tsv"
 [3] "out_contig.fa"      "out_contigBubble.fa"
 [5] "out_gapClosed.fa"  "out_lib1_insFreq.tsv"
 [7] "out_scaffold.fa"   "out_scaffoldBubble.fa"
 [9] "out_scaffoldComponent.tsv" "result_step1.txt"
[11] "result_step2.txt"   "result_step3.txt"
> list.files(pattern="out_")
 [1] "out_32merFrq.tsv"           "out_contig.fa"
 [3] "out_contigBubble.fa"      "out_gapClosed.fa"
 [5] "out_lib1_insFreq.tsv"     "out_scaffold.fa"
 [7] "out_scaffoldBubble.fa"    "out_scaffoldComponent.tsv"
> |

```

# Rコードの解説

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定
```

```
#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, fo
```

```
#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]
```

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

```
R Console
> list.files(pattern="out_")
[1] "out_32merFrq.tsv" "out_contig.fa"
[3] "out_contigBubble.fa" "out_gapClosed.fa"
[5] "out_lib1_insFreq.tsv" "out_scaffold.fa"
[7] "out_scaffoldBubble.fa" "out_scaffoldComponent.tsv"
> in_f <- "out_gapClosed.fa" #入力ファイル名を$
> out_f <- "hoge7.txt" #出力ファイル名を$
> param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基$
>
> #必要なパッケージをロード
> library(Biostrings) #パッケージの読み$
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")#in_fで指$
> |
```

# Rコードの解説

①BiostringsというRパッケージをlibrary関数で読み込んで、Biostringsパッケージが提供する関数群を利用可能な状態にする。②(Biostringsが提供する)readDNAStringSet関数を用いて、③FASTA形式の④入力ファイルを読み込んだ結果を、⑤fastaというオブジェクト(もの、という理解でよい)に格納。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ (et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイルです。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

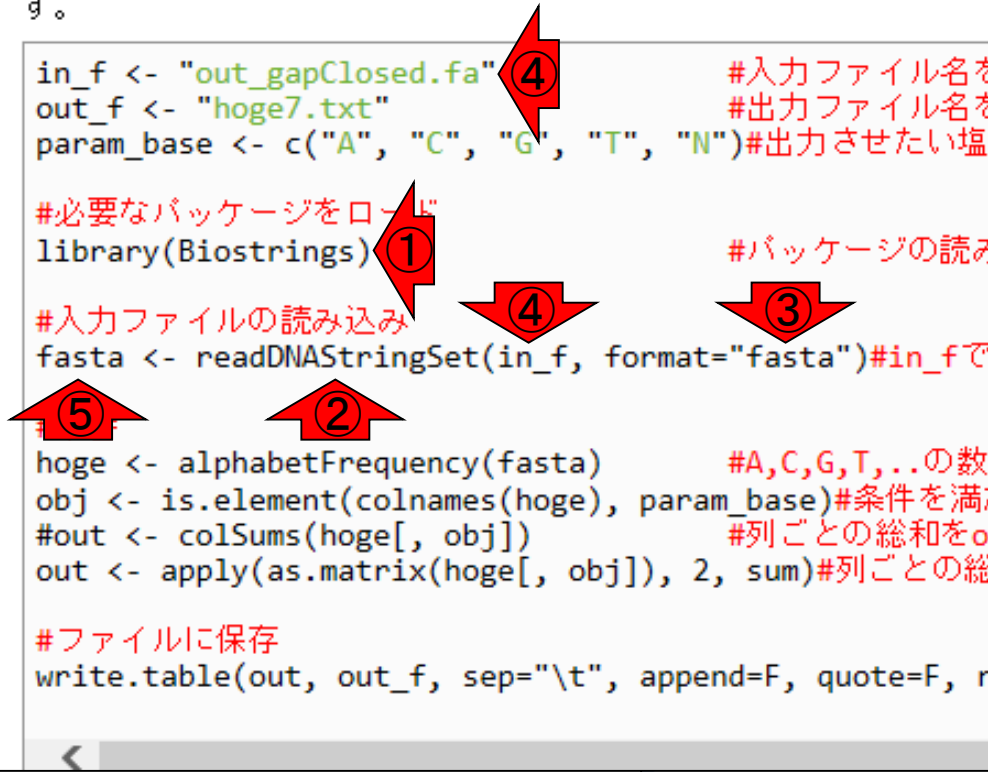
#必要なパッケージをロード
library(Biostrings) #パッケージの読み込み

#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

hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobj
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納

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

```



```

contig.fa"
_gapClosed.fa"
scaffold.fa"
scaffoldComponent.tsv"
#入力ファイル名を$
#出力ファイル名を$
"N") #出力させたい塩基$

#パッケージの読み$

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

```



# Tips: 配列数

①fastaオブジェクトの中身を表示。(ここでの目的の)配列数は②117個。スカラー値として配列数情報のみ取り出したい場合は、③ベクトルの要素数を調べるlength関数を利用する

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa"
out_f <- "hoge7.txt"
param_base <- c("A", "C", "G", "T")

#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, fo

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

```

#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
> fasta
A DNASTringSet instance of length 117
      width seq
[1] 10520 ATCGATATTATT...ATGGATTGCTG scaffold1_cov55
[2] 136075 TTTAAGGACCCT...ATCCGAATTAA scaffold2_cov60
[3] 64531 CACCGTATCAGG...ACTGTATCTAG scaffold3_cov43
[4] 35091 CAAAATGCACTG...TTTAGATTGAA scaffold4_cov52
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
...
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTAACTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGGTCTTTC...TTCATCTTTCG scaffold117_cov182

> length(fasta)
[1] 117
> |

```



# 答え合わせ

①DDBJ Pipeline実行結果の数値(117個)と同じことがわかります。②最長の配列(Maximum contig size; 257,728 bp)と最短の配列(Minimum contig size; 101 bp)もR上で把握できます。

<b>ID</b>													
21211													
<b>Tool (Version)</b>													
Platanus (1.2.2)													
<b>RunAccession or Filename</b>	<b>Download</b>	<b>Read length</b>	<b>Alias</b>										
QC.1.trimmed.fastq.gz	<a href="#">QC.1.trimmed.fastq.gz</a>	N.A. bp	L.hokkaidonensis_MiSeq_denovo										
<b>Download modified queries</b>													
<ul style="list-style-type: none"> <li><a href="#">QC.1.trimmed.fastq.gz (Original size 189.4 MB)</a></li> <li><a href="#">QC.2.trimmed.fastq.gz (Original size 189.6 MB)</a></li> </ul>													
<b>Download wgs file</b>													
<ul style="list-style-type: none"> <li><a href="#">out_WGS.fasta.gz (Original size 2.3 MB)</a></li> </ul>													
<b>Assembly statistics</b>													
<table border="0"> <tr> <td>Contig #</td> <td>: 117</td> </tr> <tr> <td>Total contig size</td> <td>: 2,356,019</td> </tr> <tr> <td>Maximum contig size</td> <td>: 257,728</td> </tr> <tr> <td>Minimum contig size</td> <td>: 101</td> </tr> <tr> <td>N50 contig size</td> <td>: 92,304</td> </tr> </table>				Contig #	: 117	Total contig size	: 2,356,019	Maximum contig size	: 257,728	Minimum contig size	: 101	N50 contig size	: 92,304
Contig #	: 117												
Total contig size	: 2,356,019												
Maximum contig size	: 257,728												
Minimum contig size	: 101												
N50 contig size	: 92,304												
<b>Time</b>													
<b>Wait time</b>	<b>Start time</b>	<b>End time</b>											
0: 0:9	2016-01-20 18:33:36	2016-01-21 10:10:06											
<b>Command</b>	<b>Start time</b>	<b>End time</b>	<b>Log1</b>	<b>Log2</b>	<b>Result</b>	<b>MD5</b>							
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>							
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>							
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>							



配列長の情報は、(DNASTringSetという形式で保持されている)fastaオブジェクト中の、①width列の位置に相当する。

# Tips: 配列長

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

```

```

#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, fo

```

```

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj])

```

```

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

```

```

R Console
> fasta
A DNASTringSet instance of length 117
      width seq          names
[1] 10520 ATCGATATTATT...ATGGATTGCTG scaffold1_cov55
[2] 136075 TTTAAGGACCCT...ATCCGAATTAA scaffold2_cov60
[3] 64531 CACCGTATCAGG...ACTGTATCTAG scaffold3_cov43
[4] 35091 CAAAATGCACTG...TTTAGATTGAA scaffold4_cov52
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
...
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTAACTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGGTCTTTC...TTCATCTTTCG scaffold117_cov182
> |

```

# Tips: 配列長

配列長情報は①width(fasta)とやることで、数値ベクトルとして取り出すことができる。この程度(117個)の配列数なら、パッと見て②最長と③最短のものを確認できるが...

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa"
out_f <- "hoge7.txt"
param_base <- c("A", "C", "G", "T")

#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, fromFile=TRUE)

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), param_base)
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), MARGIN=2, FUN=function(x) sum(x))

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

#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
> width(fasta)
 [1] 10520 136075 64531 35091 105 125 125
 [8] 467 125 257728 20654 68508 84851 512
[15] 94505 65563 54737 83975 117 125 125
[22] 73481 92304 6859 18430 154194 6893 125
[29] 352 96074 103 279 41161 63901 125
[36] 124 125 138 124 125 101 125
[43] 8311 6439 37583 10647 204783 108 64495
[50] 613 125 126 125 35878 125 251
[57] 156 539 96931 200 103 10030 14088
[64] 125 125 195 96261 31108 16776 112
[71] 19996 71496 58979 125 1918 125 125
[78] 168 213 125 125 125 714 115
[85] 125 794 195 1442 124 457 247
[92] 155 123 125 117 798 125 255
[99] 125 1439 117 1220 113 125 167
[106] 146 209 289 652 180 165 177
[113] 113 747 159 117 263
> |
```



# Tips: 配列長

ベクトル演算の基本関数を駆使して全貌を把握する。上矢印キーを1回押して、以前打ち込んだコマンドを出すなど、上下左右の矢印キーを駆使して効率的に打ち込むべし

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

[DDBJ Pipeline \(Nagasaki et al., DNA Res., 2013\)](#)上で de novoゲノムアセンブリプログラム [Platanus \(Kajitani et al., Genome Res., 2014\)](#) を実行して得られた multi-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定
```

```
#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, fo
```

```
#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]
```

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

```
R Console
[92] 155 123 125 117 798 125 255
[99] 125 1439 117 1220 113 125 167
[106] 146 209 289 652 180 165 177
[113] 113 747 159 117 263
> min(width(fasta))
[1] 101
> max(width(fasta))
[1] 257728
> range(width(fasta))
[1] 101 257728
> summary(width(fasta))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   101    125    213   20140   14090   257700
> |
```

# Tips: description部分

①description行部分は、②namesという関数を用いることで、(文字列)ベクトルとして取り扱うことができる。ここでは③1:4という指定を行って、最初の4個分のみ表示させている

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させない塩基を指定

```

```

#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, fo

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]

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

```

```

R Console
> fasta
A DNASTringSet instance of length 117
      width seq
[1] 10520 ATCGATATTATT...ATGGATTGCTG scaffold1_cov55
[2] 136075 TTTAAGGACCCT...ATCCGAATTAA scaffold2_cov60
[3] 64531 CACCGTATCAGG...ACTGTATCTAG scaffold3_cov43
[4] 35091 CAAAATGCACTG...TTTAGATTGAA scaffold4_cov52
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
...
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTATTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGATTC...TTCATCTTTCG scaffold117_cov182
> names(fasta)[1:4]
[1] "scaffold1_cov55" "scaffold2_cov60" "scaffold3_cov43"
[4] "scaffold4_cov52"
> |

```

# Tips: 塩基配列部分

但し、①このノリは塩基配列部分には通用しないw。②seqという関数は別の意味を持つこと、fastaオブジェクトの主要な中身がこの塩基配列情報であるため、と理解すればよい

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013) 上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させない塩基を指定

```

```

#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, fo

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]

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

```

```

R Console
> fasta
A DNASTring instance of length 117
      width seq
[1] 10520 ATCGATATTATT...ATGGATTGCTG scaffold1_cov55
[2] 136075 TTTAAGGACCCT...ATCCGAATTAA scaffold2_cov60
[3] 64531 CACCGTATCAGG...ACTGTATCTAG scaffold3_cov43
[4] 35091 CAAAATGCACTG...TTTAGATTGAA scaffold4_cov52
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
...
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTAACTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGGTCTTTC...TTCATCTTTCG scaffold117_cov182

> seq(fasta)[115:117]
[1] 115 116 117
> |

```

# Tips: 塩基配列部分

どうしても文字列ベクトルなどで取り出したい場合は①as.character関数を使うが、②DNASTringSet形式のfastaオブジェクトをそのまま用いて各種塩基配列解析を行うのが通常やり方。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013) 上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

```

```

#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, fo

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]

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

```

```

R Console
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
... ..
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACCTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTAACTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGGTCTTTC...TTCATCTTTCG scaffold117_cov182
> seq(fasta) [115:117]
[1] 115 116 117
> as.character(fasta) [115:117]
"ATGGGGTCTTTCCGTCCTGTCGCGGGTAACCTGCATCTTCACAGGTACTTCAATTTCA$
> |

```





# Contents1

## ■ イン트로ダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# alphabetFrequency

塩基ごとの出現頻度解析の中核となっている関数は①alphabetFrequency。②実行結果であるhogeの中身は数値行列。この段階で塩基配列解析から数値解析に切り替わる。塩基の種類には多型がある。例えば③M (A or C)、④K (G or T)など。門田は、シロイヌナズナのゲノム配列で、ACGTN以外のものを見た記憶あり。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out

```
in_f <- "out_gapClosed.fa"
out_f <- "hoge7.txt"
param_base <- c("A", "C", "G", "T")
```

```
#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, fo
```

```
#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]
```

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

#入力ファイル名を指定  
#出力ファイル名を指定

```
R Console
> fasta
A DNASTringSet instance of length 117
      width seq
[1] 10520 ATCGATATTATT...ATGGATTGCTG scaffold1_cov55
[2] 136075 TTTAAGGACCCT...ATCCGAATTAA scaffold2_cov60
[3] 64531 CACCGTATCAGG...ACTGTATCTAG scaffold3_cov43
[4] 35091 CAAAATGCACTG...TTTAGATTGAA scaffold4_cov52
[5] 105 CAAAACGAGTAA...TGATTGTGCTA scaffold5_cov98
...
[113] 113 CATTAGTGGATT...TTTTACGATGA scaffold113_cov168
[114] 747 CGGGAGTATGCT...CAAATTCACGC scaffold114_cov106
[115] 159 ACAAACTTGTTT...ATTACTAAATT scaffold115_cov182
[116] 117 CTTTAACTCATC...TAAGAAATTGT scaffold116_cov184
[117] 263 ATGGGGTCTTTC...TTCATCTTTCG scaffold117_cov182

> hoge <- alphabetFrequency(fasta) #A, C, G, T, ...の数を$
> hoge
      A      C      G      T M R W S Y K V H D B N - + .
[1,] 3149 1852 2101 3418 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 41810 28606 23051 42608 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

# dim

①dim関数で行数と列数を把握。alphabetFrequencyは、配列ごとに結果を返しているのので、117行からなると解釈。②18列であることから、塩基の種類数は18個と解釈。③行列の一部要素の取り出し例

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa"
out_f <- "hoge7.txt"
param_base <- c("A", "C", "G", "T")

#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, fo

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge),
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, ob

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

R Console

```
#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
```

[109,]	137	203	150	162	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[110,]	48	30	32	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[111,]	44	26	31	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[112,]	48	44	25	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[113,]	32	28	16	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[114,]	199	146	129	273	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[115,]	47	35	22	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[116,]	29	25	22	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[117,]	49	79	60	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
> dim(hoge)
[1] 117 18
> hoge[112:116, 2:6]
  C  G  T  M  R
[1,] 44 25 60 0 0
[2,] 28 16 37 0 0
[3,] 146 129 273 0 0
[4,] 35 22 55 0 0
[5,] 25 22 41 0 0
> |
```

# is.element

①は、出現頻度情報を得たい塩基の種類を指定するところ。ほとんどの場合ACGTNのみで事足りるので、このようにしている。is.element関数は条件判定(集合演算)を行っている。②行列hogeの列名(column names)からなるベクトルの中から、③param\_baseで指定された要素が存在する場所をTRUE、そうでないところをFALSEと評価するのが④is.element関数

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ (et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイルです。

```

in_f <- "out_gapClosed.fa" #入力ファイル名
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]), MARGIN=2, FUN=colSums)

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

```

R Console
> param_base
[1] "A" "C" "G" "T" "N"
> colnames(hoge)
[1] "A" "C" "G" "T" "M" "R" "W" "S" "Y" "K" "V" "H" "D" "B"
[15] "N" "-" "+" "."
> obj <- is.element(colnames(hoge), param_base) #条件を満たす
> obj
[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
[10] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
>
    
```

# 条件を満たす列のみ

行列のsubsettingは、[行, 列]で指定する。[, 列]で列のみの指定、[行, ]で行のみの指定となる。  
 ①hoge[, obj]は、objベクトルのTRUEとなっている列の位置のみ取り出すことに相当する。  
 ②hoge[1:2, ]でhoge行列の最初の2行分のみ表示。  
 ③hoge[1:2, obj]の合わせ技で、さらにparam\_baseで指定した塩基のみを出力できるようになる。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセン et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル( )す。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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=

#本番
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), param
#out <- colSums(hoge[, obj])
out <- apply(as.matrix(hoge[, obj]),

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

```

R Console
> dim(hoge)
[1] 117 18
> dim(hoge[, obj])
[1] 117 5
> hoge[1:2, ]
      A      C      G      T M R W S Y K V H D B N - + .
[1,] 3149 1852 2101 3418 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 41810 28606 23051 42608 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> hoge[1:2, obj]
      A      C      G      T N
[1,] 3149 1852 2101 3418 0
[2,] 41810 28606 23051 42608 0
> |
    
```



# colSums

① alphabetFrequency実行結果は、配列ごとに各塩基の出現頻度を計算している。そのため、hogeは117行分の要素からなる。② colSumsは、行列データを入力として、列ごとに総和(column sum)を計算する関数。colSumsを適用することで、配列ごとではなくファイル全体の出現頻度を得ることができる(今得たい情報はこれ)。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセン et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル( )です。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta)    #A,C,G,T,..の数を各配列ごと
obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうか
#out <- colSums(hoge[, obj])         #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納

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

```



```

R Console
> dim(hoge[, obj])
[1] 117 5
> colSums(hoge[, obj])
      A      C      G      T      N
729772 458211 440585 727451      0
> colSums(hoge[1:2, obj])
      A      C      G      T      N
44959 30458 25152 46026      0
> colSums(hoge[1, obj])
colSums(hoge[1, obj]) でエラー:
  'x' は少なくとも二次元の配列でなければなりません
> |

```

# colSums

①は最初の2行分のみで列ごとの総和を計算する場合。②ではエラーとなっている。colSumsの入力として与えているhoge[1, obj]は、最初の1行分のみからなる。つまり、入力が2次元の行列データではなく1次元のベクトルになってしまっているため。③行頭に#をつけており、実際にはこのコードは動作していない。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセン et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル( )す。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta)     #A,C,G,T,..の数を各配列ごとに
col <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを
#out <- colSums(hoge[, obj])         #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納

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

```

```

R Console
> dim(hoge[, obj])
[1] 117 5
> colSums(hoge[, obj])
      A      C      G      T      N
729772 458211 440585 727451      0
> colSums(hoge[1:2, obj])
      A      C      G      T      N
44959 30458 25152 46026      0
> colSums(hoge[1, obj])
colSums(hoge[1, obj]) でエラー:
  'x' は少なくとも二次元の配列でなければなりません
> |

```

このコードで実際に動かしているのは、①apply関数を用いるほう。結果はcolSumsと同じ。おそらく行列演算で行ごとや列ごとに何かを行うときには、一般にapply関数を用いるので、一応示した。

# applyが一般的かも

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta)     #A,C,G,T,..の数を数える
obj <- is.element(colnames(hoge), param_base) #条件を満たす列を抽出
#out <- colSums(hoge[, obj])          #列ごとの総和を計算
out <- apply(as.matrix(hoge[, obj]), 2, sum) #①行ごとの総和を計算

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

R Console

```

> colSums(hoge[, obj])
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(hoge[, obj], 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(as.matrix(hoge[, obj]), 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
    
```





# applyの説明

applyは、①入力データに対して、②列ごと(行ごとの場合はここを1にする)に、③総和を計算するsum関数を適用する、みたいな指定を行う。colSumsだと、sumを計算することしかできないが、applyの場合は③のところの関数名をmean, median, maxなどいろいろ自在に変更できる。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセン et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル( )す。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta)     #A,C,G,T,..の数をカウント
obj <- is.element(colnames(hoge), param_base)#条件を満たす列を抽出
#out <- colSums(hoge[, obj])         #抽出した列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和を計算

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

```

R Console
> colSums(hoge[, obj])
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(hoge[, obj], 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(as.matrix(hoge[, obj]), 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
    
```



# as.matrix

実はこの入力ファイルの場合は、①as.matrixという関数を②つけなくても、③つけたときと同じ結果が得られる。つけている理由は、`apply(as.matrix(...), 1, sum)`などとして行ごとにsum関数を適用したいときに、配列数が複数の場合でも単数の場合でも统一的にエラーなく処理できたという記憶があったから。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ (Nagasaki et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイルです。

```

in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

#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に格納

#本番
hoge <- alphabetFrequency(fasta)     #A,C,G,T,..の数
obj <- is.element(colnames(hoge), param_base) #条件を満たす列を抽出
#out <- colSums(hoge[, obj])          #列ごとの総和を計算
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和を計算

#ファイルに保存 ①
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F) #tn
    
```

```

R Console
> colSums(hoge[, obj])
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(hoge[, obj], 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
> apply(as.matrix(hoge[, obj]), 2, sum)
      A      C      G      T      N
729772 458211 440585 727451      0
    
```



# as.matrix

挙動の違いは、①入力データの行列が1行しかない(配列数が1つの)場合に出てくる。②複数行からなる(配列数が2つ以上の)場合と比べればエラーメッセージの違いがわかります。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を算出
obj <- is.element(colnames(hoge), param_base)#条件を満たす列を抽出
#out <- colSums(hoge[, obj]) #列ごとの総和を算出
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和を算出

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

R Console

```
> colSums(hoge[, obj])
  A      C      G      T      N
729772 458211 440585 727451      0
> apply(hoge[, obj], 2, sum)
  A      C      G      T      N
729772 458211 440585 727451      0
> apply(as.matrix(hoge[, obj]), 2, sum)
  A      C      G      T      N
729772 458211 440585 727451      0

> colSums(hoge[1, obj])
colSums(hoge[1, obj]) でエラー:
'x' は少なくとも二次元の配列でなければなりません
> apply(hoge[1, obj], 2, sum)
apply(hoge[1, obj], 2, sum) でエラー:
dim(X) は正の長さを持たねばなりません
> apply(as.matrix(hoge[1, obj]), 2, sum)
[1] 10520
> |
```



# 思考停止するべからず

①as.matrixをつけてエラーメッセージが出てないからといって、これが正しいわけではないことに気づこう



## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "out_gapClosed.fa"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"           #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定
```

```
#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
```

```
#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数をカウント
obj <- is.element(colnames(hoge), param_base)#条件を満たす列を抽出
#out <- colSums(hoge[, obj])      #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
```

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

```
R Console
> colSums(hoge[, obj])
  A      C      G      T      N
729772 458211 440585 727451      0
> apply(hoge[, obj], 2, sum)
  A      C      G      T      N
729772 458211 440585 727451      0
> apply(as.matrix(hoge[, obj]), 2, sum)
  A      C      G      T      N
729772 458211 440585 727451      0
> colSums(hoge[1, obj])
colSums(hoge[1, obj]) でエラー:
  'x' は少なくとも二次元の配列でなければなりません
> apply(hoge[1, obj], 2, sum)
apply(hoge[1, obj], 2, sum) でエラー:
  dim(X) は正の長さを持たねばなりません
① > apply(as.matrix(hoge[1, obj]), 2, sum)
[1] 10520
> |
```



# 思考停止するべからず

①の実行結果である10520という数値は、単純に②1番目の配列の長さ。今調べたいのは塩基ごとの出現頻度情報なので、③が正解!

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力される塩基を指定

#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")

#本番
hoge <- alphabetFrequency(fasta) #A
obj <- is.element(colnames(hoge), param_base) #列ごとの出現頻度
out <- apply(as.matrix(hoge[, obj]), 2, sum)

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

```

R Console
> apply(as.matrix(hoge[1, obj]), 2, sum)
[1] 10520
> fasta[1]
A DNASTringSet instance of length 1
width seq names
[1] 10520 ATCGATATTA...GGATTGCTG scaffold1_cov55
> hoge[1, ]
      A      C      G      T      M      R      W      S      Y      K
3149 1852 2101 3418      0      0      0      0      0      0
      V      H      D      B      N      -      +      .
      0      0      0      0      0      0      0      0
> hoge[1, obj]
      A      C      G      T      N
3149 1852 2101 3418      0
> sum(hoge[1, obj])
[1] 10520
> |
    
```

# 思考停止するべからず

少なくともこのサンプルコードは、配列数が1つしかない場合にはうまく動かないことが既知。欲しい結果が(この場合は)数値ベクトルになっていない段階でおかしいと思えるようになりましょう。一般論としては、得られる結果をイメージし、特にイメージと異なる場合に疑いの目で結果を眺めよう。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

[DDBJ Pipeline \(Nagasaki et al., DNA Res., 2013\)](#)上で de novoゲノムアセンブリプログラム [et al., Genome Res., 2014](#) を実行して得られた multi-FASTA形式ファイル(out\_gapClosed.fa)を処理する。

```
in_f <- "out_gapClosed.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                 #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta)     #A,C,G,T,..の数を各配列ごとにカウントした結果
obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj])         #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納

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

# Rコードの解説

赤下線のように沢山のオプションを駆使している。① sep="¥t"は区切り文字を指定するオプション。¥tはタブ区切りの意味。②row.names=Tは、行の名前(row names)をTRUEにせよという意味。ここがTになると、③赤枠部分の情報が追加される。FALSEにすると、この列は消える。④ col.names=Fは、col.names=Tにしたときに無意味なヘッダ一行が含まれるのが嫌だったのでこうしているだけ。

## 7. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novo (et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式です。

```

in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N")#出力させたい塩基を指定

#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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果
obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納

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



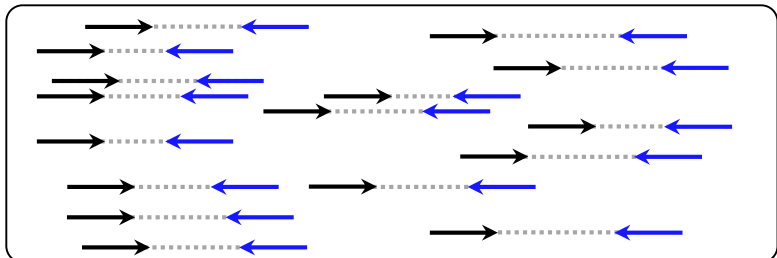
	A	B
1	A	729772
2	C	458211
3	G	440585
4	T	727451
5	N	0

③ points to the 'N' row in the table.

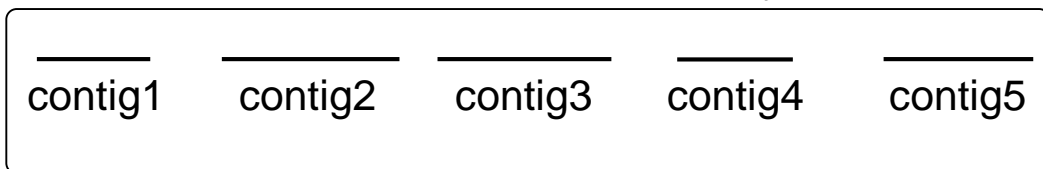
配列数は、①Step1 → Step2で減り、②Step2 → Step3では不変だろうと予想

# 目的をおさらい

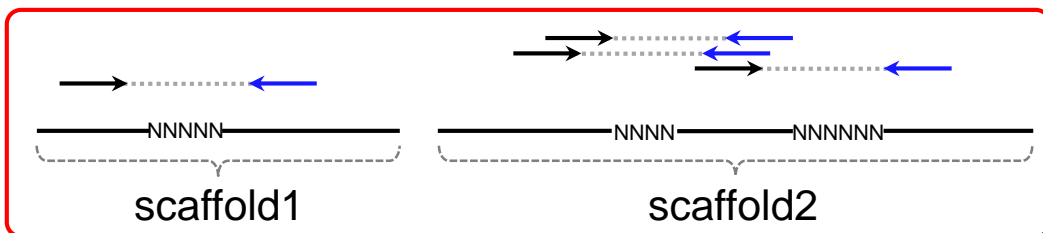
入力: paired-end FASTQファイル



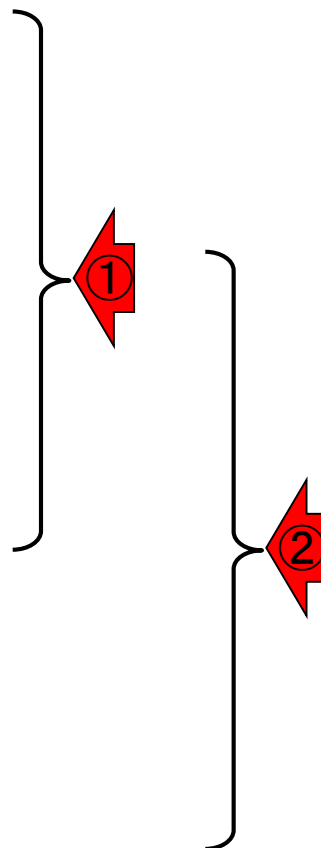
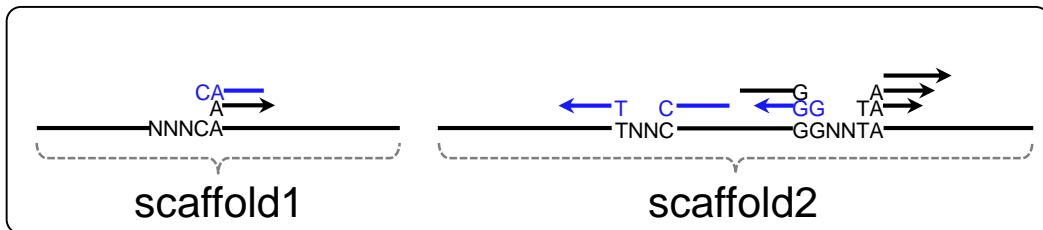
Step1: Assembly



Step2: Scaffold



Step3: Gap close





# 目的をおさらい

配列数は、①Step1 → Step2で減り、② Step2 → Step3では不変だろうと予想。③hogeフォルダ直下のrcode2.txtは、配列数をカウントする必要最小限のコード。349 → 117 → 117で予想通りの結果

```
rcode2.txt ③
library(Biostrings) #パッケージの読み込み↓
↓
### Step 1 ###↓
in_f <- "out_contig.fa" #入力ファイル
fasta <- readDNAStringSet(in_f, format="fasta")#in
length(fasta) #配列数を表
↓
### Step 2 ###↓
in_f <- "out_scaffold.fa" #入力ファイル
fasta <- readDNAStringSet(in_f, format="fasta")#in
length(fasta) #配列数を表
↓
### Step 3 ###↓
in_f <- "out_gapClosed.fa" #入力ファイル
fasta <- readDNAStringSet(in_f, format="fasta")#in
length(fasta) #配列数を表
```

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files(pattern="*.fa")
[1] "out_contig.fa" "out_contigBubble.fa"
[3] "out_gapClosed.fa" "out_scaffold.fa"
[5] "out_scaffoldBubble.fa"
> ### Step 1 ###
> in_f <- "out_contig.fa" #入力フ$
> fasta <- readDNAStringSet(in_f, format="fasta")
> length(fasta) #配列数$
[1] 349
>
> ### Step 2 ###
> in_f <- "out_scaffold.fa" #入力フ$
> fasta <- readDNAStringSet(in_f, format="fasta")
> length(fasta) #配列数$
[1] 117
>
> ### Step 3 ###
> in_f <- "out_gapClosed.fa" #入力フ$
> fasta <- readDNAStringSet(in_f, format="fasta")
> length(fasta) #配列数$
[1] 117
```

# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



## 塩基配列解析基礎2

ID		21211				
Tool (Version)		Platanus (1.2.2)				
RunAccession or Filename	Download	Read length	Alias			
QC.1.trimmed.fastq.gz	<a href="#">QC.1.trimmed.fastq.gz</a>	N.A. bp	L.hokkaidonensis_MiSeq_denovo			
Download modified queries						
<ul style="list-style-type: none"> <li><a href="#">QC.1.trimmed.fastq.gz (Original size 189.4 MB)</a></li> <li><a href="#">QC.2.trimmed.fastq.gz (Original size 189.6 MB)</a></li> </ul>						
Download wgs file						
<ul style="list-style-type: none"> <li><a href="#">out_WGS.fasta.gz (Original size 2.3 MB)</a></li> </ul>						
Assembly statistics						
		<div style="border: 1px solid red; padding: 5px;">           Contig # : 117            Total contig size : 2,356,019            Maximum contig size : 257,728            Minimum contig size : 101            N50 contig size : 92,304         </div>				
Time						
Wait time	Start time	End time				
0: 0:9	2016-01-20 18:33:36	2016-01-21 10:10:06				
Command	Start time	End time	Log1	Log2	Result	MD5
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>



# 塩基配列解析基礎2

- ・ [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 modified 2015/09/12)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [BSgenome](#) | [基本情報を取得](#) (last modified 2015/09/12)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTA形式](#) | [基本情報を取得](#) (last modified 2015/09/12)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTA形式](#) | [description行の記号を整形](#) (last modified 2014/04/26)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [基礎](#) (last modified 2015/07/26)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [応用](#)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [description行の記号を整形](#)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [Illuminaの\\* seq.txt](#)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [Illuminaの\\* qseq.txt](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [||](#)について (last modified 2015/09/12)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [BAM -> BED](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [FASTQ -> FASTA](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [Genbank -> FASTA](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [qseq -> FASTA](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [qseq -> Illumina](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [qseq -> Sanger](#)
- ・ [前処理](#) | [クオリティコントロール](#) | [||](#)について (last modified 2015/09/12)
- ・ [前処理](#) | [クオリティチェック](#) | [QuasR\(Gaidatzis 2015\)](#)
- ・ [前処理](#) | [クオリティチェック](#) | [orac](#) (last modified 2015/09/12)

## 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ー。

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

```

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

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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta))) #長さ情報を降順にソートした結果をsortedに格納
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすかどうかを判定した結果をobjに格納(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に格納
    
```

例題の①入力ファイル名部分を  
②out\_gapClosed.faに変更した  
③rcode3.txtをコピペで実行

# 塩基配列解析基礎2

イントロ | NGS | 読み込み | FASTA形式 | [基本情報を取得](#) ③

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなど「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

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

```

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

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

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

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

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

```

rcode3.txt *
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納

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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta))) #長さ情報を降順にソートした結果をsortedに格納
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすかどうかを判定した結果
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) #A,C,G,T,...の数を配列ごとにカウントし、結果をhogeに格納
CG <- rowSums(hoge[,2:3]) #C,Gの総数を計算してCGに格納(2015年9月)
ACGT <- rowSums(hoge[,1:4]) #A,C,G,Tの総数を計算してACGTに格納(2015年9月)
CG <- apply(as.matrix(hoge[,2:3]), 1, sum)#C,Gの総数を計算してCGに格納(2015年9月)
ACGT <- apply(as.matrix(hoge[,1:4]), 1, sum)#A,C,G,Tの総数を計算してACGTに格納(2015年9月)
GC_content <- sum(CG)/sum(ACGT) #トータルのGC含量の情報を取得
    
```

# 塩基配列解析基礎2

①出力ファイル(hoge1.txt)を開かずに、②write.table関数で書きだしているtmpの中身を表示

```
rcode3.txt * x
in_f <- "out_gapClose_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で読み込み↓
#本番(基本情報取得)↓
Total_len <- sum(width(fasta)) #配列の「トータル長」取得
Number_of_contigs <- length(fasta) #「配列数」を取得
Average_len <- mean(width(fasta)) #配列の「平均長さ」取得
Median_len <- median(width(fasta)) #配列の「中央値」取得
Max_len <- max(width(fasta)) #配列の長さの「最大値」取得
Min_len <- min(width(fasta)) #配列の長さの「最小値」取得
#本番(N50情報取得)↓
sorted <- rev(sort(width(fasta))) #長さ情報を降順に並び替え
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たす最初のindex
N50 <- sorted[obj][1] #objがTRUEとなる最初のindex
#本番(GC含量情報取得)↓
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を取得
#CG <- rowSums(hoge[,2:3]) #C,Gの総数を計算
#ACGT <- rowSums(hoge[,1:4]) #A,C,G,Tの総数を計算
CG <- apply(as.matrix(hoge[,2:3]), 1, sum)#C,Gの総数を計算
ACGT <- apply(as.matrix(hoge[,1:4]), 1, sum)#A,C,G,Tの総数を計算
GC_content <- sum(CG)/sum(ACGT) #トータルのGC含量
```

```
R Console
> #ファイルに保存
> tmp <- NULL
> tmp <- rbind(tmp, c("Total length (bp)", Total_$
> tmp <- rbind(tmp, c("Number of contigs", Number_$
> tmp <- rbind(tmp, c("Average length", Average_l$
> 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, quo$
> tmp
      [,1]           [,2]
[1,] "Total length (bp)" "2356019"
[2,] "Number of contigs" "117"
[3,] "Average length"    "20136.9145299145"
[4,] "Median length"     "213"
[5,] "Max length"        "257728"
[6,] "Min length"        "101"
[7,] "N50"               "92304"
[8,] "GC content"        "0.381489283405609"
> |
```

# 塩基配列解析基礎2

```
rcode3.txt * x
in_f <- "out_gapClosed.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で読み込み↓
#本番(基本情報取得)↓
Total_len <- sum(width(fasta)) | #配列の「トータル長」取得
Number_of_contigs <- length(fasta) | #「配列数」を取得
Average_len <- mean(width(fasta)) | #配列の「平均長さ」取得
Median_len <- median(width(fasta)) | #配列の「中央値」取得
Max_len <- max(width(fasta)) | #配列の長さの「最大値」取得
Min_len <- min(width(fasta)) | #配列の長さの「最小値」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
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たす最初のインデックス
N50 <- sorted[obj][1] | #objがTRUEとなる最初の長さ
#本番(GC含量情報取得)↓
hoge <- alphabetFrequency(fasta) | #A,C,G,T,..の数を取得
#CG <- rowSums(hoge[,2:3]) | #C,Gの総数を計算
#ACGT <- rowSums(hoge[,1:4]) | #A,C,G,Tの総数を計算
CG <- apply(as.matrix(hoge[,2:3]), 1, sum)#C,Gの総数を計算
ACGT <- apply(as.matrix(hoge[,1:4]), 1, sum)#A,C,G,Tの総数を計算
GC_content <- sum(CG)/sum(ACGT) | #トータルのGC含量を計算
```

```
R Console
> #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)
> tmp
      [,1]      [,2]
[1,] "Total length (bp)" "2356019"
[2,] "Number of contigs" "117"
[3,] "Average length" "20136.9145299145"
[4,] "Median length" "213"
[5,] "Max length" "257728"
[6,] "Min length" "101"
[7,] "N50" "92304"
[8,] "GC content" "0.381489283405609"
> |
```

Contig # : 117  
 Total contig size : 2,356,019  
 Maximum contig size : 257,728  
 Minimum contig size : 101  
 N50 contig size : 92,304



# 塩基配列解析基礎2

①配列数の算出法length(fasta)や、  
②最短配列長min(width(fasta))も前のスライドで解説したものと同じです

```
rcode3.txt * x
in_f <- "out_gapClosed.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で読み込み↓
#本番(基本情報取得)↓
Total_len <- sum(width(fasta)) | #配列の「トータル長」取得
Number_of_contigs <- length(fasta) | #「配列数」取得
Average_len <- mean(width(fasta)) | #配列の「平均長」取得
Median_len <- median(width(fasta)) | #配列の「中央値」取得
Max_len <- max(width(fasta)) | #配列の長さの「最大値」取得
Min_len <- min(width(fasta)) | #配列の長さの「最小値」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
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たす最初のインデックス
N50 <- sorted[obj][1] | #objがTRUEとなる最初のインデックス
#本番(GC含量情報取得)↓
hoge <- alphabetFrequency(fasta) | #A,C,G,T,..の総数を取得
#CG <- rowSums(hoge[,2:3]) | #C,Gの総数を取得
#ACGT <- rowSums(hoge[,1:4]) | #A,C,G,Tの総数を取得
CG <- apply(as.matrix(hoge[,2:3]), 1, sum)#C,Gの総数を取得
ACGT <- apply(as.matrix(hoge[,1:4]), 1, sum)#A,C,G,Tの総数を取得
GC_content <- sum(CG)/sum(ACGT) | #トータルのGC含量取得
```

```
R Console
> #ファイルに保存
> tmp <- NULL
> tmp <- rbind(tmp, c("Total length (bp)", sum(width(fasta))))
> tmp <- rbind(tmp, c("Number of contigs", length(fasta)))
> tmp <- rbind(tmp, c("Average length", mean(width(fasta))))
> tmp <- rbind(tmp, c("Median length", median(width(fasta))))
> tmp <- rbind(tmp, c("Max length", max(width(fasta))))
> tmp <- rbind(tmp, c("Min length", min(width(fasta))))
> 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)
> tmp
      [,1]      [,2]
[1,] "Total length (bp)" "2356019"
[2,] "Number of contigs" "117"
[3,] "Average length"    "20136.9145299145"
[4,] "Median length"     "213"
[5,] "Max length"        "257728"
[6,] "Min length"        "101"
[7,] "N50"               "92304"
[8,] "GC content"        "0.381489283405609"
```

① Contig # : 117  
Total contig size : 2,356,019  
Maximum contig size : 257,728  
② Minimum contig size : 101  
N50 contig size : 92,304





# 塩基配列解析基礎3

このアセンブル結果の最短配列長は①101 bp。  
通常、アセンブル結果ファイルから一定の配列長(例:300 bp)未満のものは除去される

ID		21211				
Tool (Version)		Platanus (1.2.2)				
RunAccession or Filename	Download	Read length	Alias			
QC.1.trimmed.fastq.gz	<a href="#">QC.1.trimmed.fastq.gz</a>	N.A. bp	L.hokkaidonensis_MiSeq_denovo			
Download modified queries						
<ul style="list-style-type: none"> <li><a href="#">QC.1.trimmed.fastq.gz (Original size 189.4 MB)</a></li> <li><a href="#">QC.2.trimmed.fastq.gz (Original size 189.6 MB)</a></li> </ul>						
Download wgs file						
<ul style="list-style-type: none"> <li><a href="#">out_WGS.fasta.gz (Original size 2.3 MB)</a></li> </ul>						
Assembly statistics						
		Contig # : 117 Total contig size : 2,356,019 Maximum contig size : 257,728 Minimum contig size : 101 N50 contig size : 92,304				
Time						
Wait time	Start time	End time				
0: 0:9	2016-01-20 18:33:36	2016-01-21 10:10:06				
Command	Start time	End time	Log1	Log2	Result	MD5
platanus assemble -m 120 -f QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-20 18:33:37	2016-01-21 10:08:51		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus scaffold -c out_contig.fa -b out_contigBubble.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:02	2016-01-21 10:09:12		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>
platanus gap_close -c out_scaffold.fa -IP1 QC.1.trimmed.fastq QC.2.trimmed.fastq	2016-01-21 10:09:23	2016-01-21 10:09:34		<a href="#">View</a>	<a href="#">Download(2.2 MB)</a>	<a href="#">MD5</a>



①

# 塩基配列解析基礎3

①FASTA形式ファイルを読み込んで、指定した配列長以上のもののみ残してFASTA形式ファイルで出力する項目。②例題5は、out\_gapClosed.faを読み込んで、300 bp以上の配列のみhoge5.fastaファイルに保存するスクリプト

- 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 \(last modified 2013/06/18\)](#)
- 前処理 | フィルタリング | [重複のない配列セットを作成 \(last modified 2013/06/18\)](#)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出 \(last modified 2014/02/07\)](#)
- 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 \(last modified 2015/02/26\)](#)

## 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。

### 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

#### 1. multi-FASTAファイル

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014) を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```
in_f <- "hoge4"
out_f <- "hoge5"
param <- 50

#必要なパッケージをインストール
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNASTRINGSET("hoge4.fasta")

#本番
obj <- as.logical(fasta)
fasta <- fasta[obj,]

#ファイルに保存
writeXStringSet(fasta, "hoge5.fasta")
```

```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
param_length <- 300

#必要なパッケージをインストール
library(Biostrings)

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

#本番
obj <- as.logical(fasta)
fasta <- fasta[obj,]

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

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/platanusResult"
> list.files()
[1] "out_32merFrq.tsv"
[2] "out_contig.fa"
[3] "out_contigBubble.fa"
[4] "out_gapClosed.fa"
[5] "out_lib1_insFreq.tsv"
[6] "out_scaffold.fa"
[7] "out_scaffoldBubble.fa"
[8] "out_scaffoldComponent.tsv"
> |
```

# 塩基配列解析基礎3

①赤枠部分をコピー。入力ファイルを読み込んだ直後の②fastaオブジェクトは③117個の配列からなる。赤下線で見えているものが300 bp未満なのでフィルタリングされる

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)を処理する。

```
in_f <- "out_gapClosed.fa" #入力ファイル名
out_f <- "hoge5.fasta" #出力ファイル名
param_length <- 300 #配列長の閾値

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

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

#本番
obj <- as.logical(width(fasta) >= param_length) #objがTRUE
fasta <- fasta[obj] #確認して
fasta

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

```
R Console
> in_f <- "out_gapClosed.fa" #入力ファイル名
> out_f <- "hoge5.fasta" #出力ファイル名
> param_length <- 300 #配列長の閾値
>
> #必要なパッケージをロード
> library(Biostrings) #パッケージ名
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")$
> fasta #確認して
A DNASTringSet instance of length 117
      width seq          names
[1]  10520 ATCGATA...TTGCTG scaffold1_cov55
[2] 136075 TTTAAGG...AATTAA scaffold2_cov60
[3]  64531 CACCGTA...ATCTAG scaffold3_cov43
[4]  35091 CAAAATG...ATTGAA scaffold4_cov52
[5]    105 CAAAACG...GTGCTA scaffold5_cov98
...
[113]  113 CATTAGT...CGATGA scaffold113_cov168
[114]   747 CGGGAGT...TCACGC scaffold114_cov106
[115]  159 ACAAAC...TAAATT scaffold115_cov182
[116]  117 CTTTAAC...AATTGT scaffold116_cov184
[117]  263 ATGGGGT...CTTTCG scaffold117_cov182
> |
```

# 塩基配列解析基礎3

①width(fasta)は配列長情報からなる数値ベクトル。300 bpという閾値情報からなるparam\_lengthで条件判定した結果がobjに格納されている。

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa)を処理する。

```
in_f <- "out_gapClosed.fa"
out_f <- "hoge5.fasta"
param_length <- 300

#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

#本番
obj <- as.logical(width(fasta) >= param_length)
fasta <- fasta[obj]
fasta

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



```
R Console
> width(fasta)
 [1] 10520 136075 64531 35091 105 125
 [7] 125 467 125 257728 40654 68508
[13] 84851 512 94505 65563 54737 83975
[19] 117 125 125 73481 92304 6859
[25] 18430 154194 6893 125 352 96074
[31] 103 279 41161 63901 125 124
[37] 125 138 124 125 101 125
[43] 8311 6439 37583 10647 204783 108
[49] 64495 613 125 126 125 35878
[55] 125 251 156 539 96931 200
[61] 103 10030 14088 125 125 195
[67] 96261 31108 16776 112 19996 71496
[73] 58979 125 1918 125 125 168
[79] 213 125 125 125 714 115
[85] 125 794 195 1442 124 457
[91] 247 155 123 125 117 798
[97] 125 255 125 1439 117 1220
[103] 113 125 167 146 209 289
[109] 652 180 165 177 113 747
[115] 159 117 263
> |
```

# 塩基配列解析基礎3

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novo (Nagasaki et al., Genome Res., 2014) を実行して得られた multi-FASTA形式です。

```

in_f <- "out_gapClosed.fa"      #入力ファ
out_f <- "hoge5.fasta"         #出力ファ
param_length <- 300           #配列長の

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

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

#本番
obj <- as.logical(width(fasta) >= param_length)
fasta <- fasta[obj]           #objがTRUE
fasta                               #確認して

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

```

```

R Console
> obj <- as.logical(width(fasta) >= param_length)
> obj
 [1]  TRUE  TRUE  TRUE  TRUE  FALSE FALSE
 [7]  FALSE TRUE  FALSE TRUE  TRUE  TRUE
[13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[19]  FALSE FALSE FALSE TRUE  TRUE  TRUE
[25]  TRUE  TRUE  TRUE FALSE TRUE  TRUE
[31]  FALSE FALSE TRUE  TRUE FALSE FALSE
[37]  FALSE FALSE FALSE FALSE FALSE FALSE
[43]  TRUE  TRUE  TRUE  TRUE  TRUE  FALSE
[49]  TRUE  TRUE  FALSE FALSE FALSE  TRUE
[55]  FALSE FALSE FALSE  TRUE  TRUE  FALSE
[61]  FALSE TRUE  TRUE  FALSE FALSE FALSE
[67]  TRUE  TRUE  TRUE  FALSE TRUE  TRUE
[73]  TRUE  FALSE TRUE  FALSE FALSE FALSE
[79]  FALSE FALSE FALSE FALSE  TRUE  FALSE
[85]  FALSE  TRUE  FALSE  TRUE  FALSE  TRUE
[91]  FALSE FALSE FALSE FALSE FALSE  TRUE
[97]  FALSE FALSE FALSE  TRUE  FALSE  TRUE
[103] FALSE FALSE FALSE FALSE FALSE FALSE
[109]  TRUE  FALSE FALSE FALSE FALSE  TRUE
[115]  FALSE FALSE FALSE
> |

```

# 塩基配列解析基礎

R Console

オリジナルの117配列からなるfastaオブジェクトの中から、①objがTRUEとなる(300 bp以上の)配列は②52個。

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ (et al., Genome Res., 2014) を実行して得られた multi-FASTA形式です。

```

in_f <- "out_gapClosed.fa"      #入力ファイル名
out_f <- "hoge5.fasta"          #出力ファイル名
param_length <- 300            #配列長の閾値

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

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

#本番
obj <- as.logical(width(fasta) >= param_length)
fasta <- fasta[obj]             #objがTRUEの配列を抽出
fasta                             #確認して

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

```



```

> fasta
A DNASTringSet instance of length 117
      width seq          names
[1]  10520 ATCGATA...TTGCTG scaffold1_cov55
[2] 136075 TTTAAGG...AATTAA scaffold2_cov60
[3]  64531 CACCGTA...ATCTAG scaffold3_cov43
[4]  35091 CAAAATG...ATTGAA scaffold4_cov52
[5]    105 CAAAACG...GTGCTA scaffold5_cov98
...
[113]  113 CATTAGT...CGATGA scaffold113_cov168
[114]  747 CGGGAGT...TCACGC scaffold114_cov106
[115]  159 ACAAAC T...TAAATT scaffold115_cov182
[116]  117 CTTAAC...AATTGT scaffold116_cov184
[117]  263 ATGGGGT...CTTTCG scaffold117_cov182

> fasta[obj]
A DNASTringSet instance of length 52
      width seq          names
[1]  10520 ATCGATA...ATTGCTG scaffold1_cov55
[2] 136075 TTTAAGG...GAATTAA scaffold2_cov60
[3]  64531 CACCGTA...TATCTAG scaffold3_cov43
[4]  35091 CAAAATG...GATTGAA scaffold4_cov52
[5]    467 GTACCAA...TTAGAAA scaffold8_cov49
...
[48]   798 TGTTACG...TCTTCCA scaffold96_cov160
[49]  1439 AAATAAA...AAGCTTC scaffold100_cov358
[50]  1220 TTCTCAC...CGGAAAT scaffold102_cov196
[51]   652 CCAACCT...TAGAGTG scaffold109_cov158
[52]   747 CGGGAGT...TTCACGC scaffold114_cov106

```



# 塩基配列解析基礎3

①こういう上書きはアリです。もちろん fasta2みたいな別名にしてもいいが、ヒトゲノム配列などを取り扱うときにはノートPCレベルではメモリの的に厳しくなります

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

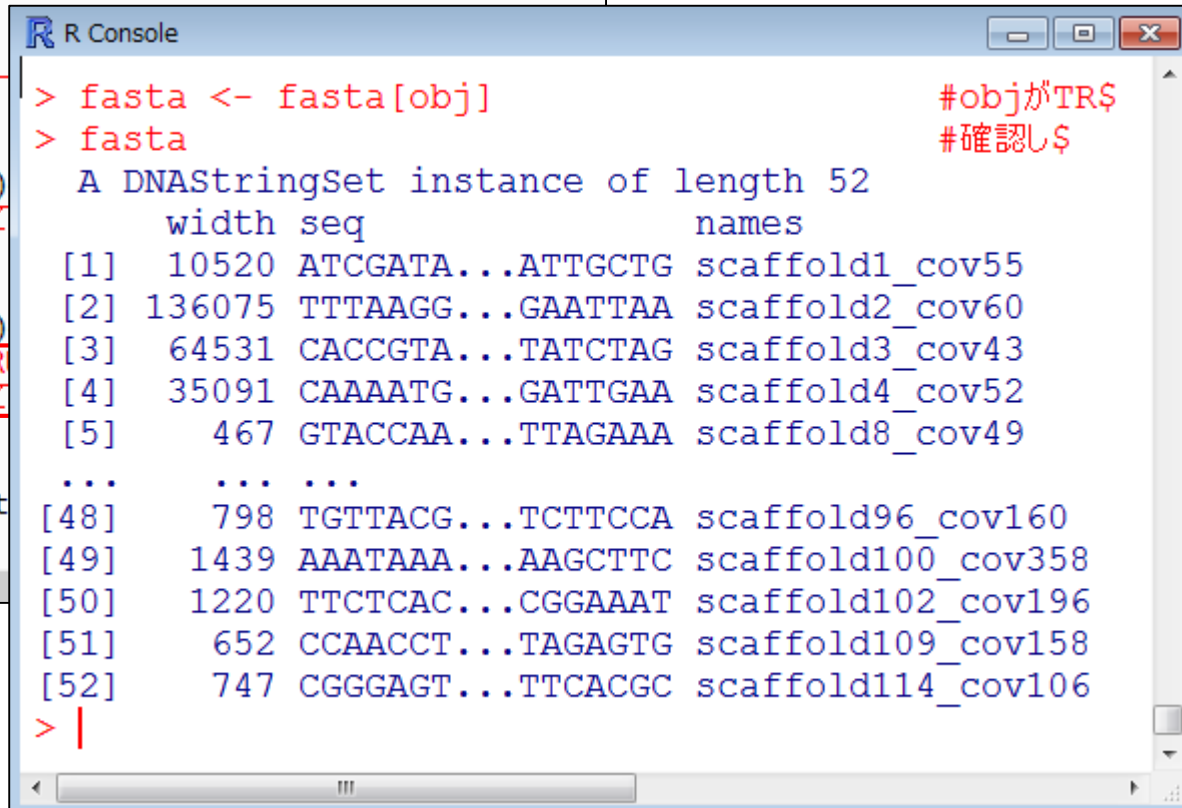
```
in_f <- "out_gapClosed.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
param_length <- 300 #配列長の閾値を指定
```

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

```
#入力ファイルの読み込み
fasta <- readDNASTringSet(in_f, format="fasta") #確認して
fasta
```

```
#本番
obj <- as.logical(width(fasta) >= param_length)
fasta <- fasta[obj] #objがTR$
fasta #確認して
```

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



```
R Console
> fasta <- fasta[obj] #objがTR$
> fasta #確認し$
A DNASTringSet instance of length 52
      width seq names
[1] 10520 ATCGATA...ATTGCTG scaffold1_cov55
[2] 136075 TTTAAGG...GAATTAA scaffold2_cov60
[3] 64531 CACCGTA...TATCTAG scaffold3_cov43
[4] 35091 CAAAATG...GATTGAA scaffold4_cov52
[5] 467 GTACCAA...TTAGAAA scaffold8_cov49
...
[48] 798 TGTTACG...TCTTCCA scaffold96_cov160
[49] 1439 AAATAAA...AAGCTTC scaffold100_cov358
[50] 1220 TTCTCAC...CGGAAAT scaffold102_cov196
[51] 652 CCAACCT...TAGAGTG scaffold109_cov158
[52] 747 CGGGAGT...TTCACGC scaffold114_cov106
> |
```

# 塩基配列解析基礎

①writeXStringSet関数を使えば、②fastaオブジェクトの中身を指定したファイルに書きだすことができる。  
 ①のXStringSetのXは、何でもよいみたいな意味。②fastaがDNAStringSetという形式で格納されていること、アミノ酸配列(Amino Acids)を格納する形式としてAAStringSetという形式が存在することから、それらを同じ関数で統一的に取り扱えるようにするため。

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリ (et al., Genome Res., 2014) を実行して得られた multi-FASTA形式ファイルです。

```
in_f <- "out_gapClosed.fa"
out_f <- "hoge5.fasta"
param_length <- 300

#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

#本番
obj <- as.logical(width(fasta) >= param_length)
fasta <- fasta[obj]
fasta

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

#入力ファイル名を指定してout\_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

#パッケージをロード  
 #確認して  
 #objがTRUEの配列を抽出  
 #確認して

```
R Console
> fasta <- fasta[obj]
> fasta
A DNAStringSet instance of length 52
  width seq          names
[1]  10520 ATCGATA...ATTGCTG scaffold1_cov55
[2] 136075 TTTAAGG...GAATTAA scaffold2_cov60
[3]  64531 CACCGTA...TATCTAG scaffold3_cov43
[4]  35091 CAAAATG...GATTGAA scaffold4_cov52
[5]    467 GTACCAA...TTAGAAA scaffold8_cov49
...
[48]   798 TGTTACG...TCTTCCA scaffold96_cov160
[49]  1439 AAATAAA...AAGCTTC scaffold100_cov358
[50]  1220 TTCTCAC...CGGAAAT scaffold102_cov196
[51]   652 CCAACCT...TAGAGTG scaffold109_cov158
[52]   747 CGGGAGT...TTCACGC scaffold114_cov106
> |
```





# 塩基配列解析基礎3

出力ファイルは、FASTA形式で保存した  
①hoge5.fasta。②1行あたりの塩基数を  
50個に指定している。

## 5. FASTA形式ファイル(out\_gapClosed.fa)の場合:

DDBJ Pipeline (Nagasaki et al., DNA Res., 2013)上で de novoゲノムアセンブリプログラムPlatanus (Kajitani et al., Genome Res., 2014)を実行して得られたmulti-FASTA形式ファイル(out\_gapClosed.fa; 約2.4MB)です。

```

in_f <- "out_gapClosed.fa"
out_f <- "hoge5.fasta"
param_length <- 300

#必要なパッケージをロード
library(Biostrings)

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

#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかど
fasta <- fasta[obj]
fasta

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fa

```



#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

#パッケージの読み込み

#確認してるだけです

#objがTRUEとなる要素のみ抽出  
#確認してるだけです



```

hoge5.fasta
> scaffold1_cov55↓
ATCGATATTATTTAGTTTTGAGAGCGCAAGTTCTCATCATAGTTATAGCT
ATGAATAAGCACTATAGTGTGGCGGCGATGGCCAGAAGGATACACCTGTT
CCCATGCCGAACACAGAAGTTAAGCTTCTGAGCGCCGATAGTAGTTGGGG
GATCGCTCCCTGCGAGGATAGGACGTTGCCATGCTTCAATGGAGGATTAG
CTCAGTTGGGAGAGCGTCTGCCTTACAAGCAGAGGGTACAGGTTTCGAGC
CCTGTATCCTCCATTGAGCCGTTAGCTCAGTTGGTAGAGCATCTGACTTT
TAATCAGAGGGTGCACAGTTCGAGACTGTACGGCTCATTGTCAATTTTA
TATTGGCGGGCATTGTATTGTAATGCCGACTTAGCTCAGTTGGCAGAGCA
TCTGTTTTGTA AACAGGAGGTCGAAGGTTCGAATCCTTTAGTCGGCATCA
ATAATGCGGAAGTAGTTAGTGGTAGAACATCACCTTGCCATGGTGGGGG
TCGCGGGTTCGAATCCCGTCTTCCGCTTTGCCAATCTTTATTATGCCGG
GGTGGCGGAATTGGCAGACGCACAGGACTTAAAATCCTGCGGTCAGTGAT
GACCGTACCGGTTTCGATCCCGGTCCTCGGCACCTTTTTGCACCCATAGCGC
AACTGGATAGAGTGTCTGACTACGAATCAGAAGGTTGTAGTTTCGACTCC
TACTGGGTGCATTTTCGGGAAGTAGCTCAGCTTGGTAGAGCACCTGGTTT
GGGACCAGGGGGTCGCAGGTTTGAATCCTGTCTTCCGATTTAATAATT
TGTTTTTAATTATTTATTATCTATTGTTTCATGTTTTTTCGCGGTGTAGCT
CACCTCCCTACAGCCCTCCGCTTCATACCCCCAGCTCCACCCCTTCCATCC

```



# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# アノテーション

アノテーション(遺伝子注釈付け)は、アSEMBル後の配列を入力として与え、どこ(座標)にどんな遺伝子(gene symbols; gene names; products)があり、どんなGene Ontology IDやKEGG Pathway上に存在するかなどを得る作業。①広範囲、②KEGG系、③バクテリアに特化、などいろいろあります

[Nucleic Acids Res.](#) 2016 Jan 4;44(D1):D286-93. doi: 10.1093/nar/gkv

## eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences.

[Huerta-Cepas J<sup>1</sup>](#), [Szklarczyk D<sup>2</sup>](#), [Forsslund K<sup>1</sup>](#), [Cook H<sup>3</sup>](#), [Heller D<sup>2</sup>](#), [Walter MC<sup>4</sup>](#), [Rattei T<sup>5</sup>](#), [Mende DR<sup>6</sup>](#), [Sunagawa S<sup>1</sup>](#), [Kuhn M<sup>7</sup>](#), [Jensen LJ<sup>3</sup>](#), [von Mering C<sup>5</sup>](#), [Bork P<sup>3</sup>](#).



[J Mol Biol.](#) 2015 Nov 14. pii: S0022-2836(15)00649-X. doi: 10.1016/j.jmb.2015.11.006. [Epub ahead of print]

## BlastKOALA and GhostKOALA: KEGG Tools for Functional Characterization of Genome and Metagenome Sequences.

[Kanehisa M<sup>1</sup>](#), [Sato Y<sup>2</sup>](#), [Morishima K<sup>3</sup>](#).

[Nucleic Acids Res.](#) 2007 Jul;35(Web Server issue):W182-5. Epub 2007 May 25.

## KAAS: an automatic genome annotation and pathway reconstruction server.

[Moriya Y<sup>1</sup>](#), [Itoh M](#), [Okuda S](#), [Yoshizawa AC](#), [Kanehisa M](#).



[BMC Genomics.](#) 2015 Aug 18;16:616. doi: 10.1186/s12864-015-1826-4.

## BEACON: automated tool for Bacterial GENome Annotation ComparisON.

[Kalkatawi M<sup>1</sup>](#), [Alam I<sup>2</sup>](#), [Bajic VB<sup>3</sup>](#).





他にrefFlat形式など様々なファイル形式が存在します。

# GFF/GTF形式ファイルの例

## GFF3形式 (シロイヌナズナ; TAIR10\_GFF3\_genes.gff)

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

## GTF形式 (ゼブラフィッシュ; Danio\_rerio.Zv9.75.gtf)

▲	A	B	C	D	E	F	G	H	I
1									
2									
3									
4									
5									
6	7	protein_coding	gene	100958	101715	.	+	.	gene_id "ENSDARG00000076051"; gene_name "CABZ01062994.1"; gene
7	7	protein_coding	transcript	100958	101715	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENS DART00000113409
8	7	protein_coding	exon	100958	100975	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENS DART00000113409
9	7	protein_coding	CDS	100958	100975	.	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENS DART00000113409
10	7	protein_coding	exon	101077	101715	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENS DART00000113409
11	7	protein_coding	CDS	101077	101715	.	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENS DART00000113409
12	7	protein_coding	gene	116160	117573	.	+	.	gene_id "ENSDARG00000088691"; gene_name "BX511027.1"; gene_sour
13	7	protein_coding	transcript	116160	117573	.	+	.	gene_id "ENSDARG00000088691"; transcript_id "ENS DART00000129330



# GFFの読み込み

①例題7。②ここで用いているGFF形式の入力ファイルは、③から取得しました。③をクリックしたつもりでよいw

- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [biomaRt\(Durink 2009\)](#) (last modified 2013/09/26)
- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [TxDb](#) | [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](#) | [TxDb](#) | [TxDb.\\*から](#) (last modified 2015/02/19)
- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [TxDb](#) | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2015/02/19) 推奨
- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [TxDb](#) | [GFF/GTF形式ファイルから](#) (last modified 2016/02/09) **NEW**
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [BSgenome](#) | [基本情報を取得](#) (last modified 2015/09/12)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTA形式](#) | [基本情報を取得](#) (last modified 2015/09/12)

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

QuasRパッケージを用いてゲノムへのマッピング結果からカウント情報を得たいときに、"TxDb"という形式のオブジェクトを利用する必要があります。ここでは、[GenomicFeatures](#)パッケージを用いて手元にあるGFF/GTF形式ファイルを入力としてTxDbオブジェクトを得るやり方を示します。基本的には[GenomicFeatures](#)パッケージ中のmakeTxDbFromGFF関数を用いてGFF/GTF形式ファイルを読み込むことでTxDbオブジェクトをエラーなく読み込むこと自体は簡単にできます。しかし、得られたTxDbオブジェクトとゲノムマッピング結果ファイルを用いてカウント情報を得る場合に、ゲノム配列提供元とアノテーション情報提供元が異なっているとエラーとなります。具体的には、GFF/GTFファイル中にゲノム配列中にない染色体名があるとエラーが出る場合があります。

### 1. [TAIR\(Lamesch et al., Nucleic Acids Res., 2012\)](#) から提供されているArabidopsisのGFF3形式ファイル([TAIR10 GFF3 genes.gff](#))の場合:

基本形です。エラーは出ませんが、2015年3月4日現在、ChrCが環状ではないと認識されてしまっています。

```
in_f <- "TAIR10_GFF3_genes.gff" #入力ファイル名を指定してin_fに格納(GFF/GTFファイル)
```

### ① 7. GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))の場合:

[Ensembl \(Flicek et al., 2014\)](#) から提供されている [Lactobacillus hokkaidonensis JCM 18461](#) [Mizawa et al., 2015](#) のデータです。

```
in_f <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3" #入
```

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

```
#本番(TxDbオブジェクトの作成)
txdb <- makeTxDbFromGFF(in_f, format="auto") #txdbオブジェクトの作成
txdb #確認してるだけです
```

# Ensembl解説

①GFFファイルはここから取得。②のgzip圧縮ファイルをダウンロードして解凍したものが入力ファイル。③のあたりがバージョン番号。概ね月単位でバージョン番号が上がっていく。

Lactobacillus hokkaidonensis JCM 18461 (ASM82939v1)

## Lactobacillus hokkaidonensis JCM 18461

FTP ディレクトリ /pub/bacteria/release-30/gff3/bacteria\_93\_collection/lactobacillus\_hokkaidonensis\_jcm\_18461 / ftp.ensemblgenomes.org

1 階層上のディレクトリへ

11/19/2015 12:51午前	290	<a href="#">CHECKSUMS</a>
11/18/2015 03:51午後	151,495	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3.gz</a>
11/18/2015 03:51午後	161,367	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.gff3.gz</a>
11/18/2015 03:51午後	3,509	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.nonchromosomal.gff3.gz</a>
11/18/2015 03:51午後	11,322	<a href="#">README</a>

① Download GFF3 FASTA - GFF3

② [Lactobacillus hokkaidonensis\\_jcm\\_18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3.gz](#)

③ 1.30

エクスプローラーでこの FTP サイトを表示するには、Alt キーを押して、[表示]をクリックし、[エクスプローラーで FTP サイトを開く]をクリックしてください。

## 1 階層上のディレクトリへ

11/19/2015 12:51午前	290	<a href="#">CHECKSUMS</a>
11/18/2015 03:51午後	151,495	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3.gz</a>
11/18/2015 03:51午後	161,367	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.gff3.gz</a>
11/18/2015 03:51午後	3,509	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.nonchromosomal.gff3.gz</a>
11/18/2015 03:51午後	11,322	<a href="#">README</a>



# Ensembl解説

①で、このゲノムの全貌をある程度把握可能。原著論文の情報なども合わせることで、②1 chromosome and 2 plasmids、環状ゲノムであることも認識可能。③でゲノム配列も取得できる

**Lactobacillus hokkaidonensis JCM 18461**  
Lactobacillus hokkaidonensis JCM 18461  
Provider [European Nucleotide Archive](#) | Taxonomy ID [1291742](#)

Search

e.g. [ligA](#) or [Chromosome:633970-636406](#) or [synthetase](#)

**About Lactobacillus hokkaidonensis JCM 18461**

**Genome assembly:** [GCA\\_000829395.1](#)

**Gene annotation**  
What can I find? Protein-coding and non-coding genes, splice variants, cDNA and protein sequences, non-coding RNAs.

**Gene counts**

Coding genes:	2,344
Non coding genes:	68
Small non coding genes:	68
Gene transcripts:	2,412

**Coordinate Systems**

Sequence	Length (bp)
<a href="#">Chromosome</a>	2277985

Sequence	Length (bp)
<a href="#">pLOOC260-1</a>	81630
<a href="#">pLOOC260-2</a>	40971

Ensembl Bacteria release 30 - December 2015 © EBI

# Ensembl解説

いろいろなものがあるって私はよくわかりませんが、GFFファイルと一緒に取り扱いたいときには、GFFファイルと似た名前の①を採用します

FTP ディレクトリ /pub/bacteria/release-30/fasta/bacteria\_93\_collection/lactobacillus\_hokkaidonensis\_jcm\_18461/dna/ /ftp.ensemblgenomes.org

エクスプローラーでこの FTP サイトを表示するには、Alt キーを押して、[表示] をクリックし、[エクスプローラーで FTP サイトを開く] をクリックしてください。

## 1階層上のディレクトリへ

11/19/2015 02:23午前	1,747	<a href="#">CHECKSUMS</a>
11/18/2015 06:29午後	706,063	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa.gz</a>
11/18/2015 06:29午後	744,218	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.genome.fa.gz</a>
11/18/2015 06:29午後	38,588	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.plasmid.fa.gz</a>
11/18/2015 06:29午後	25,855	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.plasmid.pL00C260-1.fa.gz</a>
11/18/2015 06:29午後	13,187	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.plasmid.pL00C260-2.fa.gz</a>
11/18/2015 06:29午後	744,218	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.toplevel.fa.gz</a>
11/18/2015 06:29午後	706,071	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.chromosome.Chromosome.fa.gz</a>
11/18/2015 06:29午後	744,236	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.genome.fa.gz</a>
11/18/2015 06:29午後	38,603	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.plasmid.fa.gz</a>
11/18/2015 06:29午後	25,863	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.plasmid.pL00C260-1.fa.gz</a>
11/18/2015 06:29午後	13,195	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.plasmid.pL00C260-2.fa.gz</a>
11/18/2015 06:29午後	744,236	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_rm.toplevel.fa.gz</a>
11/18/2015 06:29午後	706,071	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.chromosome.Chromosome.fa.gz</a>
11/18/2015 06:29午後	744,236	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.genome.fa.gz</a>
11/18/2015 06:29午後	38,603	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.plasmid.fa.gz</a>
11/18/2015 06:29午後	25,863	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.plasmid.pL00C260-1.fa.gz</a>
11/18/2015 06:29午後	13,195	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.plasmid.pL00C260-2.fa.gz</a>
11/18/2015 06:29午後	744,236	<a href="#">Lactobacillus hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna_sm.toplevel.fa.gz</a>
11/18/2015 06:29午後	3,619	<a href="#">README</a>



①例題7が読み込みの基本形。②GenomicFeaturesというパッケージが提供する③makeTxDbFromGFF関数を用いてGFFファイルを読み込んで、TxDbという独特の形式で取り扱えるようにする。入力ファイルは④「hoge - L.hokkaidonensis」中にある。

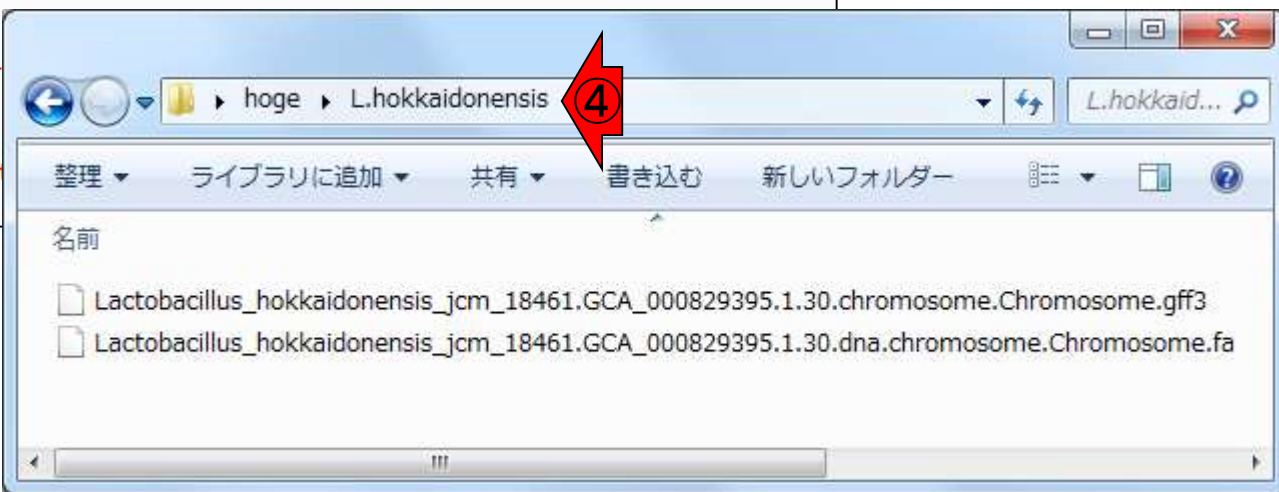
# GFFの読み込み

7. GFF形式ファイル(Lactobacillus hokkaidonensis jcm 18461.GCA\_000829395.1.30.chromosome.Chromosome.gff3) #入  
Ensembl (Flicek et al., 2014)から提供されている Lactobacillus hokkaidonensis

```
in_f <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
```

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

```
#本番(TxDbオブジェクトの作成)  
txdb <- makeTxDbFromGFF(in_f, format="auto")#  
txdb #確認し
```



```
R Console  
> getwd()  
[1] "C:/Users/kadota/Desktop/hoge/L.hokkaidonensis"  
> list.files()  
[1] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"  
[2] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"  
> |
```

①入力ファイルがGFF ver.3という形式になっていないみたいな警告メッセージが出ているが、読み込んだ後の②txdbオブジェクトは大丈夫そうだ、たぶん。

# GFFの読み込み

## 7. GFF3形式ファイル(Lactobacillus hokkaid

Ensembl (Flicek et al., 2014)から提供されてい

```
in_f <- "Lactobacillus_hokkaidone  
#必要なパッケージをロード  
library(GenomicFeatures)  
#本番(TxDbオブジェクトの作成)  
txdb <- makeTxDbFromGFF(in_f, for  
txdb
```

```
> txdb <- makeTxDbFromGFF(in_f, format="auto") #txdbオブジェクト$  
Import genomic features from the file as a GRanges object ... OK  
Prepare the 'metadata' data frame ... OK  
Make the TxDb object ... OK
```

警告メッセージ:  
.local(con, format, text, ...) で:  
gff-version directive indicates version is 3, not 3

#確認してるだけです



```
> txdb  
TxDb object:  
# Db type: TxDb  
# Supporting package: GenomicFeatures  
# Data source: Lactobacillus_hokkaidonensis_jcm_18461.GCA_00082$  
# Organism: NA  
# Taxonomy ID: NA  
# miRBase build ID: NA  
# Genome: NA  
# transcript_nrow: 2262  
# exon_nrow: 2262  
# cds_nrow: 2194  
# Db created by: GenomicFeatures package from Bioconductor  
# Creation time: 2016-02-09 15:47:24 +0900 (Tue, 09 Feb 2016)  
# GenomicFeatures version at creation time: 1.22.8  
# RSQLite version at creation time: 1.0.0  
# DBSCHEMAVERSION: 1.1  
> |
```

若干自信がないのは、GFFファイル読み込み後の①で見えている数値と、②Ensemblウェブサイト上で見られる数値が一致していないことに由来。2,344や2,412はプラスミドを含むものなのか詳細は不明

# GFFの読み込み

## 7. GFF3形式ファイル(Lactobacillus hokkaidoensis) R Console

Ensembl (Flicek et al., 2014)から提供されています

```
in_f <- "Lactobacillus_hokkaidoensis.gff3"

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

#本番(TxDbオブジェクトの作成)
txdb <- makeTxDbFromGFF(in_f, format="gff3")
```

```
> txdb <- makeTxDbFromGFF(in_f, format="gff3")
Import genomic features from the GFF3 file
Prepare the 'metadata' data frame
Make the TxDb object ... OK
警告メッセージ:
  .local(con, format, text, ...)
  gff-version directive indicated
> txdb
TxDb object:
# Db type: TxDb
# Supporting package: GenomicFeatures
# Data source: Lactobacillus_hokkaidoensis
# Organism: NA
# Taxonomy ID: NA
# miRBase build ID: NA
# Genome: NA
# transcript_nrow: 2262
# exon_nrow: 2262
# cds_nrow: 2194
# Db created by: GenomicFeatures
# Creation time: 2016-02-09 15:00:00
# GenomicFeatures version at creation: 1.22.0
# RSQLite version at creation time: 1.0.12
# DBSCHEMAVERSION: 1.1
> |
```



Assembly:	ASM82939v1, INSDC Assembly GCA_000829395.1, Nov 2014
Database version:	83.1
Base Pairs:	2,400,586
Golden Path Length:	2,400,586
Data source:	European Nucleotide Archive
Genebuild version:	2014-11-ENA
Genebuild method:	Generated from ENA annotation

Gene counts	
Coding genes:	2,344
Non coding genes:	68
Small non coding genes:	68
Gene transcripts:	2,412



Coordinate Systems	
chromosome	1 sequence
Filter	
Sequence	Length (bp)
Chromosome	2277985
plasmid	2 sequences
Filter	
Sequence	Length (bp)
pLOOC260-1	81630
pLOOC260-2	40971

# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# 転写物配列取得

multi-FASTAファイル(ゲノム配列情報)とGFFファイル(アノテーション情報)を同時に読み込むことで、例えば①トランスクリプトーム(転写物)配列情報を一気に取得することも可能。②例題5

- ・イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenomeとTxDbから](#) (last modified 2015/05/06)
- ・イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2016/02/09)
- ・イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | [公共DBから](#) (last modified 2015/05/06)
- ・イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2016/02/09)
- ・イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | [biomaRt\(Durinck 2009\)](#) (last modified 2015/02/20)
- ・イントロ | 一般 | 読み込み | xls形式 | [openxlsx](#) (last modified 2015/11/15)



## イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | [GenomicFeatures\(Lawrence\\_2013\)](#)

NEW

[GenomicFeatures](#)パッケージを主に用いてトランスクリプトーム配列を得るやり方を示します。「`extractTranscriptSeqs`」を行うことで、様々な例題を見ることができます。`transcriptsBy`関数部分は、`exonsBy`、`cdsBy`、`intronsByTranscript`、`fiveUTRsByTranscript`、`threeUTRsByTranscript`など様々な他の関数で置き換えることができます。



### 5. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

GFF3形式ファイル([Lactobacillus\\_hokkaidonensis\\_jcm\\_18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))とFASTA形式ファイル([Lactobacillus\\_hokkaidonensis\\_jcm\\_18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))を読み込むやり方です。[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus\\_hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

### 1. ヒト ([BSgenome](#))

対応するアノテーションファイルは82,960 transcripts

```
out_f <- "
param_bsge
param_txdb
```

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

```
#前処理(ゲノム配列取得)
library(parallel)
tmp <- ls(
genome <- BSgenome.Hsapiens.UCSC.hg19
```

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#?
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#?
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_f1に格納
```

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

```
#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです
```

```
#前処理(欲しい領域の座標情報取得)
hoge <- transcripts(txdb) #指定した範囲の座標情報を取得
hoge #確認してるだけです
```

```
#本番(配列取得)
```

①は、GFFファイル情報を保持したtxdbオブジェクトから、transcriptsという関数を用いて抽出したい転写物の座標情報を取得した結果をhogeに保存している

# 転写物配列取得

## 5. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))とFASTA形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))を読み込むやり方です。  
[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#)
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#)
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
```

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標情報取得)
hoge <- transcripts(txdb)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)

#後処理(description部分を変更)
```

```
> #前処理 (欲しい領域の座標情報取得)
> hoge <- transcripts(txdb) #指定した範囲の座標情報を取得
> hoge #確認してるだけです
```

R Console

```
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand | tx_id tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome    [ 360, 1676] + | 1 dnaA-1
[2] Chromosome    [1852, 2991] + | 2 dnaN-1
[3] Chromosome    [3233, 3457] + | 3 <NA>
[4] Chromosome    [3467, 4588] + | 4 recF-1
[5] Chromosome    [4588, 6531] + | 5 gyrB-1
...
[2258] Chromosome [2273924, 2275312] - | 2258 trmE-1
[2259] Chromosome [2275488, 2276288] - | 2259 <NA>
[2260] Chromosome [2276455, 2277288] - | 2260 <NA>
[2261] Chromosome [2277304, 2277648] - | 2261 <NA>
[2262] Chromosome [2277719, 2277853] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> |
```



① GFFファイルの見方がよくわかっていなくても、うまく読み込めているらしいことはわかる。

# 転写物配列取得

```
##gff-version 3
##sequence-region Chromosome 360 2277853
#!genome-build European Nucleotide Archive ASM82939
#!genome-version GCA_000829395.1
#!genome-date 2014-11
#!genome-build-accession GCA_000829395.1
#!genebuild-last-updated 2014-11
```

Chromosome	ena	gene	360	1676	+	ID=ge
Chromosome	ena	transcript	360	1676	+	ID=tr
Chromosome	ena	exon	360	1676	+	Pare
Chromosome	ena	CDS	360	1676	+	ID=C
###						
Chromosome	ena	gene	1852	2991	+	ID=ge
Chromosome	ena	transcript	1852	2991	+	ID=tr
Chromosome	ena	exon	1852	2991	+	Pare
Chromosome	ena	CDS	1852	2991	+	ID=C
###						
Chromosome	ena	gene	3233	3457	+	ID=ge
Chromosome	ena	transcript	3233	3457	+	ID=tr
Chromosome	ena	exon	3233	3457	+	Pare
Chromosome	ena	CDS	3233	3457	+	ID=C
###						
Chromosome	ena	gene	3467	4588	+	ID=ge



```

#指定した範囲の座標情報を取得
#確認してるだけです
2262 ranges and 2 metadata columns:

```

ranges	strand	tx_id	tx_name
<IRanges>	<Rle>	<integer>	<character>
[ 360, 1676]	+	1	dnaA-1
[1852, 2991]	+	2	dnaN-1
[3233, 3457]	+	3	<NA>
[3467, 4588]	+	4	recF-1
[4588, 5531]	+	5	gyrB-1
...	...	...	...
[2273924, 2275312]	-	2258	trmE-1
[2275488, 2276288]	-	2259	<NA>
[2276455, 2277288]	-	2260	<NA>
[2277304, 2277648]	-	2261	<NA>
[2277719, 2277853]	-	2262	rpmH-1

```

seqinfo: 1 sequence from an unspecified genome; no seqlengths
> |

```

座標情報取得)  
txs (txdb)



# 転写物配列取得

①in\_f1で指定したゲノム配列情報はここで登場。①ゲノム配列から、②で指定した座標情報の塩基配列を③(Biostringsパッケージが提供する)getSeq関数を用いて取得。④(Rsamtoolsパッケージが提供する)FaFile関数は、getSeq関数利用時に必要なおまじない。

## 5. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイル

GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.gff3](#))  
ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fasta](#))  
Ensembl (Flicek et al., 2014)から提供されている [Lactobacillus hokkaidonensis JCM 18461](#) (Tanizawa et al., 2015) のデータです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fasta"#ゲノム配列ファイル
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#アノテーションファイル
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
```

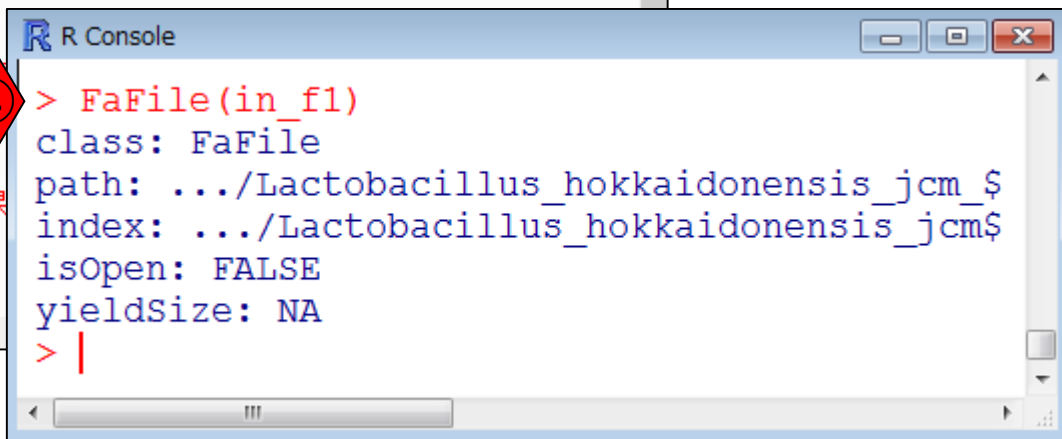
```
#必要なパッケージをロード
library(Rsamtools) #パッケージの読み込み
library(GenomicFeatures) #パッケージの読み込み
library(Biostrings) #パッケージの読み込み
```

```
#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです
```

```
#前処理(欲しい領域の座標情報取得)
hoge <- transcripts(txdb)
hoge #指定した範囲の座標情報取得
#確認してるだけです
```

```
#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果
fasta #確認してるだけです
```

```
#後処理(description部分を変更)
```



①getSeq実行後のfastaオブジェクトが、欲しいトランスクリプトーム配列情報ではあるが...

# 転写物配列取得

## 5. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))とFASTA形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))を読み込むやり方です。  
[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

```
library(Biostings) #パッケージの読み込み

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="auto") #txdbオブジェクトの作成
txdb #確認してるだけです

#前処理(欲しい領域の座標情報取得)
hoge <- transcripts(txdb)
hoge

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
fasta

#後処理(description部分を変更)
names(fasta) <- paste(seqnames(hoge),
                      end(ranges(hoge)))

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

```
R Console
> fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した$
> fasta #確認してるだけです
A DNASTringSet instance of length 2262
      width seq names
[1] 1317 GTGACTGATTTAGAA...AGCTAAAGCCATAG Chromosome
[2] 1140 ATGAAATTTACAATT...TTAGAACTTACTAA Chromosome
[3] 225 GTGCAAGAAGCAAAA...TTCAAAATGAGTAG Chromosome
[4] 1122 ATGATTTTAAAAGAA...AGGAGGAACCATAG Chromosome
[5] 1944 GTGAGCGATAAAAAA...ACTTAGATCTATAG Chromosome
...
[2258] 1389 GTGGCACAGACAGAG...GTTTAGGTAAATAG Chromosome
[2259] 801 ATGGCAATTTTACT...CTAGTGAGATGTAA Chromosome
[2260] 834 GTGAAAAGCACTTA...GTAGGCGCAAGTGA Chromosome
[2261] 345 ATGAGAAAGTCATAT...TAGATGAGCATTA Chromosome
[2262] 135 ATGAAGCGCACATTT...TATTATCTGCATAG Chromosome
> |
```



①のfastaオブジェクトをそのままFASTA形式で保存すると、②で見えているがままでのdescription情報が書きだされる。つまり、すべて"Chromosome"になってしまう

# 転写物配列取得

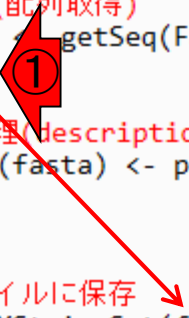
```

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果をfastaに格納
#確認してるだけです

#後処理(description部分を変更)
names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)),#"染色体名_start_end"に変更
                      end(ranges(hoge)), sep="_")#"染色体名_start_end"に変更
fasta #確認してるだけです

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

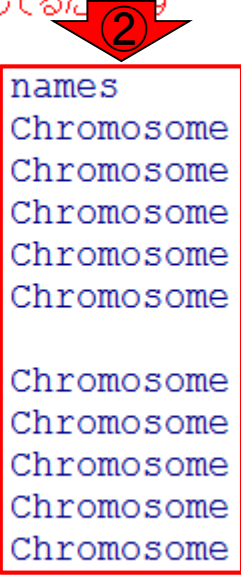
```



```

R Console
> fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した$
> fasta #確認してるだけです
A DNASTringSet instance of length 2262
      width seq
[1] 1317 GTGACTGATTTAGAA...AGCTAAAGCCATAG
[2] 1140 ATGAAATTTACAATT...TTAGAACTTACTAA
[3] 225 GTGCAAGAAGCAAAA...TTCAAAATGAGTAG
[4] 1122 ATGATTTTAAAAGAA...AGGAGGAACCATAG
[5] 1944 GTGAGCGATAAAAAA...ACTTAGATCTATAG
...
[2258] 1389 GTGGCACAGACAGAG...GTTTAGGTAAATAG
[2259] 801 ATGGCAATTTTACT...CTAGTGAGATGTAA
[2260] 834 GTGAAAAGCACTTA...GTAGGCGCAAGTGA
[2261] 345 ATGAGAAAGTCATAT...TAGATGAGCATTA
[2262] 135 ATGAAGCGCACATTT...TATTATCTGCATAG
> |

```



# 転写物配列取得

赤枠部分で行っているのは、description部分の記述内容を“Chromosome\_start\_end”としてどこの座標由来の塩基配列かがわかるようにしている。①pasteは、文字列を②sepオプションで指定した文字を間に挟んで連結する関数。③の例をみれば挙動がわかると期待。

```
#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得し
fasta #確認してるだけで

#後処理(description部分を変更)
names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)),#"染色体名
end(ranges(hoge)), sep=" ")#"染色体名_start_end" #確認してるだけです
fasta

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中
```



```
R Console
> paste("uge", "age", sep="_")
[1] "uge_age"
> seqnames(hoge)
factor-Rle of length 2262 with 1 run
Lengths:      2262
Values : Chromosome
Levels(1): Chromosome
> ranges(hoge)
IRanges of length 2262
      start      end width
[1]      360     1676 1317
[2]     1852     2991 1140
[3]     3233     3457  225
[4]     3467     4588 1122
[5]     4588     6531 1944
...
[2258] 2273924 2275312 1389
[2259] 2275488 2276288  801
[2260] 2276455 2277288  834
[2261] 2277304 2277648  345
[2262] 2277719 2277853  135
> |
```

# 転写物配列取得

① description部分が変わっていることがわかる。これを眺めるだけで、出力ファイルをみなくてももうまくいっていると判断できる(と油断していると時々落とし穴があるので注意)

```
#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果をfastaに格納
fasta #確認してるだけです

#後処理(description部分を変更)
names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)),#"染色体名_start_end"に変更
                      end(ranges(hoge)), sep="_")#"染色体名_start_end"に変更
fasta #確認してるだけです

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

```
R Console
> #後処理 (description部分を変更)
> names(fasta) <- paste(seqnames(hoge), start(ranges(hoge))$
+                       end(ranges(hoge)), sep="_")#"染色体$
> fasta #確認してるだけで$

A DNASTringSet instance of length 2262
      width seq          names
[1]  1317 GTGACTGATTT...AAAGCCATAG Chromosome_360_1676
[2]  1140 ATGAAATTTAC...AACTTACTAA Chromosome_1852_2991
[3]   225 GTGCAAGAAGC...AAATGAGTAG Chromosome_3233_3457
[4]  1122 ATGATTTTAAA...GGAACCATAG Chromosome_3467_4588
[5]  1944 GTGAGCGATAA...AGATCTATAG Chromosome_4588_6531
...
[2258] 1389 GTGGCACAGAC...AGGTAAATAG Chromosome_227392...
[2259]  801 ATGGCAATTTT...TGAGATGTAA Chromosome_227548...
[2260]  834 GTGAAAAGCA...GCGCAAGTGA Chromosome_227645...
[2261]  345 ATGAGAAAGTC...TGAGCATTAA Chromosome_227730...
[2262]  135 ATGAAGCGCAC...ATCTGCATAG Chromosome_227771...
> |
```



# Contents1

## ■ イン트로ダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得







# ① プロモーター配列取得

①例題10は、転写開始点上流100 bp、下流10 bpの領域を取得するコード。②はその領域情報。これだけだと③確かに110 bp分の領域になっていることの確認しかできない。

10. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

GFF3形式ファイル([Lactobacillus\\_hokkaidonensis\\_jcm\\_18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))とFASTA形式ファイル([Lactobacillus\\_hokkaidonensis\\_jcm\\_18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))を読み込むやり方です。  
[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus\\_hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge10.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 100
param_downstream <- 10
```

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

```
#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)
txdb
```

```
#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)
hoge
```

```
#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
```

```
> #前処理(欲しい領域の座標情報取得)
> hoge <- promoters(txdb, upstream=param_upstream, #指定した範囲の座標情報$
+               downstream=param_downstream) #指定した範囲の座標情報を取得
> hoge #確認してるだけです
```

GRanges object with 2262 ranges and 2 metadata columns:

	seqnames	ranges	strand	tx_id	tx_name
	<Rle>	<IRanges>	<Rle>	<integer>	<character>
[1]	Chromosome	[ 260, 369]	+	1	dnaA-1
[2]	Chromosome	[1752, 1861]	+	2	dnaN-1
[3]	Chromosome	[3133, 3242]	+	3	<NA>
[4]	Chromosome	[3367, 3476]	+	4	recF-1
[5]	Chromosome	[4488, 4597]	+	5	gyrB-1
...	...	...	...	...	...
[2258]	Chromosome	[2275303, 2275412]	-	2258	trmE-1
[2259]	Chromosome	[2276279, 2276388]	-	2259	<NA>
[2260]	Chromosome	[2277279, 2277388]	-	2260	<NA>
[2261]	Chromosome	[2277639, 2277748]	-	2261	<NA>
[2262]	Chromosome	[2277844, 2277953]	-	2262	rpmH-1

-----  
seqinfo: 1 sequence from an unspecified genome; no seqlengths

# ① プロモーター配列取得

①例題10は、転写開始点上流100 bp、下流10 bpの領域を取得するコード。元となっている転写開始点情報を②transcripts(txdb)でstrand情報も含めて比較するとよくわかる

## 10. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.chromosome.Chromosome.gff3](#))とFASTA形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))を読み込むやり方です。  
[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge10.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 100
param_downstream <- 10
```

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

#入力ファイルの読み込み  
 txdb <- makeTxDbFromGFF(in\_f2, in\_f1)  
 txdb

#前処理(欲しい領域の座標情報取得)  
 hoge <- promoters(txdb, upstream=param\_upstream, downstream=param\_downstream)  
 hoge

#本番(配列取得)  
 fasta <- getSeq(FaFile(in\_f1), hoge)

```
R Console

[2262] Chromosome [2277844, 2277953] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> transcripts(txdb)
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges      strand | tx_id tx_name
      <Rle>         <IRanges> <Rle> | <integer> <character>
[1] Chromosome [360, 1676] + | 1 dnaA-1
[2] Chromosome [1852, 2991] + | 2 dnaN-1
[3] Chromosome [3233, 3457] + | 3 <NA>
[4] Chromosome [3467, 4588] + | 4 recF-1
[5] Chromosome [4588, 6531] + | 5 gyrB-1
...
[2258] Chromosome [2273924, 2275312] - | 2258 trmE-1
[2259] Chromosome [2275488, 2276288] - | 2259 <NA>
[2260] Chromosome [2276455, 2277288] - | 2260 <NA>
[2261] Chromosome [2277304, 2277648] - | 2261 <NA>
[2262] Chromosome [2277719, 2277853] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

# ① プロモーター配列取得

①例題10は、転写開始点上流100 bp、下流10 bpの領域を取得するコード。最後まで実行した結果。転写開始点上流100 bp、下流10 bpの領域を取得するコードなので、②配列長が全て110 bpになっており妥当。

10. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

GFF3形式ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.chromosome.gff3](#))  
 ファイル([Lactobacillus hokkaidonensis jcm 18461.GCA\\_000829395.1.30.dna.chromosome.Chromosome.fa](#))  
[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus hokkaidonensis JCM 18461 \(Tanizawa et al., 2015\)](#) のデータです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge10.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 100
param_downstream <- 10
```

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="gff3")

#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
```

```
R Console
> #後処理(description部分を変更)
> names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)), #"染色体名_ $
+ end(ranges(hoge)), sep="_") #"染色体名_start_end$
> fasta #確認してるだけです
A DNASTringSet instance of length 2262
      width seq
[1] 110 ACAGTTGTGCAATATTAC...AGTCACTGTGACTGATT Chromosome_260_369
[2] 110 GAACTTTAATTATCAACA...ACCAATCATGAAATTTA Chromosome_1752_1861
[3] 110 AACACTGAATGGGCTGAA...TGATGAAGTGCAAGAAG Chromosome_3133_3242
[4] 110 TGAAGTTGACCAACGCCG...AACTTAAATGATTTTAA Chromosome_3367_3476
[5] 110 TTTTGACAACAACCAGTT...AACCATAGTGAGCGATA Chromosome_4488_4597
...
[2258] 110 TGCTTTGACCGCGTCTAA...TTGACCTGTGGCACAGA Chromosome_227530...
[2259] 110 TTAATGCAAGGTAGTT...TCCATAAATGGCAATTT Chromosome_227627...
[2260] 110 TTCTGCTGATCAGATGAG...ACGAAGAGTGAAAAAGC Chromosome_227727...
[2261] 110 CAAAAGGCAGAAAAGTA...TTACAGCATGAGAAAGT Chromosome_227763...
[2262] 110 TTATCTTTTTAGAATGTG...GCGAAATATGAAGCGCA Chromosome_227784...
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの$
```

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。例題10との違いは、上流の塩基配列数のみ

# 失敗例

①

11. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

10と基本的に同じでparam upstreamのところを200に変更しているだけですが、エラーが出るのがわかります。理由は、存在しないゲノム領域からプロモーター配列を取得しようとしたからです。

```

in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge11.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 200 #転写開始点上流の塩基配列数を指定
param_downstream <- 10 #転写開始点下流の塩基配列数を指定

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです

#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream,#指定した範囲の座標情報を取得
                 downstream=param_downstream)#指定した範囲の座標情報を取得
hoge #確認してるだけです

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果をfastaに格納
    
```

# 失敗例

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。例題10との違いは、上流の塩基配列数のみ。

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出るのがわかります。理由は、存在しないゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge11.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 200 #転写開始点上流の塩基配列数を指定
param_downstream <- 10
```

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
```

```
R Console
> #前処理(欲しい領域の座標情報取得)
> hoge <- promoters(txdb, upstream=param_upstream, #指定した範囲の座標情報$
+               downstream=param_downstream) #指定した範囲の座標情報を取得
> hoge #確認してるだけです
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome [160, 369] + | 1 dnaA-1
[2] Chromosome [1652, 1861] + | 2 dnaN-1
[3] Chromosome [3033, 3242] + | 3 <NA>
[4] Chromosome [3267, 3476] + | 4 recF-1
[5] Chromosome [4388, 4597] + | 5 gyrB-1
...
[2258] Chromosome [2275303, 2275512] - | 2258 trmE-1
[2259] Chromosome [2276279, 2276488] - | 2259 <NA>
[2260] Chromosome [2277279, 2277488] - | 2260 <NA>
[2261] Chromosome [2277639, 2277848] - | 2261 <NA>
[2262] Chromosome [2277844, 2278053] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

# 失敗例

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。例題10との違いは、上流の塩基配列数のみ。

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出るのがわかります。理由は、存在しないゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge11.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 200 #転写開始点上流の塩基配列数を指定
param_downstream <- 10
```

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
```

```
R Console

[2262] Chromosome [2277844, 2278053] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> transcripts(txdb)
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand | tx_id tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome [360, 1676] + | 1 dnaA-1
[2] Chromosome [1852, 2991] + | 2 dnaN-1
[3] Chromosome [3233, 3457] + | 3 <NA>
[4] Chromosome [3467, 4588] + | 4 recF-1
[5] Chromosome [4588, 6531] + | 5 gyrB-1
...
[2258] Chromosome [2273924, 2275312] - | 2258 trmE-1
[2259] Chromosome [2275488, 2276288] - | 2259 <NA>
[2260] Chromosome [2276455, 2277288] - | 2260 <NA>
[2261] Chromosome [2277304, 2277648] - | 2261 <NA>
[2262] Chromosome [2277719, 2277853] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

# 失敗例

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。例題10との違いは、上流の塩基配列数のみ。②hogeは取得したいプロモーター配列の座標情報。③getSeqを実行するとエラーが出る

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出る。これは、上流の塩基配列数のみ。hogeは取得したいプロモーター配列の座標情報。getSeqを実行するとエラーが出る

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
```

```
R Console
> hoge
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome    [ 160,  369]      +   |         1      dnaA-1
[2] Chromosome    [1652, 1861]      +   |         2      dnaN-1
[3] Chromosome    [3033, 3242]      +   |         3          <NA>
[4] Chromosome    [3267, 3476]      +   |         4      recF-1
[5] Chromosome    [4388, 4597]      +   |         5      gyrB-1
...
[2258] Chromosome [2275303, 2275512] -   |       2258      trmE-1
[2259] Chromosome [2276279, 2276488] -   |       2259          <NA>
[2260] Chromosome [2277279, 2277488] -   |       2260          <NA>
[2261] Chromosome [2277639, 2277848] -   |       2261          <NA>
[2262] Chromosome [2277844, 2278053] -   |       2262      rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]] (cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> |
```

# 思考停止するべからず

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。例題10との違いは、上流の塩基配列数のみ。②hogeは取得したいプロモーター配列の座標情報。③getSeqを実行するとエラーが出る。④fastaと打つと何か出力されるがうまく取れているわけではない！

11. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合  
 10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出る。これはゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.gb"
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$"
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1)

#前処理(欲しい領域の座標を抽出)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
```

```
R Console

[2262] Chromosome [2277844, 2278053]
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> fasta
A DNASTringSet instance of length 2262
      width seq
[1] 110 ACAGTTGTGCAATATTAC...AGTCACTGTGACTGATT Chromosome_260_369
[2] 110 GAACTTTAA...TGAATTTA Chromosome_1752_1861
[3] 110 AACACTGAA...TGCAAGAAG Chromosome_3133_3242
[4] 110 TGAAGTTGA...TGATTTTAA Chromosome_3367_3476
[5] 110 TTTTGACAA...TGAGCGATA Chromosome_4488_4597
...
[2258] 110 TGCTTTGAC...TGGCACAGA Chromosome_227530...
[2259] 110 TTACTATGC...TGGCAATTT Chromosome_227627...
[2260] 110 TTCTGCTGA...TGAAAAGC Chromosome_227727...
[2261] 110 CAAAAAGGC...TGAGAAAGT Chromosome_227763...
[2262] 110 TTATCTTTT TAGAATGTG...GCGAAATATGAAGCGCA Chromosome_227784...
```





# 思考停止するべからず

①例題11は、転写開始点上流200 bp、下流10 bpの領域を取得するコード。②このfastaオブジェクトの中身は、③このR Console画面上で以前に行っていた例題10(転写開始点上流100 bp、下流10 bpの領域を取得するコード)実行時に作成されたものが残っているだけである。その証拠に、④ここが110

11. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出る。これはゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標を抽出)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
```

```
R Console
[2262] Chromosome [2277844, 2278053]
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> fasta
A DNASTringSet instance of length 2262
      width seq
[1] 110 ACAGTTGTGCAATATTAC...AGTCACTGTGACTGATT Chromosome_260_369
[2] 110 GAACTTTAA...TGAATTTA Chromosome_1752_1861
[3] 110 AACACTGAA...TGCAAGAAG Chromosome_3133_3242
[4] 110 TGAAGTTGA...TGATTTTAA Chromosome_3367_3476
[5] 110 TTTTGACAA...TGAGCGATA Chromosome_4488_4597
...
[2258] 110 TGCTTTGAC...TGGCACAGA Chromosome_227530...
[2259] 110 TTA...TTTGGCAATTT Chromosome_227627...
[2260] 110 TTCTGCTGA...TGAAAAAGC Chromosome_227727...
[2261] 110 CAAAAAGGC...TGAGAAAGT Chromosome_227763...
[2262] 110 TTATCTTTT...GCGAAATATGAAGCGCA Chromosome_227784...
> |
```



# 大事な計算時は

①Rを再起動し、真っ新たな状態でコピーするのが一番スツキリ。私はいつもコレ。普段は②「いいえ」を押して作業スペースを保存せずに終了させるが、ここでは③キャンセルにしておく

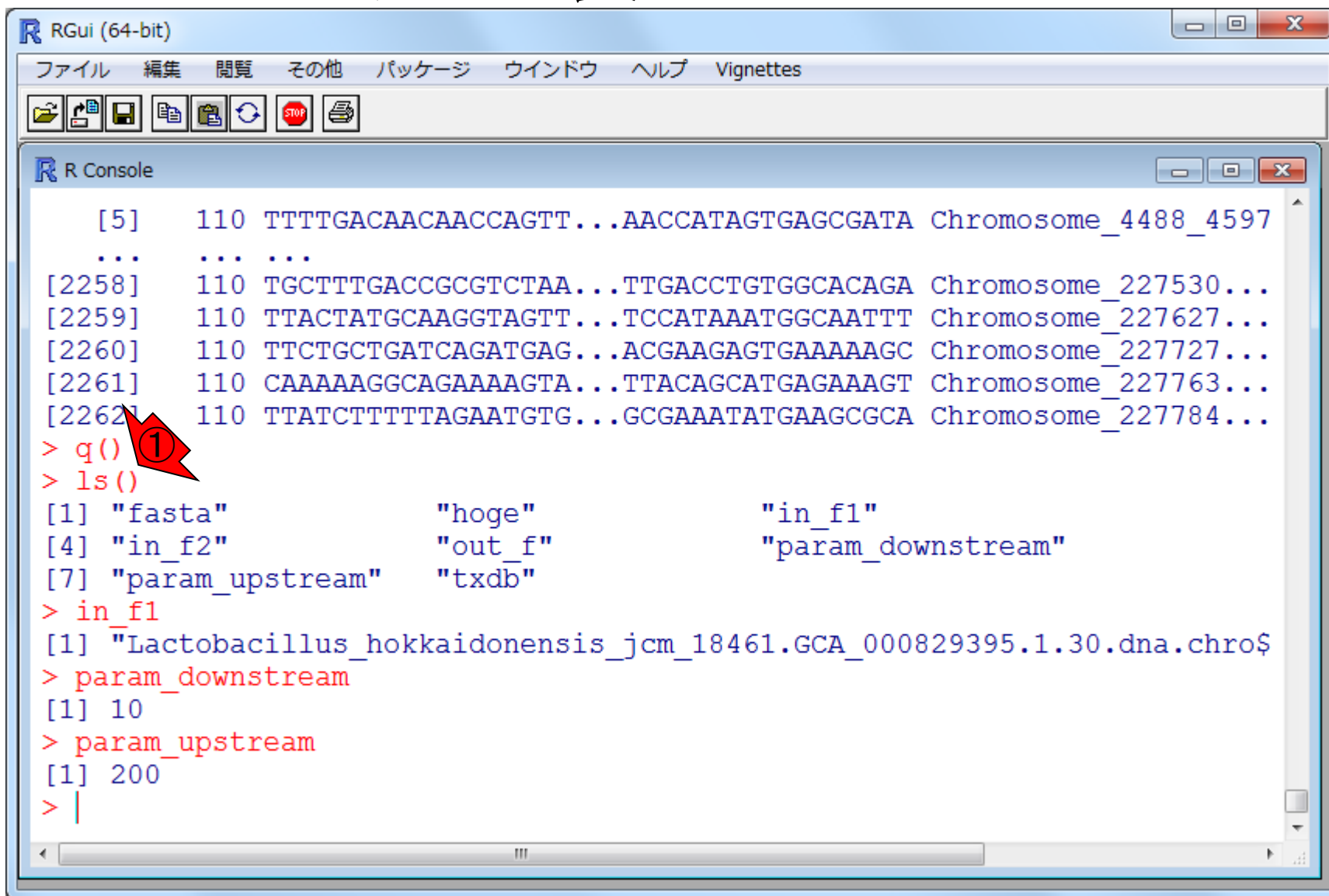


The screenshot shows the RGui interface. The R Console window displays the following text:

```
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
  record 2262 (Chromosome:2277844-2278053) was truncated
  file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> fasta
A DNASTringSet instance of length 2
width seq
[1] 110 ACAGTTGTGCAATATTAC...AGT 69
[2] 110 GAACTTTAATTATCAACA...ACC 1861
[3] 110 AACACTGAATGGGCTGAA...TGA 3242
[4] 110 TGAAGTTGACCAACGCCG...AAC 3476
[5] 110 TTTTGACAACAACCAGTT...AAC 4597
... ..
[2258] 110 TGCTTTGACCGCGTCTAA...TTC 0...
[2259] 110 TTAATGCAAGGTAGTT...TCCATAAATGGCAATTT Chromosome_227627...
[2260] 110 TTCTGCTGATCAGATGAG...ACGAAGAGTGAAAAAGC Chromosome_227727...
[2261] 110 CAAAAGGCAGAAAAGTA...TTACAGCATGAGAAAGT Chromosome_227763...
[2262] 110 TTATCTTTT TAGAATGTG...GCGAAATATGAAGCGCA Chromosome_227784...
```

A dialog box titled "質問" (Question) is overlaid on the console, asking "作業スペースを保存しますか?" (Save workspace?). It has three buttons: "はい(Y)" (Yes), "いいえ(N)" (No), and "キャンセル" (Cancel). Red arrows point to the "いいえ(N)" button (labeled 2) and the "キャンセル" button (labeled 3).

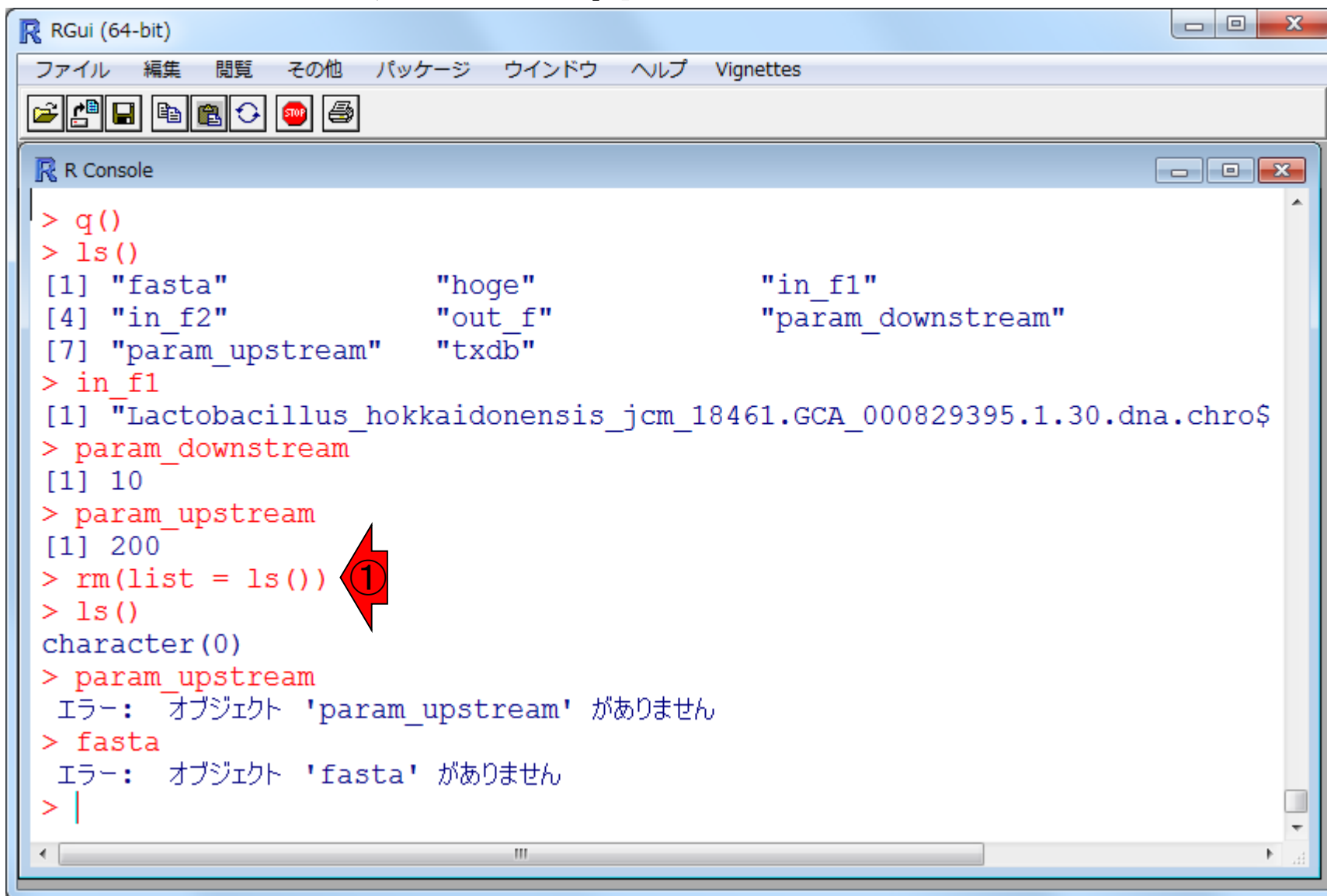
# オブジェクトの表示



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ Vignettes

R Console
[5] 110 TTTTGACAACAACCAGTT...AACCATAGTGAGCGATA Chromosome_4488_4597
... ..
[2258] 110 TGCTTTGACCGCGTCTAA...TTGACCTGTGGCACAGA Chromosome_227530...
[2259] 110 TTACTIONATGCAAGGTAGTT...TCCATAAATGGCAATTT Chromosome_227627...
[2260] 110 TTCTGCTGATCAGATGAG...ACGAAGAGTGAAAAAGC Chromosome_227727...
[2261] 110 CAAAAGGCAGAAAAGTA...TTACAGCATGAGAAAGT Chromosome_227763...
[2262] 110 TTATCTTTTTAGAAATGTG...GCGAAATATGAAGCGCA Chromosome_227784...
> q()
> ls()
[1] "fasta" "hoge" "in_f1"
[4] "in_f2" "out_f" "param_downstream"
[7] "param_upstream" "txdb"
> in_f1
[1] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chro$
> param_downstream
[1] 10
> param_upstream
[1] 200
> |
```

# オブジェクトの消去



```
> q()
> ls()
[1] "fasta"          "hoge"          "in_f1"
[4] "in_f2"          "out_f"         "param_downstream"
[7] "param_upstream" "txdb"

> in_f1
[1] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.chro$
> param_downstream
[1] 10
> param_upstream
[1] 200
> rm(list = ls())
> ls()
character(0)
> param_upstream
エラー: オブジェクト 'param_upstream' がありません
> fasta
エラー: オブジェクト 'fasta' がありません
> |
```

# Contents1

## ■ イントロダクション

- (Rで)塩基配列解析、アグリバイオ、NGSハンズオン講習会、
- 日本乳酸菌学会のNGS連載、HPCI講習会のPC環境

## ■ ゲノム解析

- NGSデータ解析戦略、DDBJ PipelineとRの関係、用語説明
- de novoアセンブリ実行、および結果をRで解析
- 塩基配列解析基礎1(塩基ごとの出現頻度解析)
- 各種テクニックや注意事項
- Rコードの解説
- 塩基配列解析基礎2(基本情報取得)
- 塩基配列解析基礎3(配列長でフィルタリング)
- アノテーション
- トランスクリプトーム配列
- プロモーター配列取得



# 例題11再実行

Tips。①ディレクトリの変更はsetwdでも可能。お約束のコマンドは、②のような任意のファイルに書き込んでおいて、R起動直後に無条件でコピペしておく(のが門田の習慣)。

## 11. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出ることも結構あります。理由は、存在のないゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829305_1.30.dna.chromosome.Chromosome.fa"#
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829305_1.30.chromosome.Chromosome.gff3"#入
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f2, format="auto")
txdb

#前処理(欲しい領域の座標情報取得)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)
hoge

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge)
```

#出力ファイル名を  
#転写開始点上流の  
#転写開始点下流の  
  
#パッケージの読み込み  
#パッケージの読み込み  
#パッケージの読み込み

```
setwd.txt * x
setwd("C:/Users/kadota/Desktop/hoge/L.hokkaidonensis")
getwd()↓
list.files()↓
```

```
R Console
> setwd("C:/Users/kadota/Desktop/hoge/L.hokkaidonensis")
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/L.hokkaidonensis"
> list.files()
[1] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829305_1.30.dna.chromosome.Chromosome.fa"
[2] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829305_1.30.chromosome.Chromosome.gff3"
[3] "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829305_1.30.chromosome.Chromosome.gff3"
> ls()
character(0)
> |
```

# 例題11再実行

①例題11再実行結果。②getSeq実行部分でコケるところまでは一緒。③以降はfastaオブジェクトがないのでそれを用いる部分は軒並みエラー祭りになっていることがわかる。重要なのは、エラーの原因を正確に把握すること。

11. GFF3形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出る。いゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.Chromosome.Chromosome.gff3"#人
out_f <- "hoge11.fasta" #出力ファイル名を指定してout_fに格納
param_upstream <- 200 #転写開始点と添の塩基配列数を指定
param_downstream <- 10
```

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

```
#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1)
txdb
```

```
#前処理(欲しい領域の座標を抽出)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)
hoge
```

```
#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
```

```
R Console
> fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果をfasta$
value[[3L]](cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
#確認してるだけです
> fasta
エラー: オブジェクト 'fasta' がありません
>
> #後処理 (description部分を変更)
names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)), #"染色体名_$
end(ranges(hoge)), sep="_")#"染色体名_start_end$
names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)), end(ranges($
オブジェクト 'fasta' がありません
> fasta #確認してるだけです
エラー: オブジェクト 'fasta' がありません
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの$
is(x, "XStringSet") でエラー: オブジェクト 'fasta' がありません
> |
```

エラーを把握すべく、②欲しい領域hogeと③getSeq実行部分を再度表示。

# エラー原因解説

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出るのがわかります。理由は、存在しないゲノム領域からプロモーター配列を取得しようとしたからです。

```

in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 100

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)

#前処理(欲しい領域の座標を抽出)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)

#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
    
```

**①** R Console

```

> hoge
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome    [ 160,  369]      + |         1      dnaA-1
[2] Chromosome    [1652, 1861]      + |         2      dnaN-1
[3] Chromosome    [3033, 3242]      + |         3          <NA>
[4] Chromosome    [3267, 3476]      + |         4      recF-1
[5] Chromosome    [4388, 4597]      + |         5      gyrB-1
...
[2258] Chromosome [2275303, 2275512]  - |        2258      trmE-1
[2259] Chromosome [2276279, 2276488]  - |        2259          <NA>
[2260] Chromosome [2277279, 2277488]  - |        2260          <NA>
[2261] Chromosome [2277639, 2277848]  - |        2261          <NA>
[2262] Chromosome [2277844, 2278053]  - |        2262      rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> |
    
```



# エラー原因解説

エラーの原因は、②に書かれているように、2262番目のレコード(Chromosome:2277844-2278053)が切り捨てられている(truncated)というもの。これは、③最後の転写開始点(2277853番目の塩基)の上流200 bpから下流10 bpの領域に相当する。

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーがでるゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_00082935.p.1.30.dna.csf"
in_f2 <- "Lactobacillus_hokkaidonensis_jcm_18461.GCA_00082935.p.1.30.dna.csf"
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(in_f1, in_f2)
txdb

#前処理(欲しい領域の座標を抽出)
hoge <- promoters(txdb, upstream=param_upstream, downstream=param_downstream)
hoge

#本番(配列取得)
fasta <- getSeq(FaFile(in_f2), hoge)
```

```
R Console
> hoge
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome    [ 160,  369]      + |         1      dnaA-1
[2] Chromosome    [1652, 1861]      + |         2      dnaN-1
[3] Chromosome    [3033, 3242]      + |         3          <NA>
[4] Chromosome    [3267, 3476]      + |         4      recF-1
[5] Chromosome    [4388, 4597]      + |         5      gyrB-1
...
[2258] Chromosome [2275303, 2275512] - |       2258      trmE-1
[2259] Chromosome [2276279, 2276488] - |       2259          <NA>
[2260] Chromosome [2277279, 2277488] - |       2260          <NA>
[2261] Chromosome [2277639, 2277848] - |       2261          <NA>
[2262] Chromosome [2277844, 2278053] - |       2262      rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
  record 2262 (Chromosome:2277844-2278053) was truncated
  file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_00082935.p.1.30.dna.csf
> |
```

# エラー原因解説

②seqinfo関数を使うことで、in\_f1で指定したファイルの配列長情報(2277985 bp)を取得可能。  
③で取得しようとしていた領域の一部が存在しないことが原因である。

## 11. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む

10と基本的に同じでparam\_upstreamのところを200に変更しているだけですが、エラーが出るのは、存在のないゲノム領域からプロモーター配列を取得しようとしたからです。

```
in_f1 <- "Lactobacillus_
in_f2 <- "Lactobacillus_
out_f <- "hoge11.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF(
txdb

#前処理(欲しい領域の座標)
hoge <- promoters(txdb,
downstream=p

hoge

#本番(配列取得)
fasta <- getSeq(FaFile(
```

```
R Console

[1] Chromosome [ 160, 369] + | 1 dnaA-1
[2] Chromosome [1652, 1861] + | 2 dnaN-1
[3] Chromosome [3033, 3242] + | 3 <NA>
[4] Chromosome [3267, 3476] + | 4 recF-1
[5] Chromosome [4388, 4597] + | 5 gyrB-1
... ..
[2258] Chromosome [2275303, 2275512] - | 2258 trmE-1
[2259] Chromosome [2276279, 2276488] - | 2259 <NA>
[2260] Chromosome [2277279, 2277488] - | 2260 <NA>
[2261] Chromosome [2277639, 2277848] - | 2261 <NA>
[2262] Chromosome [2277844, 2278053] - | 2262 rpmH-1
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths
> fasta <- getSeq(FaFile(in_f1), hoge)
value[[3L]](cond) でエラー:
record 2262 (Chromosome:2277844-2278053) was truncated
file: Lactobacillus_hokkaidonensis_jcm_18461.GCA_000829395.1.30.dna.c$
> seqinfo(FaFile(in_f1))
Seqinfo object with 1 sequence from an unspecified genome:
seqnames seqlengths isCircular genome
Chromosome 2277985 NA <NA>
> |
```

# 例題12が推奨コード

①例題12は、取得予定の座標が存在するかどうかを判定し、存在しないものをフィルタリングする部分を追加したコード(甲斐政親氏提供)。今問題となっている②の領域が除去(filter out)されればよい。それを行ってくれる

12. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場

11と基本的に同じですが、取得予定領域座標がゲノム配列の範囲外にあるものをフィルタ(甲斐政親氏提供情報)

```
in_f1 <- "Lactobacillus
in_f2 <- "Lactobacillus
out_f <- "hoge12.fasta"
param_upstream <- 200
param_downstream <- 10

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

#入力ファイルの読み込み
txdb <- makeTxDbFromGFF
txdb

#前処理(欲しい領域の座標)
hoge <- promoters(txdb,
                  downstream=p

#前処理(取得予定領域座標)
hoge3 <- param upstream
```

```
R Console
> #前処理 (欲しい領域の座標情報取得)
> hoge <- promoters(txdb, upstream=param_upstream, #指定した範囲の座標情$
+                 downstream=param_downstream) #指定した範囲の座標情報を取得
> hoge #確認してるだけです
GRanges object with 2262 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome    [ 160,  369]      + |         1      dnaA-1
[2] Chromosome    [1652, 1861]      + |         2      dnaN-1
[3] Chromosome    [3033, 3242]      + |         3          <NA>
[4] Chromosome    [3267, 3476]      + |         4      recF-1
[5] Chromosome    [4388, 4597]      + |         5      gyrB-1
...
[2258] Chromosome [2275303, 2275512] - |       2258      trmE-1
[2259] Chromosome [2276279, 2276488] - |       2259          <NA>
[2260] Chromosome [2277279, 2277488] - |       2260          <NA>
[2261] Chromosome [2277639, 2277848] - |       2261          <NA>
[2262] Chromosome [2277844, 2278053] - |       2262      rpmH-1
-----
seqinfo: 1 sequence (1 circular) from an unspecified genome; no seqle$
>
> |
```



③が取得予定の座標が存在するかどうかを判定し、存在しないものをフィルタリングする部分(甲斐政親氏提供)。

# ① 例題12が推奨コード

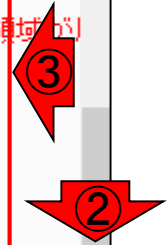
## 12. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

11と基本的に同じですが、取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリングする部分を追加しています。(甲斐政親氏 提供情報)

```
#前処理(取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリング)
hoge3 <- param_upstream + param_downstream#欲しいプロモーター配列長情報をhoge3に格納
obj <- sapply(hoge, #領域座標情報hoge1に対して下記関数を実行した結果をobjに格納
  function(hoge2) { #関数を定義
    start <- hoge2@ranges[[1]][1] #start位置情報を取得
    end <- hoge2@ranges[[1]][hoge3] #end位置情報を取得
    chr_num <- grep(paste(hoge2@seqnames@values, "$", sep=""), hoge@seqnames@values)#調査中の領域が!
    chr_length <- seqlengths(FaFile(in_f1))[chr_num]#該当染色体の配列長情報を取得
    if ((start >= 0) & (end <= chr_length)){#該当染色体の範囲内にある場合は
      return(TRUE) #TRUE
    } else { #そうでなければ
      return(FALSE) #FALSE
    }
  })
hoge <- hoge[obj] #objがTRUEとなる要素のみ抽出した結果をhoge1に格納
hoge #確認してるだけです

#本番(配列取得)
fasta <- getSeq(FaFile(in_f1), hoge) #配列情報を取得した結果をfastaに格納
fasta #確認してるだけです

#後処理(description部分を変更)
```



赤枠のフィルタリング実行後の状態。②エラーの原因であった2262番目の領域がなくなっていることがわかる。

# 例題12が推奨コード

12. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

11と基本的に同じですが、取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリングする部分を追加しています。(甲斐政親氏 提供情報)

```
#前処理(取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリング)
hoge3 <- param_upstream
obj <- sapply(hoge,
  function(hoge2) {
    start <- hoge2@range
    end <- hoge2@ranges
    chr_num <- grep(paste0("chr", chr_num), chr_name)
    chr_length <- seqle
    if ((start >= 0) &
      return(TRUE)
    } else {
      return(FALSE)
    }
  })
hoge <- hoge[obj]
hoge

#本番(配列取得)
fasta <- getSeq(FaFile(
fasta

#後処理(description部分)
```

```
R Console
+         return(FALSE)           #FALSE
+     }
+ })
> hoge <- hoge[obj]              #objがTRUEとなる要素のみ抽出し$
> hoge                           #確認してるだけです
GRanges object with 2261 ranges and 2 metadata columns:
      seqnames      ranges strand |      tx_id      tx_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1] Chromosome [ 160,  369]   +   |         1      dnaA-1
[2] Chromosome [1652, 1861]   +   |         2      dnaN-1
[3] Chromosome [3033, 3242]   +   |         3         <NA>
[4] Chromosome [3267, 3476]   +   |         4      recF-1
[5] Chromosome [4388, 4597]   +   |         5      gyrB-1
...
[2257] Chromosome [2273881, 2274090] -   |       2257      gidA-1
[2258] Chromosome [2275303, 2275512] -   |       2258      trmE-1
[2259] Chromosome [2276279, 2276488] -   |       2259         <NA>
[2260] Chromosome [2277279, 2277488] -   |       2260         <NA>
[2261] Chromosome [2277639, 2277848] -   |       2261         <NA>
-----
seqinfo: 1 sequence (1 circular) from an unspecified genome; no seqle$
> |
```



# ① 例題12が推奨コード

12. GFF形式のアノテーションファイルとFASTA形式のゲノム配列ファイルを読み込む場合:

11と基本的に同じですが、取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリングする部分を追加しています。(甲斐政親氏 提供情報)

```
#前処理(取得予定領域座標がゲノム配列の範囲外にあるものをフィルタリング)
hoge3 <- param_upstream
obj <- sapply(hoge,
  function(hoge2) {
    start <- hoge2@ranges
    end <- hoge2@ranges
    chr_num <- grep(paste0("chr", chr_num), seqnames(hoge2))
    chr_length <- seqinfo(hoge2)$chr_lengths[chr_num]
    if ((start >= 0) & (end <= chr_length))
      return(TRUE)
    } else {
      return(FALSE)
    }
  })
hoge <- hoge[obj]
hoge
```

```
#本番(配列取得)
fasta <- getSeq(FaFile("genome.fasta"), hoge)
fasta

#後処理(description部分を変更)
```

```
R Console
>
> #後処理(description部分を変更)
> names(fasta) <- paste(seqnames(hoge), start(ranges(hoge)), #"染色体名_$(
+   end(ranges(hoge)), sep="_") #"染色体名_start_end$(
> fasta
#確認してるだけです
A DNASTringSet instance of length 2261
      width seq
[1] 210 CTTGATACAACGGACGTA...AGTCACTGTGACTGATT Chromosome_160_369
[2] 210 CCTTAAAATGGAGCTAAA...ACCAATCATGAAATTTA Chromosome_1652_1861
[3] 210 TGCACATTTTTGAGGGGA...TGATGAAGTGCAAGAAG Chromosome_3033_3242
[4] 210 CATTTATTACAGTGGGAC...AACTTAAATGATTTTAA Chromosome_3267_3476
[5] 210 GCGAATATCCAATTTTA...AACCATAGTGAGCGATA Chromosome_4388_4597
...
[2257] 210 GTGATGCATTGAAGGATG...GTTAAAATGGAGCAAG Chromosome_227388...
[2258] 210 GATTGCACCAGCTAGTGA...TTGACCTGTGGCACAGA Chromosome_227530...
[2259] 210 GAAACGTAATGCCCGTAA...TCCATAAATGGCAATTT Chromosome_227627...
[2260] 210 CGGTTTCATCGTAACTGGG...ACGAAGAGTGAAAAAGC Chromosome_227727...
[2261] 210 GCGCACATTTCAACCAA...TTACAGCATGAGAAAGT Chromosome_227763...
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの$(
> |
```





# Contents2

## ■ トランスクリプトーム解析

- インTRODクシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)



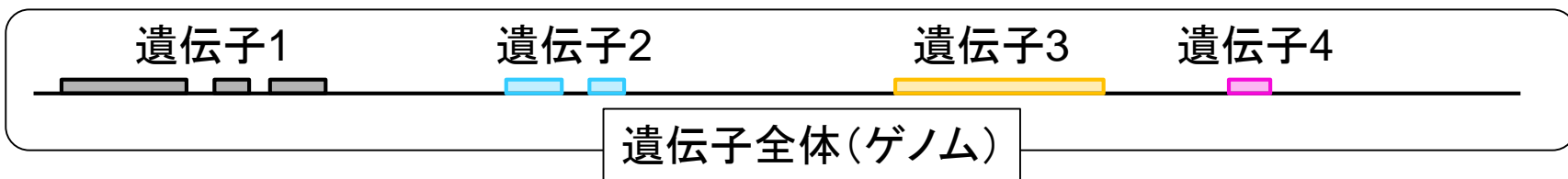


働いているRNAの種類  
や量を調べるのが目的

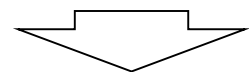
光刺激

# トランスクリプトーム解析

- ある状態のあるサンプル(例:目)のあるゲノムの領域



・どの染色体上のどの領域にどの遺伝子があるかは調べる個体(例:ヒト)が同じなら不変(目だろうが心臓だろうが...)

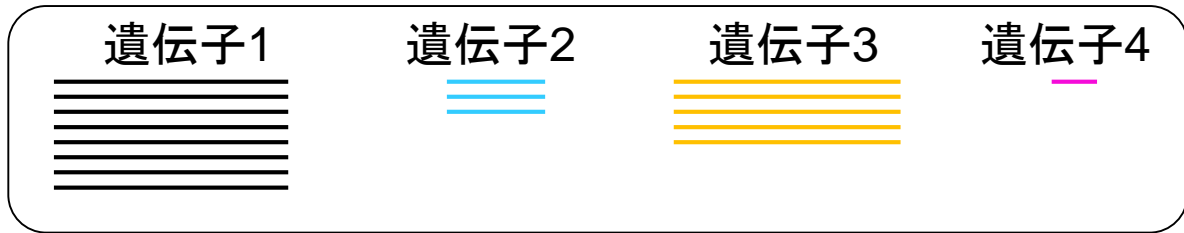


- ・遺伝子2は光刺激に应答して発現亢進
- ・遺伝子4も光刺激に应答して発現亢進

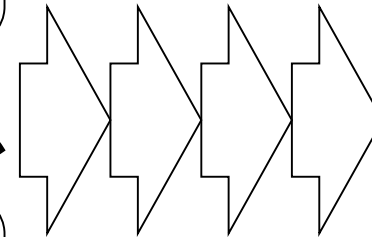
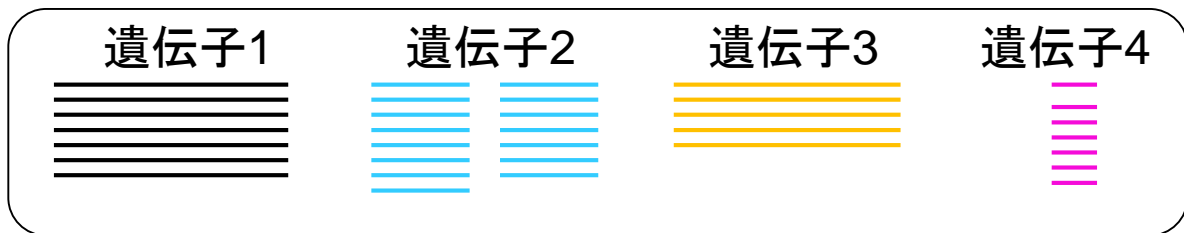
状態の異なる複数サンプルのデータを取得して解析するのが一般的。サンプル間比較。

# トランスクリプトーム解析

## ■ 光刺激前 (T1) の目のトランスクリプトーム



## ■ 光刺激後 (T2) の目のトランスクリプトーム

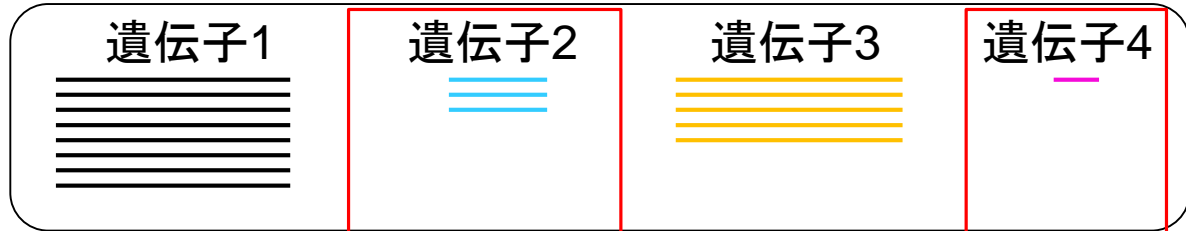


	T1	T2
遺伝子1	8	7
遺伝子2	3	15
遺伝子3	5	5
遺伝子4	1	7
...	...	...

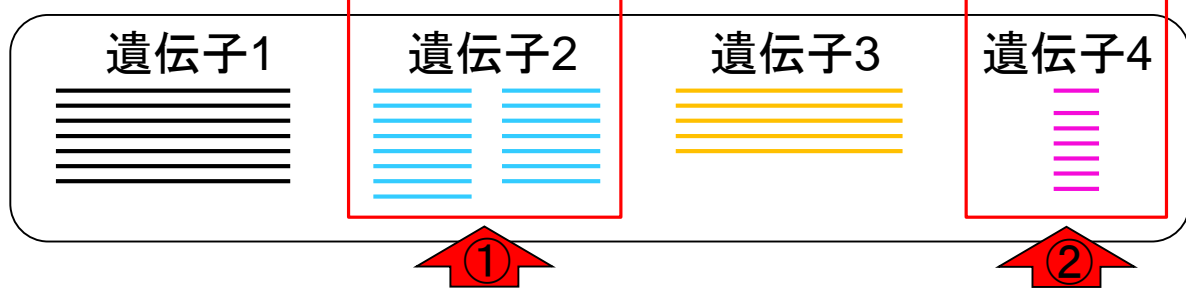
具体的な目的は、①や②の  
発現変動遺伝子同定など。

# トランスクリプトーム解析

## ■ 光刺激前 (T1) の目のトランスクリプトーム



## ■ 光刺激後 (T2) の目のトランスクリプトーム



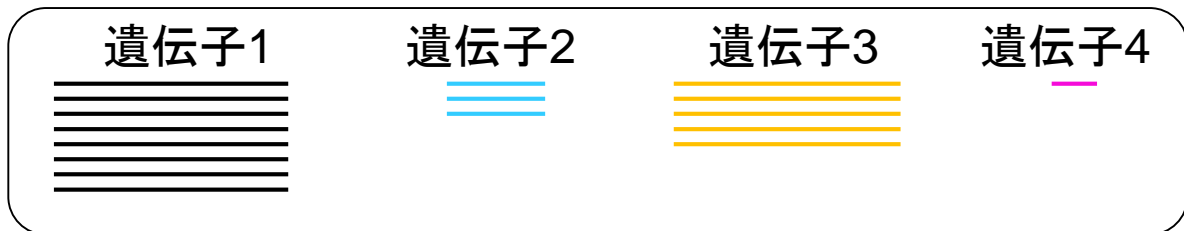
これがいわゆる  
「遺伝子発現行列」

	T1	T2
遺伝子1	8	7
遺伝子2	3	15
遺伝子3	5	5
遺伝子4	1	7
...	...	...

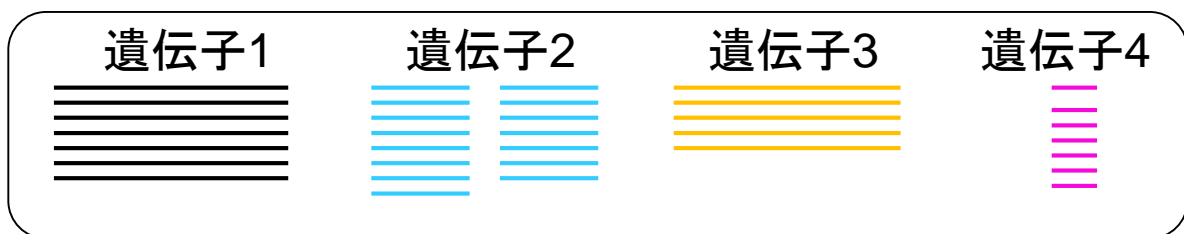
# データ取得

現在はNGSの利用が主流。  
NGSを用いたRNAの配列決定  
(sequencing)なので、RNA-seq

## ■ 光刺激前 (T1) の目のトランスクリプトーム

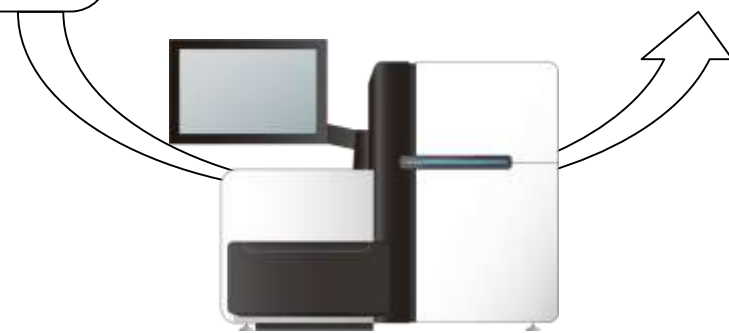


## ■ 光刺激後 (T2) の目のトランスクリプトーム

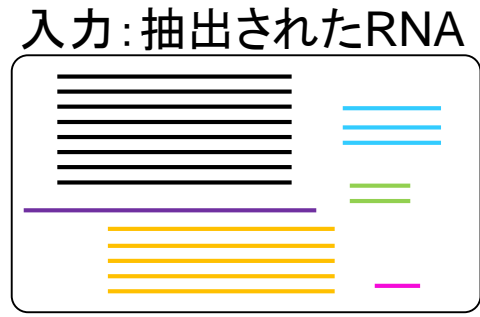


これがいわゆる  
「遺伝子発現行列」

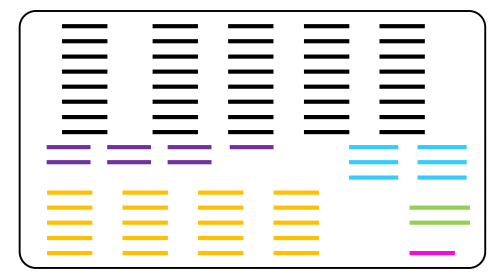

	T1	T2
遺伝子1	8	7
遺伝子2	3	15
遺伝子3	5	5
遺伝子4	1	7
...	...	...



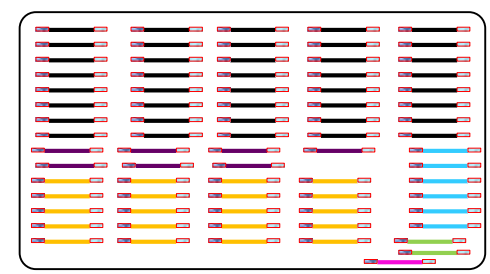

# RNA-seq概略



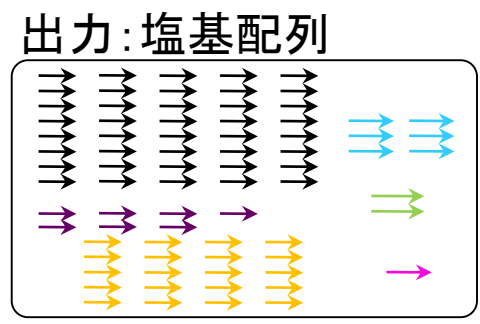
断片化



アダプター付加



NGSで  
配列決定



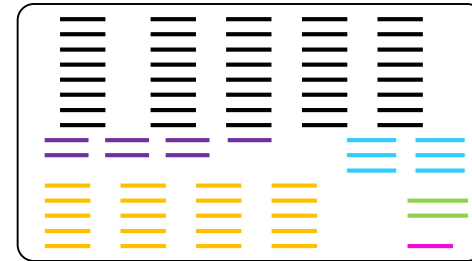
# RNA-seq概略

NGSの出力は、リードと呼ばれる100塩基程度の短い配列が延々と続く巨大なファイル。各矢印が1つのリードに相当。この段階では、まだどのリードがどの転写物由来かは不明(なので灰色一色)。

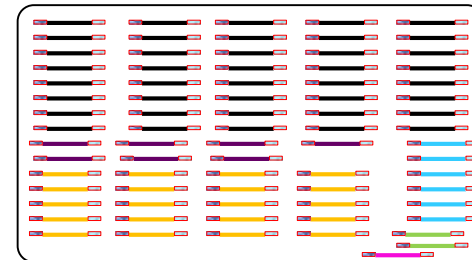
入力: 抽出されたRNA



断片化



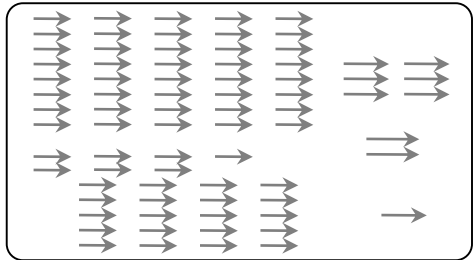
アダプター付加



NGSで  
配列決定



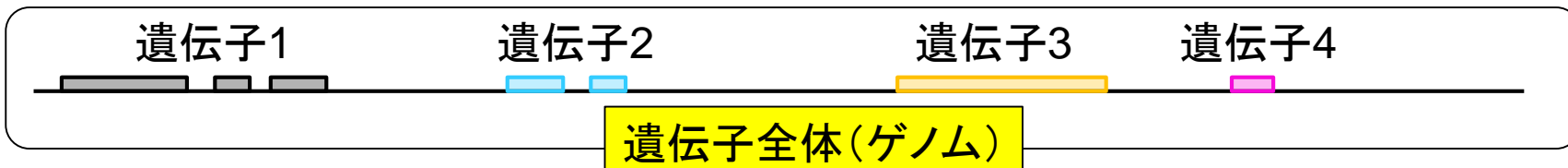
出力: 塩基配列



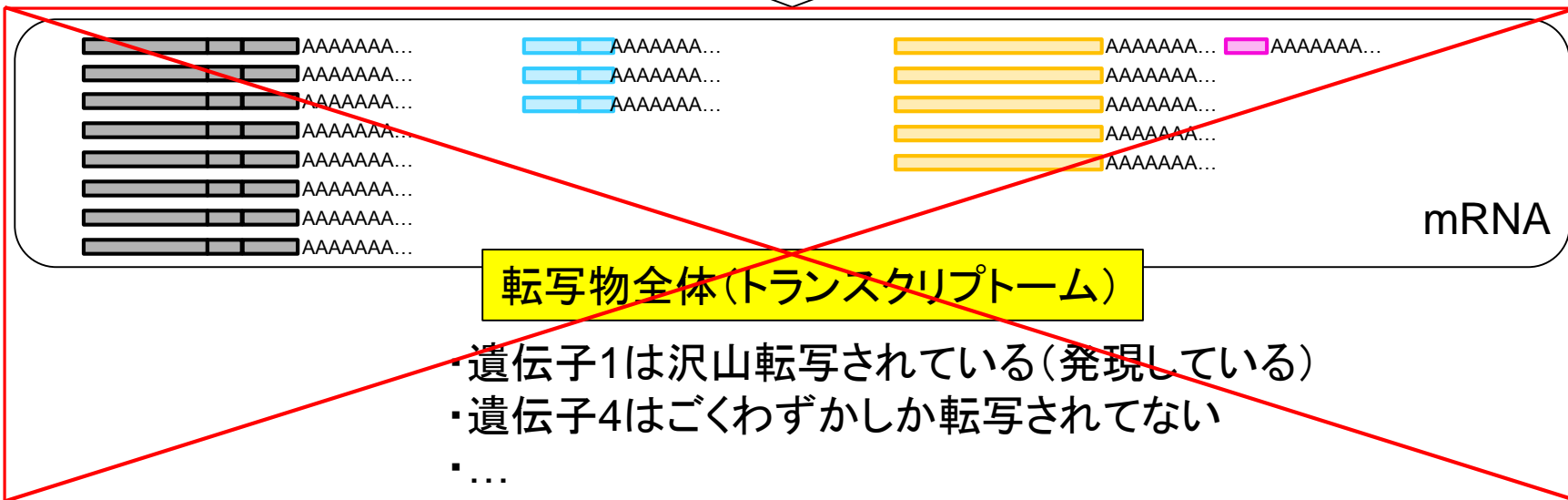
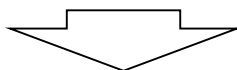
# 遺伝子 ≠ 転写物

赤枠部分の表現は、本当は不正確。昔は実験機器の解像度が事実上遺伝子レベルだった。遺伝子発現解析という表現はその名残り。

- ある状態のあるサンプル(例:目)のあるゲノムの領域



- ・どの染色体上のどの領域にどの遺伝子があるかは調べる個体(例:ヒト)が同じなら不変(目だろうが心臓だろうが...)





ある遺伝子領域から転写 (transcription) されている転写物 (transcript) は、1種類とは限らない

# 遺伝子 ≠ 転写物

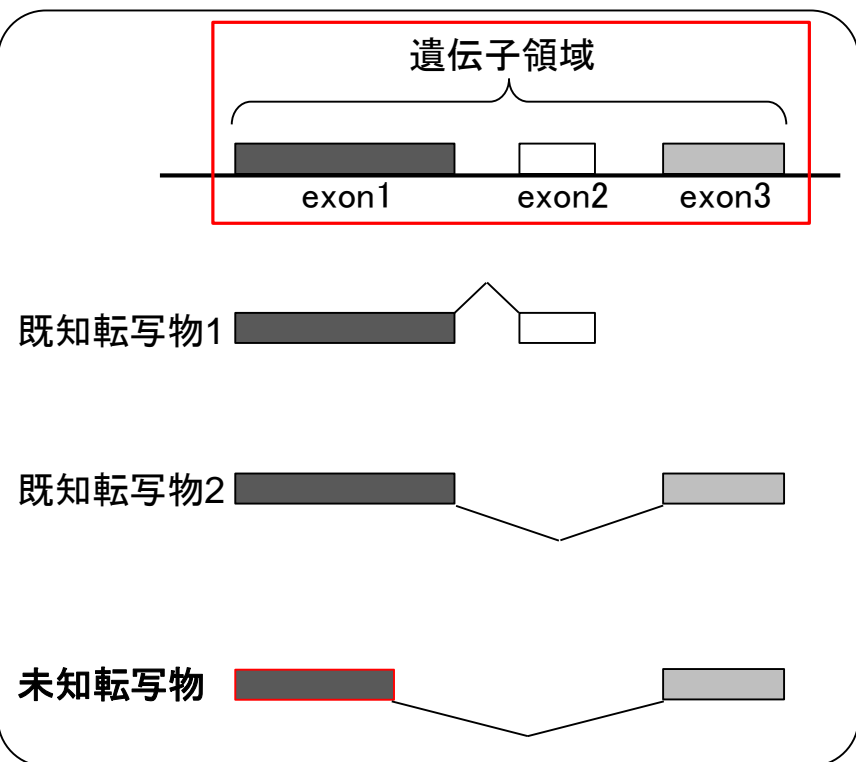
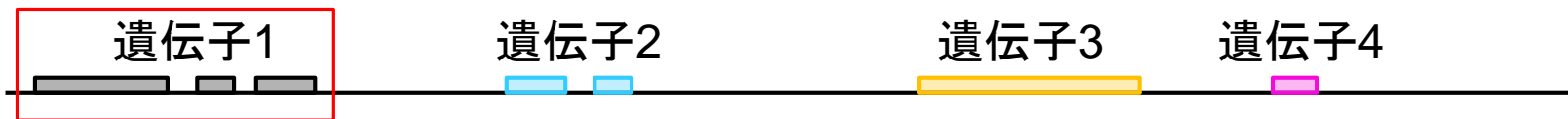
- ある状態のあるサンプル (例: 目) のあるゲノムの領域



# 遺伝子 ≠ 転写物

- ある状態のあるサンプル(例:目)の

ある遺伝子領域から転写(transcription)されている転写物(transcript)は、1種類とは限らない。例えば、遺伝子1の領域では、3種類の真の転写物が存在し、そのうち2種類は既知とする。

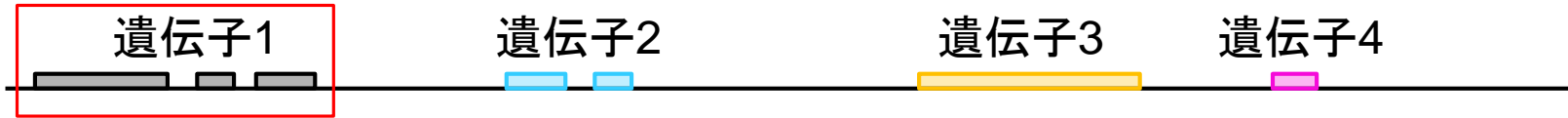


真の転写物情報

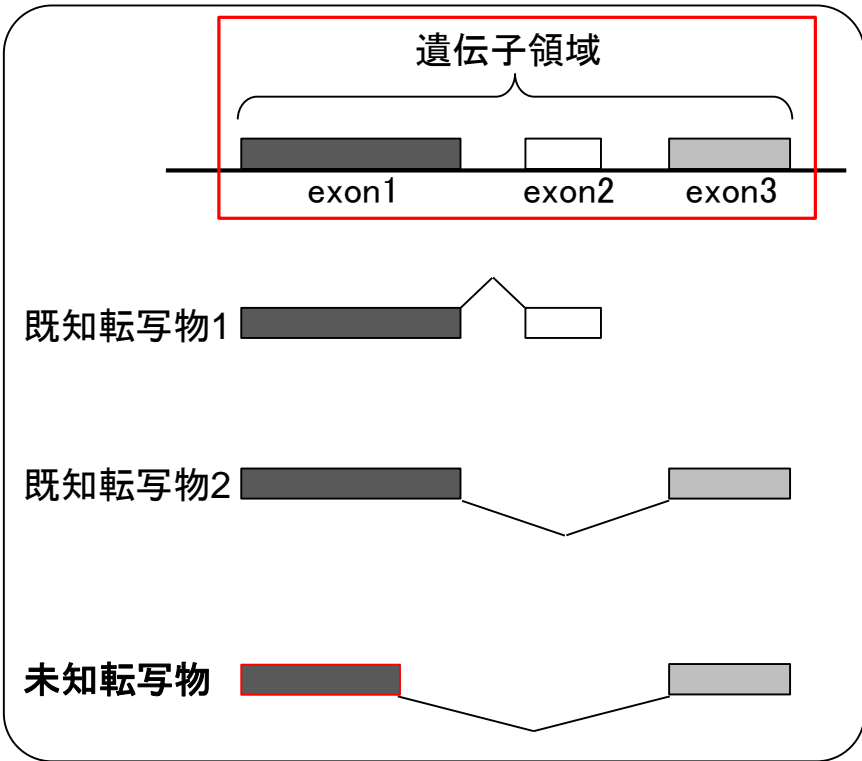
実際の細胞内(例:目のサンプル)での発現情報(働いている度合い)が①のような感じだったとする

# 遺伝子 ≠ 転写物

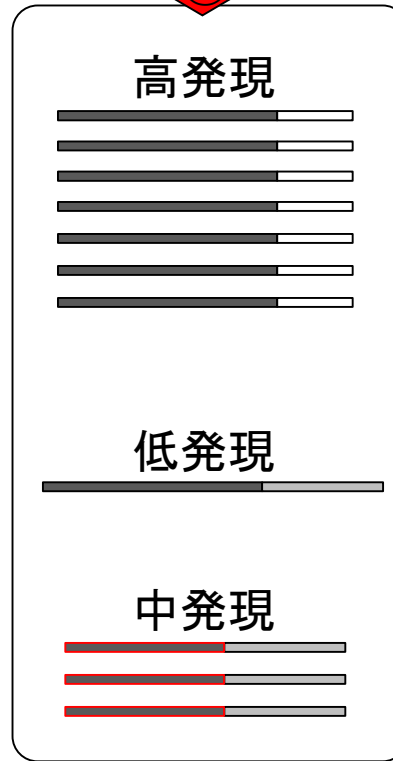
- ある状態のあるサンプル(例:目)のあるゲノムの領域



①



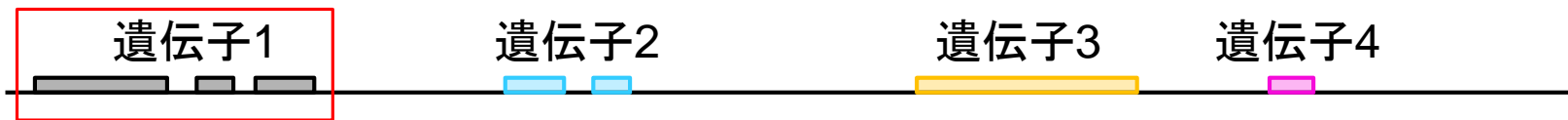
真の転写物情報



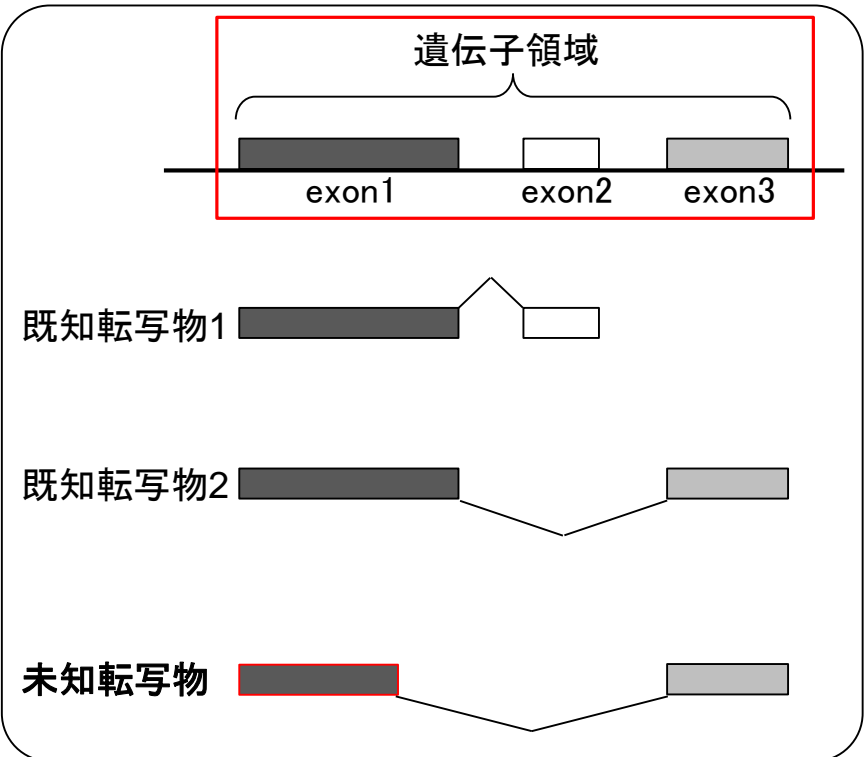
真の発現情報

# 遺伝子 ≠ 転写物

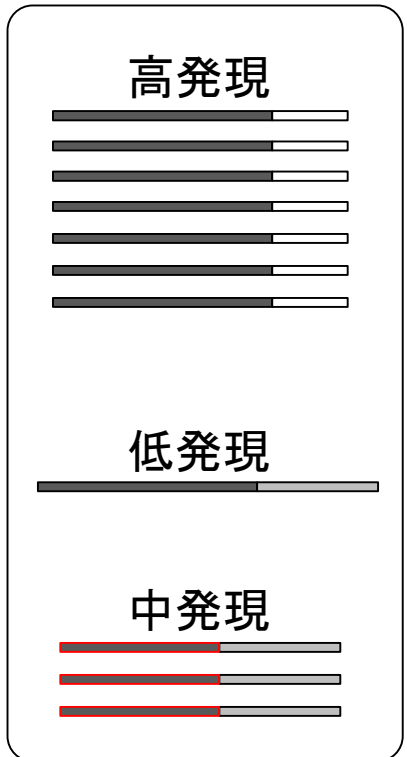
- ある状態のあるサンプル(例:目)のあるゲノムの領域



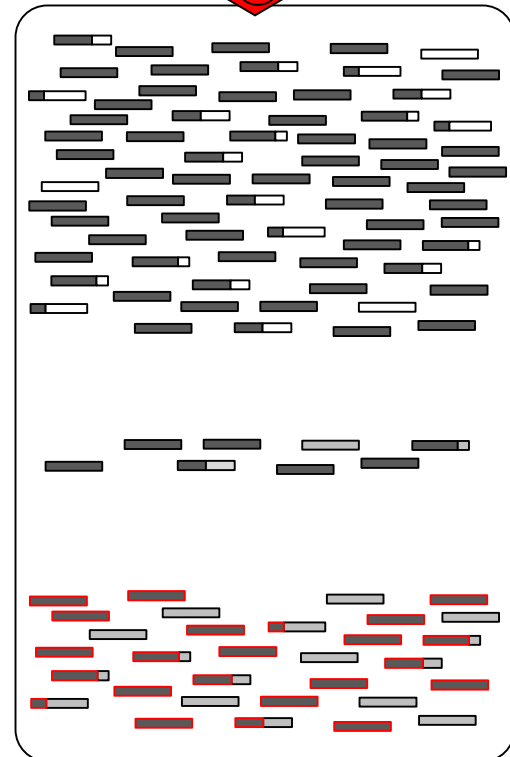
①



真の転写物情報



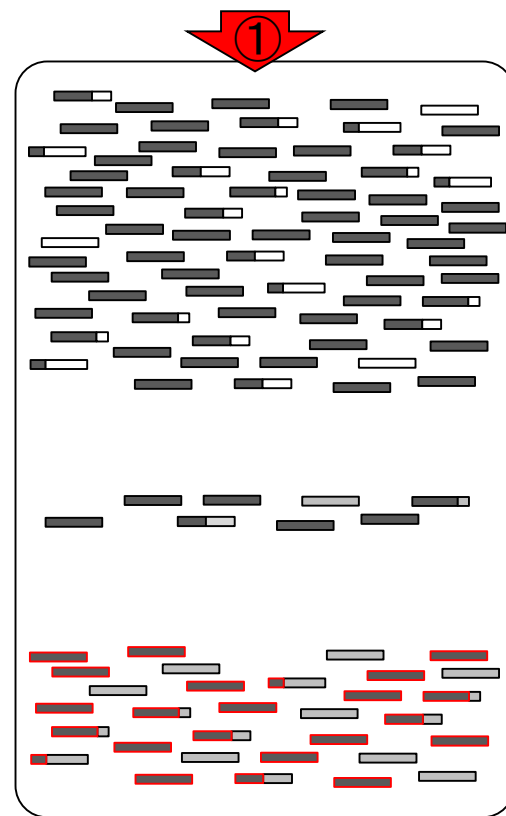
真の発現情報



RNA-seqで得られるリード情報 (色は不明)

# データ解析の出発点

トランスクリプトーム (RNA-seq) データ解析の出発点は、①RNA-seqデータファイル、



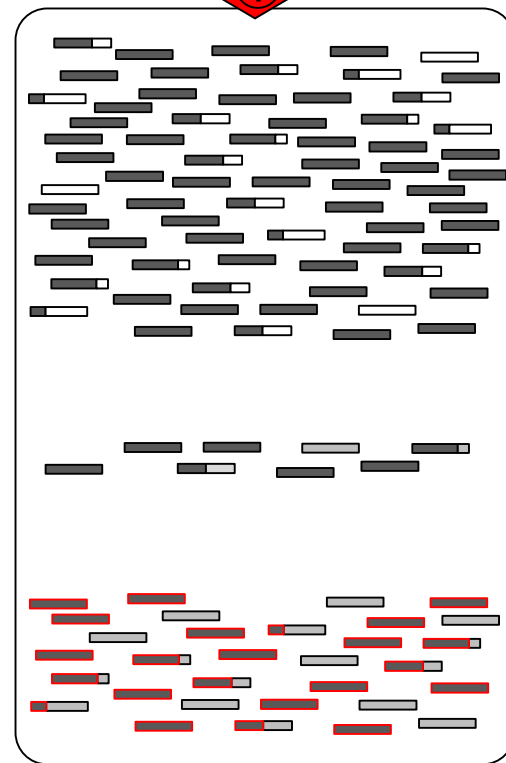
RNA-seqデータ

# データ解析の出発点

トランスクリプトーム (RNA-seq) データ解析の出発点は、①RNA-seqデータファイル、②ゲノム配列情報、

②

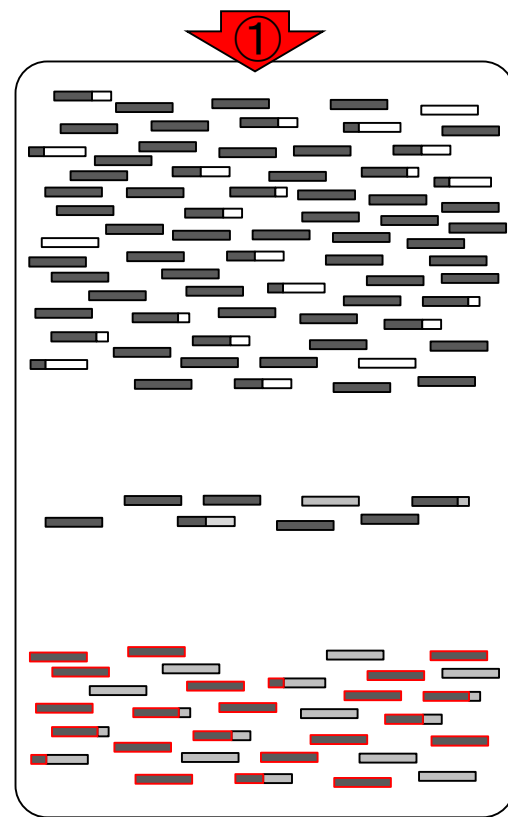
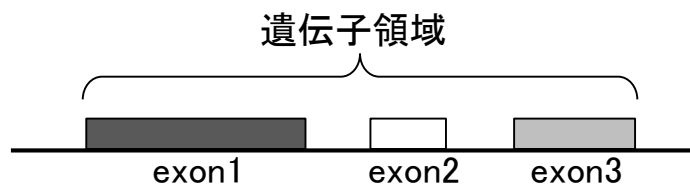
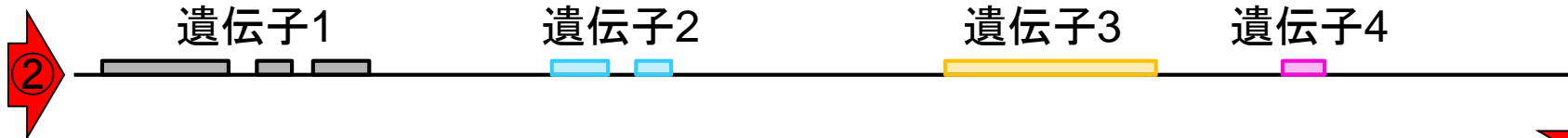
①



RNA-seqデータ

# データ解析の出発点

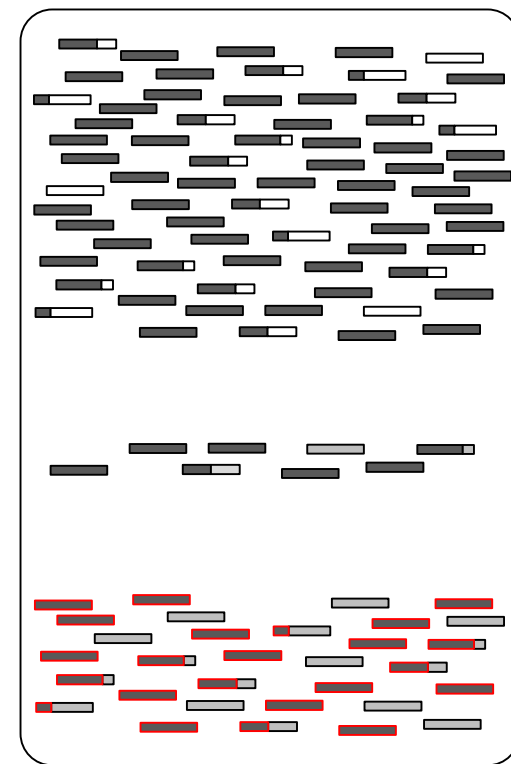
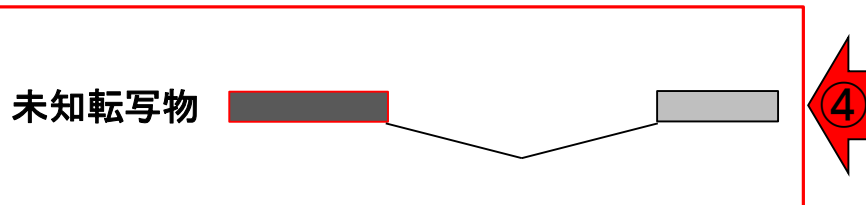
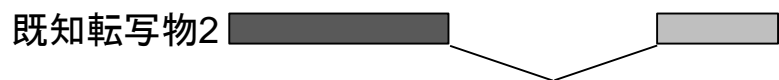
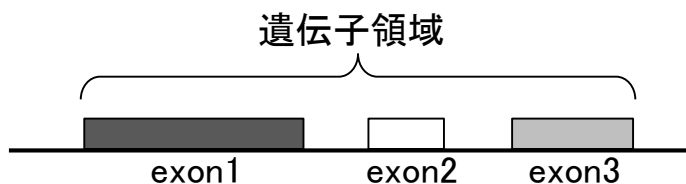
トランスクリプトーム (RNA-seq) データ解析の出発点は、①RNA-seqデータファイル、②ゲノム配列情報、③ゲノム上のどこにどんな遺伝子、exon、転写物が存在するかというアノテーション情報



RNA-seqデータ

# 解析結果のイメージ

①RNA-seqデータ、②ゲノム配列情報、③アノテーション情報を利用して、④未知転写物(新規isoform)の同定ができる。

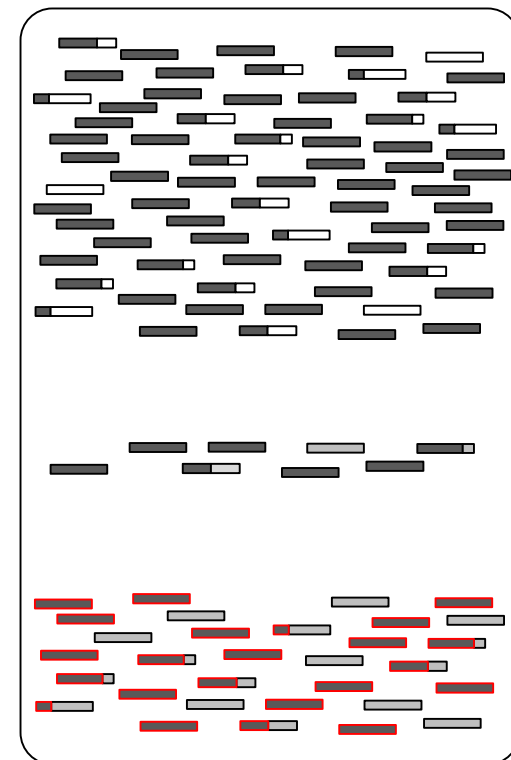
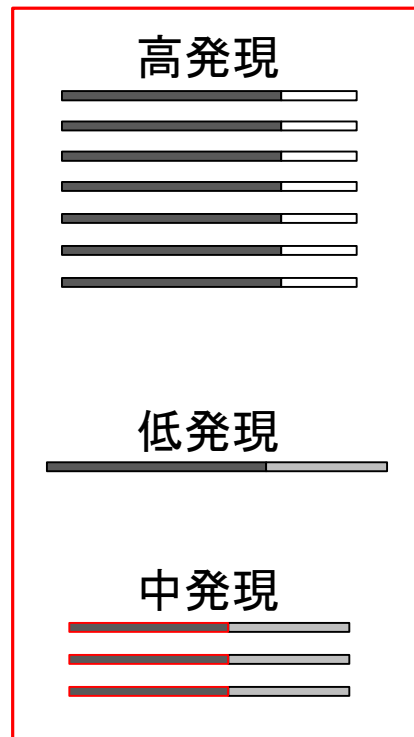
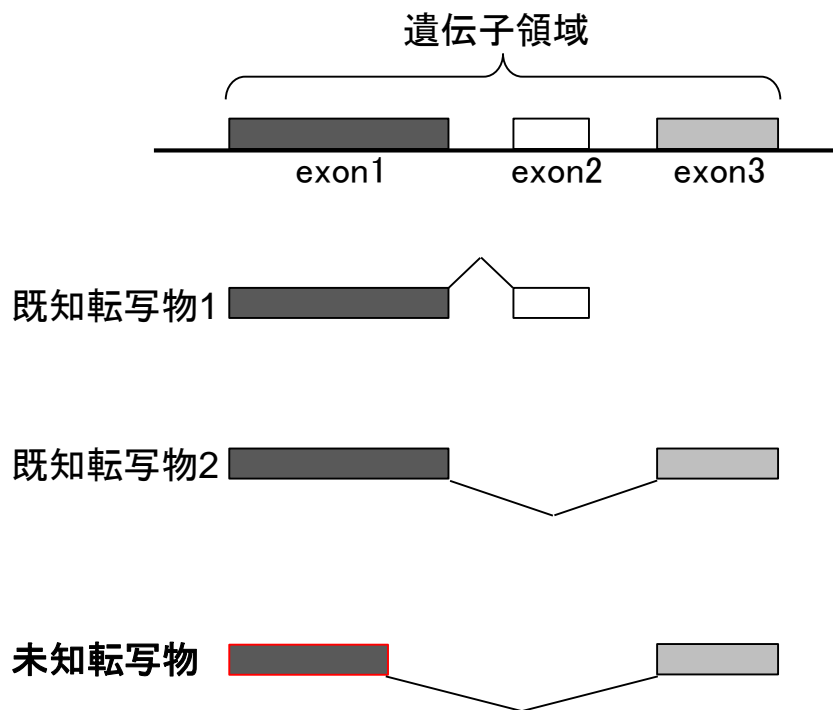


RNA-seqデータ



# 解析結果のイメージ

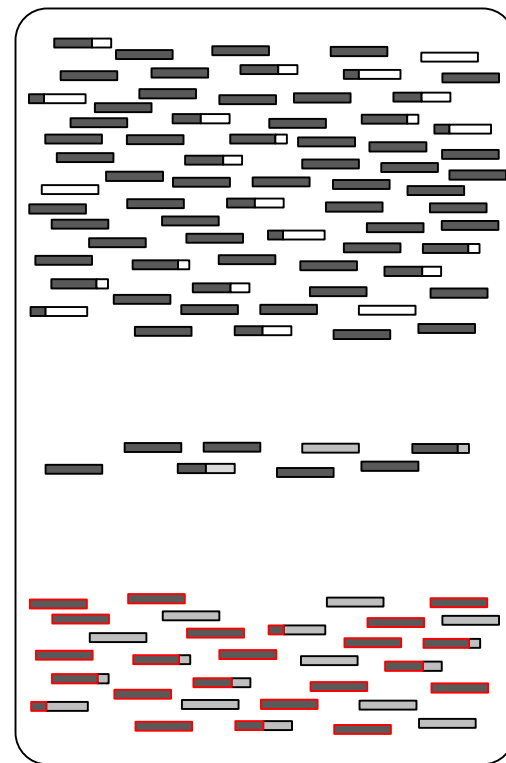
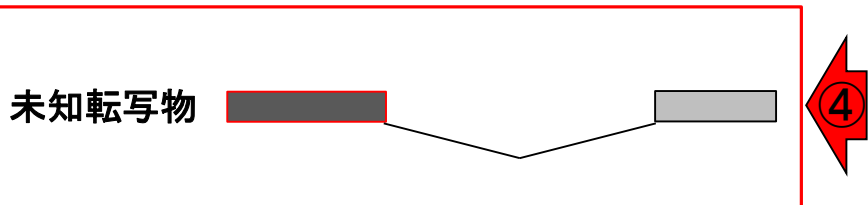
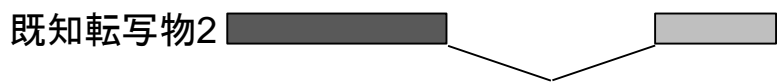
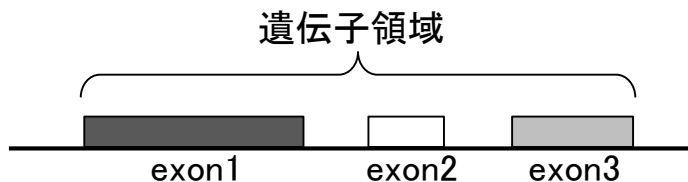
①RNA-seqデータ、②ゲノム配列情報、③アノテーション情報を利用して、④未知転写物(新規isoform)の同定ができる。⑤転写物の発現量(働いている度合い)推定も原理的に可能。



RNA-seqデータ

# 具体的な戦略は？

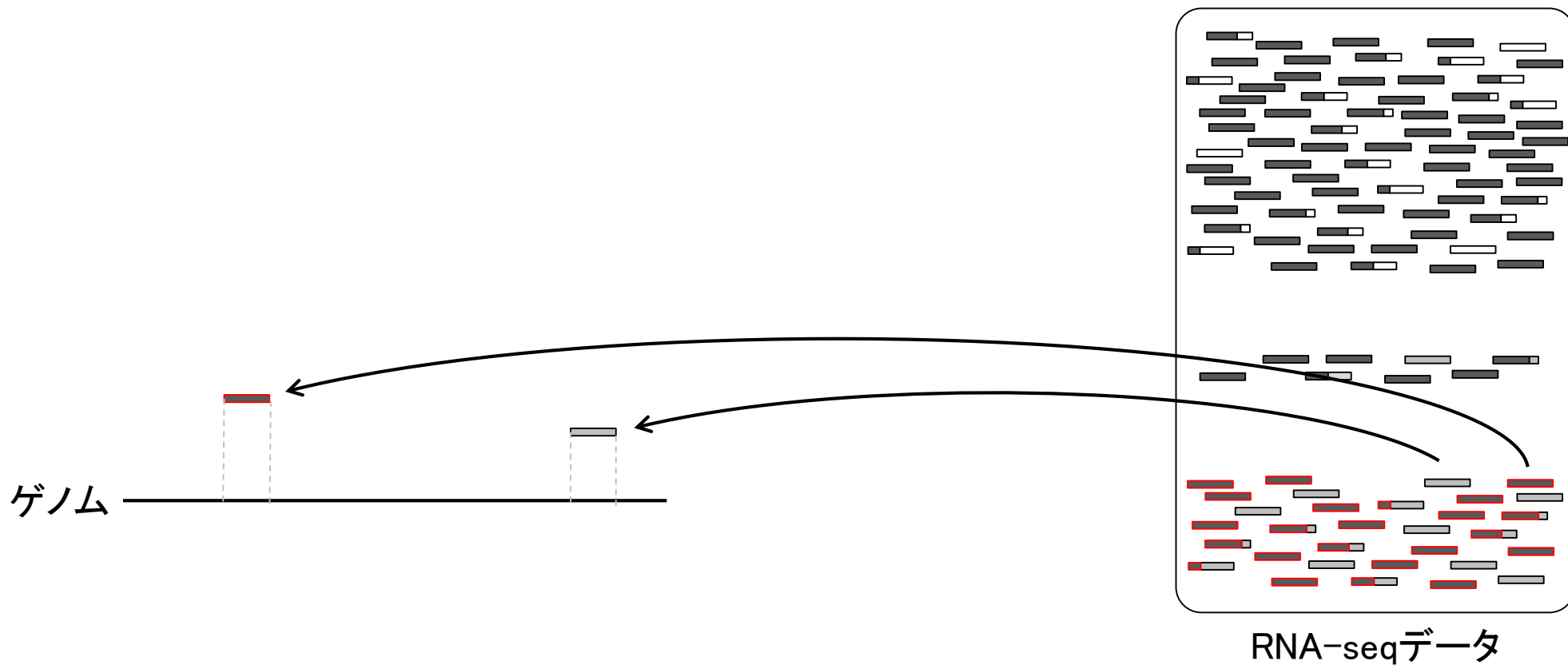
①RNA-seqデータ、②ゲノム配列情報、③アノテーション情報を利用して、④未知転写物(新規isoform)の同定ができる。



RNA-seqデータ

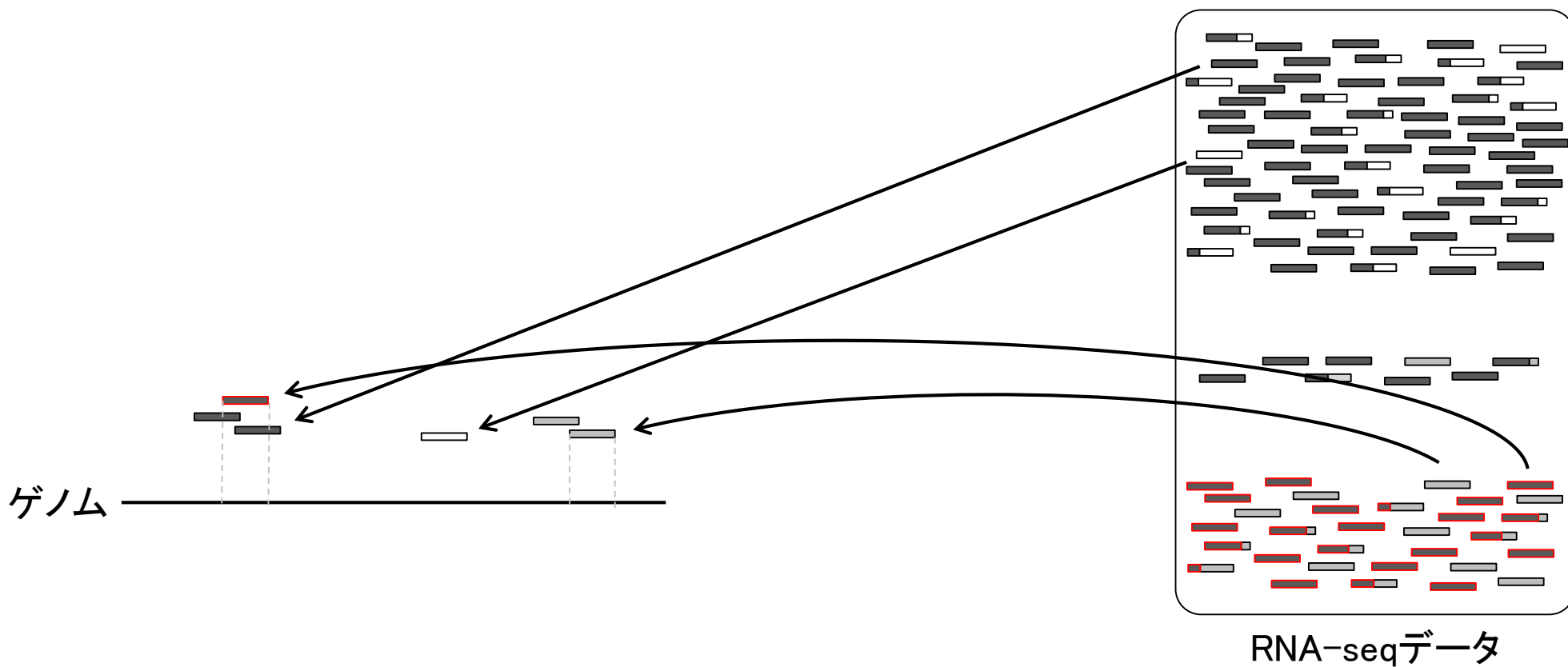
# 具体的な戦略

RNA-seqデータ中の1本1本のリード(横棒)がゲノム上のどの領域から転写されたのかを調べる。文字列検索と本質的に同じであり、これがマッピングという作業に相当する。



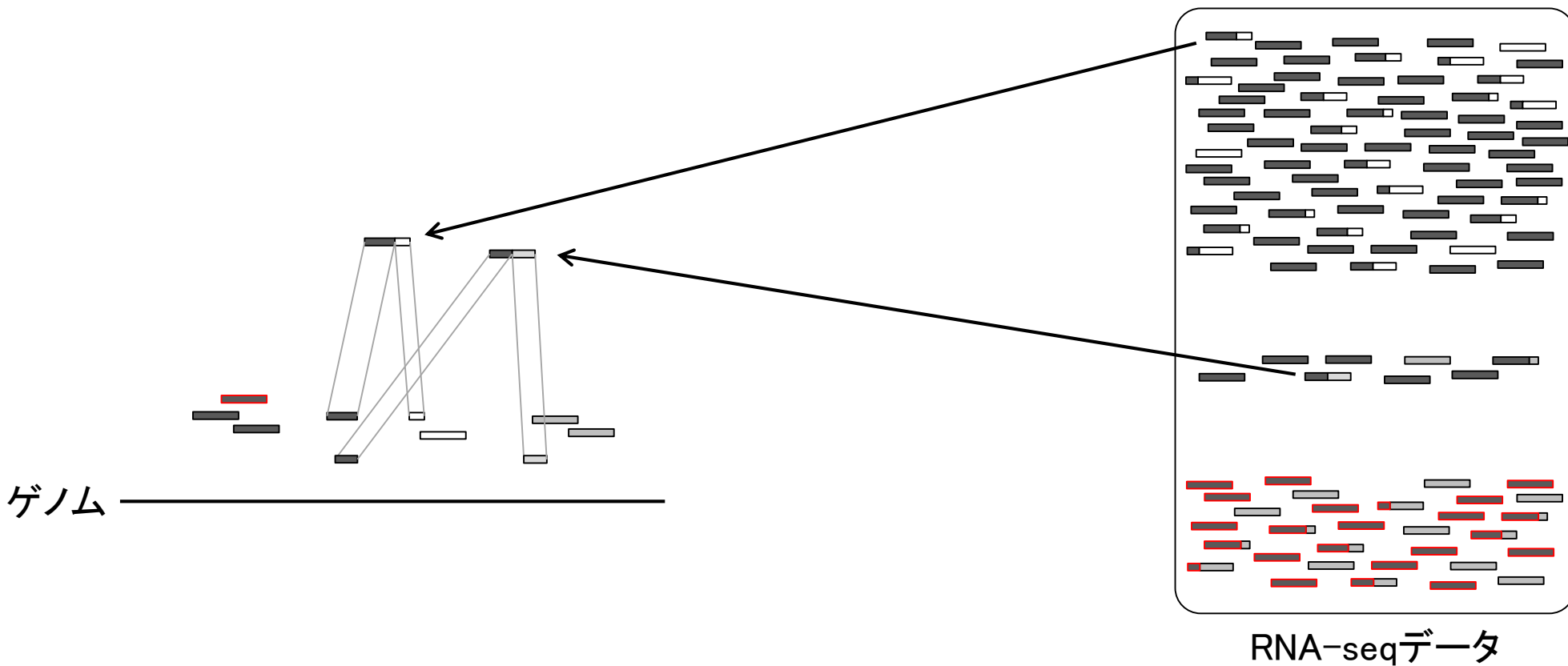
# 具体的な戦略

RNA-seqデータ中の1本1本のリード(横棒)がゲノム上のどの領域から転写されたのかを調べる。文字列検索と本質的に同じであり、これがマッピングという作業に相当する。



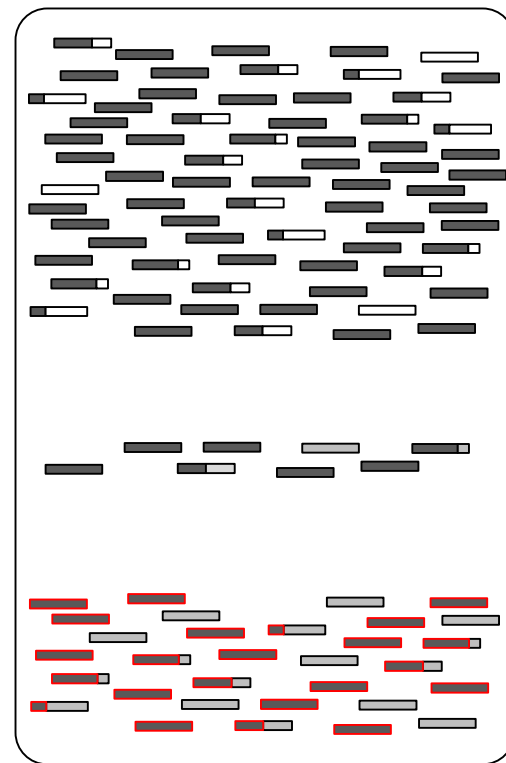
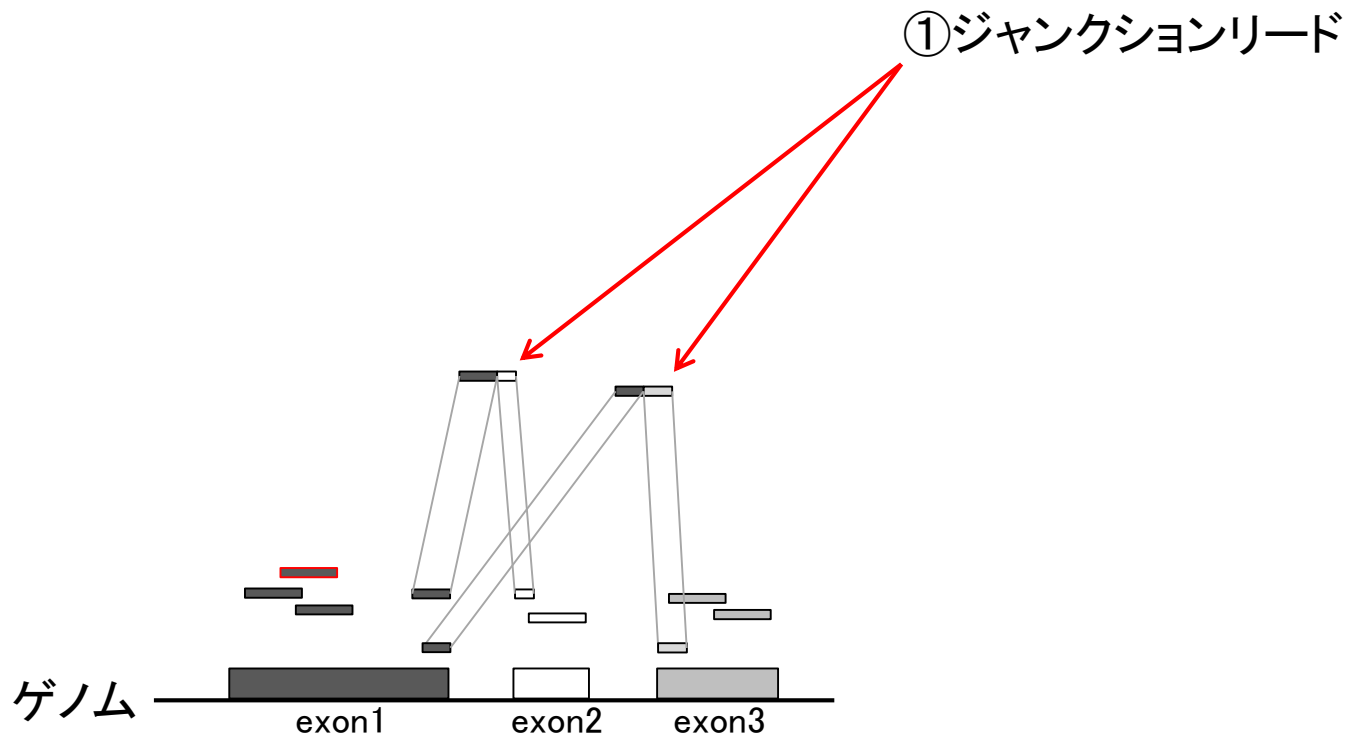
# 具体的な戦略

リードの長さが初期は35塩基程度だったが、現在は150塩基程度まで伸びている。そのおかげで、リードを分割してマッピングすることもできる。



# 具体的な戦略

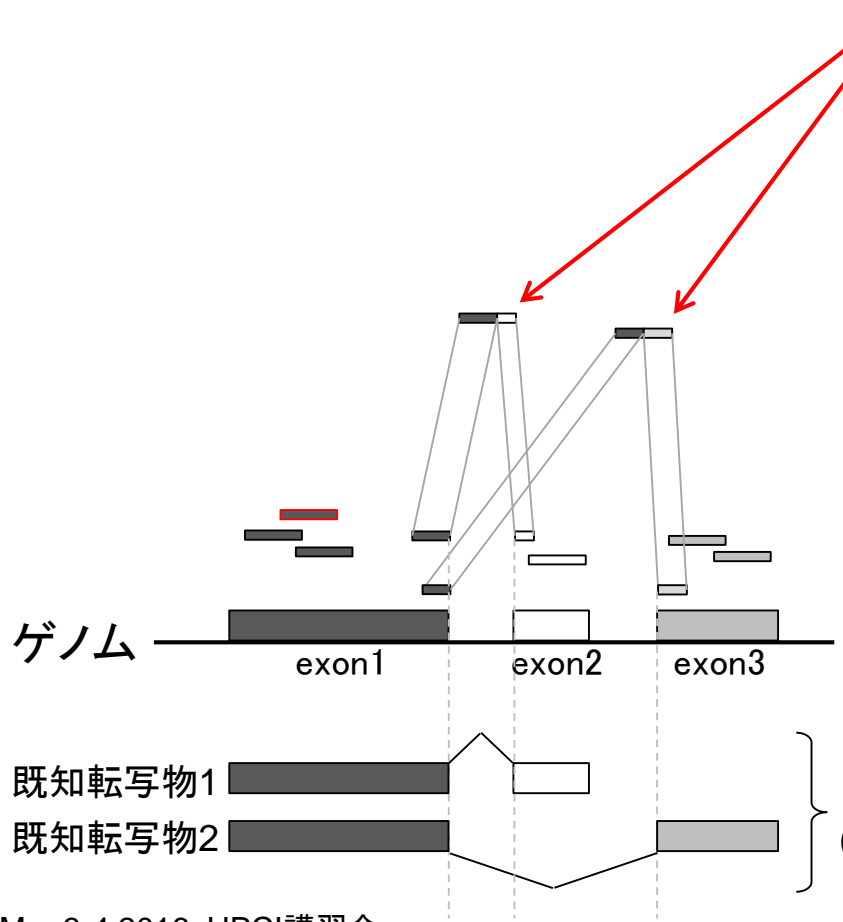
分割してマップされたリードは、大抵の場合複数のエクソン(exon)をまたぐリードであり、①ジャンクションリード(junction read)と呼ばれる。



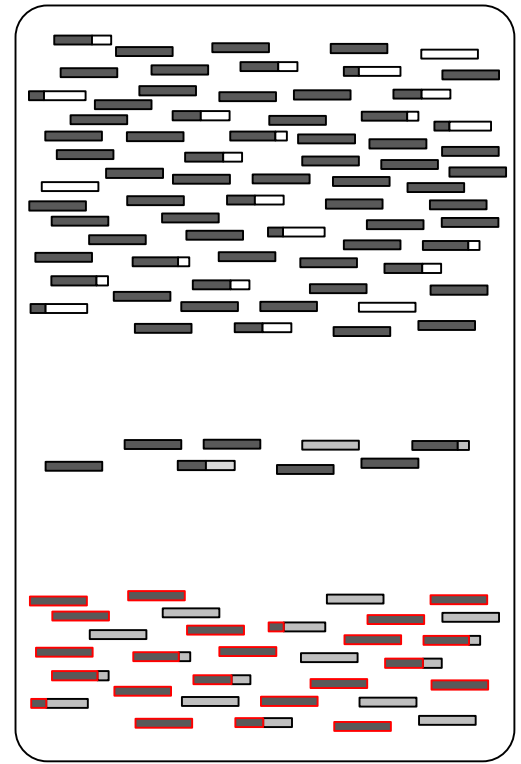
RNA-seqデータ

# 具体的な戦略

①ジャンクションリード



アノテーション情報  
(既知遺伝子座標情報)

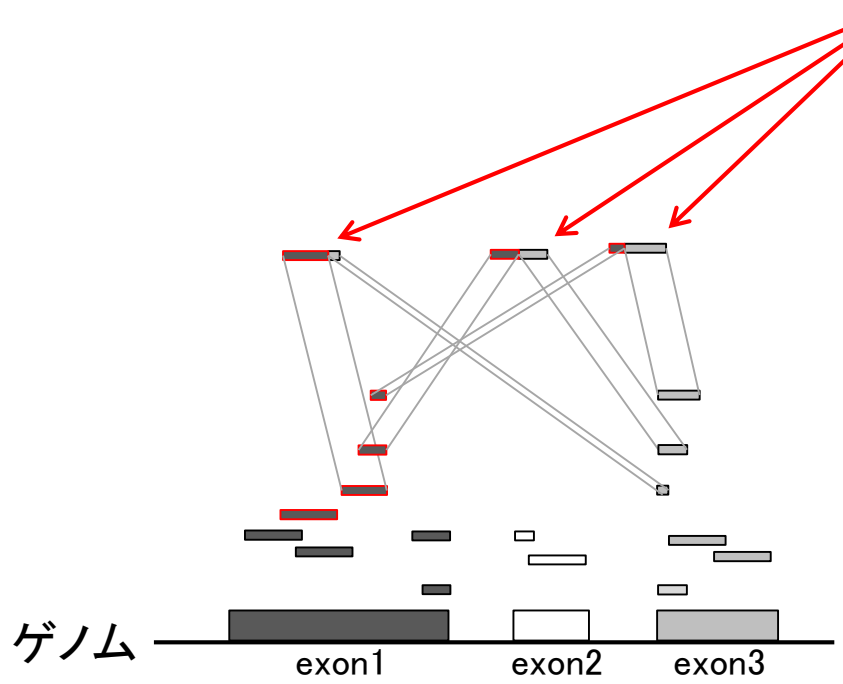


RNA-seqデータ

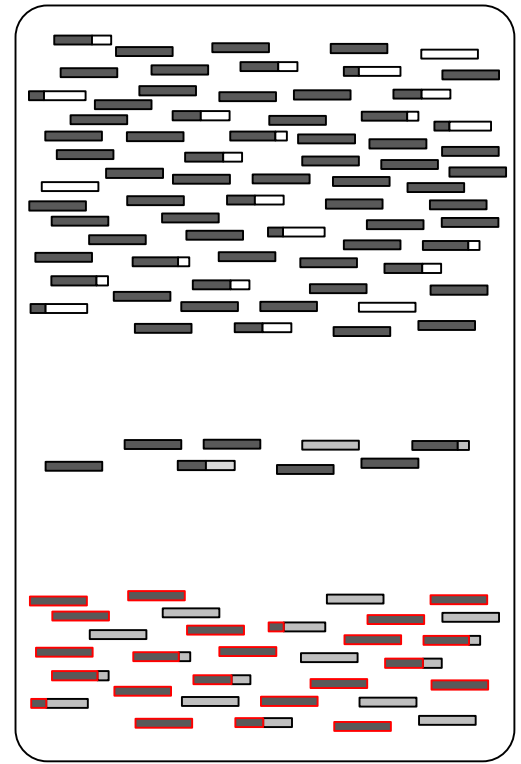
同様にして、他のジャンクションリードも既知転写物と比較することで…

# 具体的な戦略

①ジャンクションリード



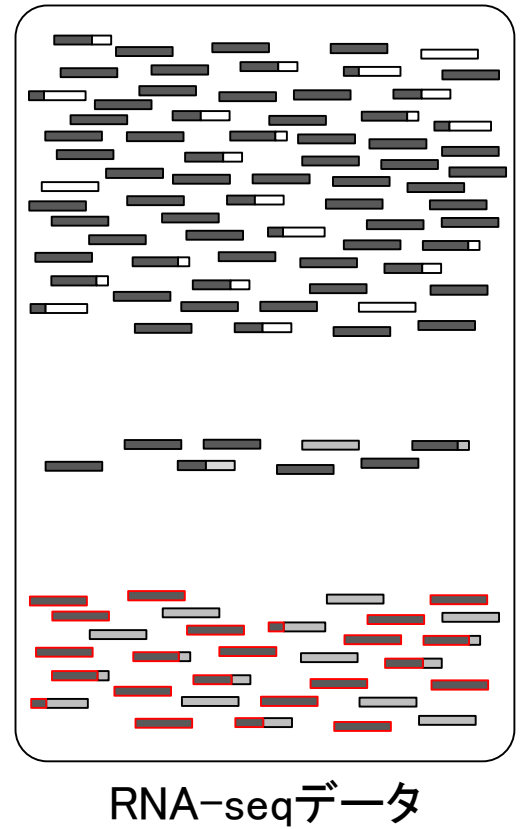
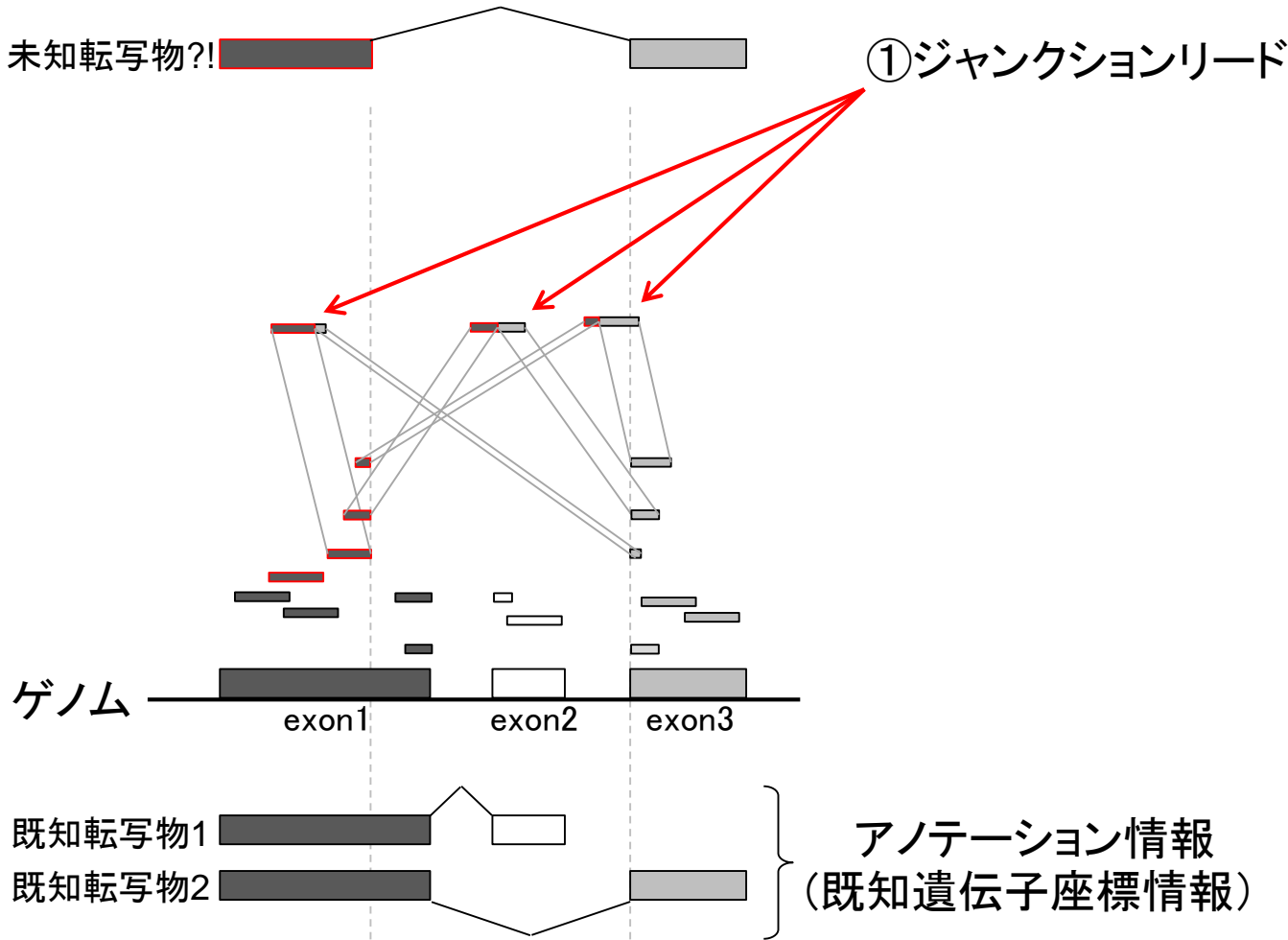
アノテーション情報  
(既知遺伝子座標情報)



RNA-seqデータ



# 具体的な戦略



# Contents2

## ■ トランスクリプトーム解析

- インTRODクシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)



# 様々な解析目的

## ■ トランスクリプトーム配列取得

### □ ゲノム配列未知の場合

- [アセンブル | について](#) (last modified 2014/06/20)
- [アセンブル | ゲノム用](#) (last modified 2015/08/2)
- [アセンブル | トランスクリプトーム\(転写物\)用](#) **①** (last modified 2015/08/18)
- [マッピング | について](#) (last modified 2015/11/1)
- [マッピング | basic aligner](#) (last modified 2014/08/08)
- [マッピング | splice-aware aligner](#) (last modified 2015/11/11)
- [マッピング | Bisulfite sequencing用](#) (last modified 2014/07/09)
- [マッピング | \(ESTレベルの長さの\)contig](#) (last modified 2014/06/24)
- [マッピング | 基礎](#) (last modified 2013/06/19)

トランスクリプトーム配列の *de novo* アセンブリに相当。多くのプログラムは発現量(FPKM値)も出力してくれます。

### アセンブル | トランスクリプトーム(転写物)用

Rパッケージはおそらくありません。

プログラム:

- [Multiple-k](#): Surget-Groba and Montoya-Burgos, *Genome Res.*, 2010
- [Trans-ABYSS](#): Robertson et al., *Nat Methods*, 2010
- [Rnnotator](#): Martin et al., *BMC Genomics*, 2010
- [Trinity](#): Grabherr et al., *Nat Biotechnol.*, 2011
- [Oases](#): Schulz et al., *Bioinformatics*, 2012
- [EBARDenovo](#): Chu et al., *Bioinformatics*, 2013
- [BRANCH](#): Bao et al., *Bioinformatics*, 2013
- [IDBA-tran](#): Peng et al., *Bioinformatics*, 2013
- [SOAPdenovo-Trans](#): Xie et al., *Bioinformatics*, 2014
- [VTBuilder](#): Archer et al., *BMC Bioinformatics*, 2014
- [Rockhopper 2\(バクテリア用\)](#): Tjaden B, *Genome Biol.*, 2015
- [DETONATE\(RSEM-EVAL\)](#): Li et al., *Genome Biol.*, 2014
- [Bridger](#): Chang et al., *Genome Biol.*, 2015
- [IFRAT](#): Mbandi et al., *BMC Bioinformatics*, 2015

Review、ガイドライン、パイプライン系:

- [Review](#): Martin and Wang, *Nat Rev Genet.*, 2011

# 様々な解析目的

## 発現量の正確な推定

- 解析 | 基礎 | 平均-分散プロット | [Biological replicates](#) (last modified 2014/02/25)
- 解析 | [新規転写物同定\(ゲノム配列を利用\)](#) (last modified 2015/08/25)
- 解析 | [発現量推定\(トランスクリプトーム配列を利用\)](#) (last modified 2015/11/10)
- 解析 | [クラスタリング](#) | [クラスタリングについて](#) (last modified 2014/02/25)

### 解析 | 発現量推定(トランスクリプトーム配列を利用)

新規転写物(新規isoform)の発見などが目的でなく、既知転写物の発現量を知りたいだけの場合には、やたらと時間がかかるゲノム配列へのマッピングを避けるのが一般的です。有名なCufflinksも一応GTF形式のアノテーションファイルを与えることでゲノム全体にマップするのを避けるモードがあるらしいので、一応リストアップしています。転写物へのマッピングの場合には、splice-aware alignerを用いたジャンクションリードのマッピングを行う必要がないので、高速にマッピング可能なbasic alignerで十分です。但し、複数個所にマップされるリードは考慮する必要があり、確率モデルのパラメータを最尤法に基づいて推定するexpectation-maximization (EM)アルゴリズムがよく用いられます。マッピングを行わずに、k-merを用いてalignment-freeで行う発現量推定を行うSailfishやRNA-Skimは従来法に比べて劇的に高速化がなされているようです。間違いがいくつか含まれているとは思いますが、2015年11月に調べた結果をリストアップします:

#### プログラム:

- [Cufflinks](#): Trapnell et al., Nat Biotechnol., 2010
- [NEUMA](#): Lee et al., Nucleic Acids Res., 2011
- [IsoEM](#): Nicolae et al., Algorithms Mol. Biol., 2011
- [RSEM](#): Li and Dewey, BMC Bioinformatics, 2011
- [eXpress](#): Roberts and Pachter, Nat Methods, 2013
- [ReXpress](#): Roberts et al., Bioinformatics, 2013
- [TIGAR](#): Nariai et al., Bioinformatics, 2013
- [eXpress-D](#): Roberts et al., BMC Bioinformatics, 2013
- [PennSeq](#): Hu et al., Nucleic Acids Res., 2014
- [Sailfish](#): Patro et al., Nat Biotechnol., 2014
- [RNA-Skim](#): Zhang and Wang, Bioinformatics, 2014
- [TIGER2](#): Nariai et al., BMC Genomics, 2014
- [EMSAR](#): Lee et al., BMC Bioinformatics, 2015
- [NLDMseq](#): Liu et al., BMC Bioinformatics, 2015

転写物の発現量を正確に推定したい場合は、専用のプログラムを使うべし。②RSEMが有名。③Sailfishも高速なアルゴリズムとして有名。④TIGER2は日本語で質問できる上、最近の手法比較論文(Kanitz et al. *Genome Biol.*, 2015)でも高評価でおススメ。

# 様々な解析目的

## 発現変動解析(2群間比較)

- [解析 | フィルタリング | について](#) (last modified 2015/11/10)
- [解析 | 発現変動 | について](#) (last modified 2014/07/10)
- [解析 | 発現変動 | 2群間 | 対応なし | について](#) (last modified 2015/11/13)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DESeq2\(Love 2014\)](#) (last modified 2015/11/15)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC\(Sun 2013\)](#) (last modified 2015/07/07) 推奨 **①**
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | Blekhanmanデータ | TCC\(Sun 2013\)](#) (last modified 2015/07/07)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq\(Li 2013\)](#) (last modified 2014/02/07)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR\(Robinson 2010\)](#) (last modified 2014/07/24)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | WAD\(Kadota 2008\)](#) (last modified 2015/03/30)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC\(Sun 2013\)](#) (last modified 2014/03/05) 推奨 **②**
- [解析 | 発現変動 | 2群間 | 対応なし | 複製なし | DESeq\(Anders 2010\)](#) (last modified 2014/03/20)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製なし | edgeR\(Robinson 2010\)](#) (last modified 2014/03/20)

2群間比較で①反復あり(複製あり)データの場合はedgeR、②反復なしの場合はDESeq2を内部的に用いて頑健な結果を返すTCCがおススメ。反復の有無に応じて、内部的に用いるパッケージを自動で切り替える

3群間比較で①反復あり(複製あり)データの場合は edgeR、②反復なしの場合はDESeq2を内部的に用いて頑健な結果を返すTCCがおススメ (Tang et al., BMC Bioinformatics, 2015)。反復の有無に応じて、内部的に用いるパッケージを自動で切り替える。

# 様々な解析目的

## 発現変動解析(3群間比較)

- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [について](#) (last modified 2015/11/05)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [DESeq2\(Love 2014\)](#) (last modified 2015/02/04)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [TCC\(Sun 2013\)](#) (last modified 2015/11/05)推奨
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [EBSeq\(Leng 2013\)](#) (last modified 2015/02/10)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [SAMseq\(Li 2013\)](#) (last modified 2015/02/10)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [edgeR\(Robinson 2010\)](#) (last modified 2015/02/03)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [基礎](#) | [DESeq\(Anders 2010\)](#) (last modified 2014/03/13)
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製あり](#) | [応用](#) | [TCC\(Sun 2013\)](#) (last modified 2015/11/05)推奨
- [解析](#) | [発現変動](#) | [3群間](#) | [対応なし](#) | [複製なし](#) | [TCC\(Sun 2013\)](#) (last modified 2015/11/05)推奨



*Lactobacillus casei* 12A株のデータ。乳酸菌NGS連載第3回最後のほうでダウンロードしたものと基本的に同じ。トリムの理由は第5回でわかる。

# 解析データ(乳酸菌)

## ■ マップする側 (paired-end RNA-seqデータ; SRR616268)

### □ オリジナルデータ (Illumina HiSeq 2000で取得) の情報

- リード長: forward側は107 bp、reverse側は93 bp
- リード数: とともに134,755,996リード(約1.35億)
- データ量: bzip2圧縮状態で計約15GB。非圧縮FASTQで計約80GB



### □ 下記手順実行後のデータ(計約110MB)をマッピングに利用

1. 最初の100万リードのみ抽出(計200万リード)
2. forward側: 3'側7 bpをトリム後にアダプターを除去。998,649リード
3. reverse側: 3'側2 bpをトリム後にアダプターを除去。999,233リード
4. 両方で存在するリードのみ抽出。998,521 × 2 = 計1,997,042リード



### □ forward側([SRR616268sub\\_trim3\\_1.fastq.gz](#))、reverse側([SRR616268sub\\_trim3\\_2.fastq.gz](#))

## ■ マップされる側 (*Lactobacillus casei* 12A)

### □ ゲノムサイズ: 2,907,892 bp、遺伝子数: 2,799個

- [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)
- [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#)



FASTA形式がわかるヒトは、それにquality情報のみが追加されたものという理解でよい。

# FASTA形式とFASTQ形式

## ■ FASTA形式

- 1行目：“>”ではじまる一行のdescription行
- 2行目：配列情報

```
>SEQ_ID  
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
```

## □ FASTQ形式

- 1行目：“@”ではじまる1行のdescription行
- 2行目：配列情報
- 3行目：“+”からはじまる1行（のdescription行）
- 4行目：クオリティ情報

```
@SEQ_ID  
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT  
+  
!' '* ((( (**+)) %%%++) (%%%) .1***-+*'') **55CCF>>>>>CCCCCCC65
```

[http://en.wikipedia.org/wiki/FASTQ\\_format](http://en.wikipedia.org/wiki/FASTQ_format)



# 解析データ(乳酸菌)

マップされる側の①ゲノム配列データと②アノテーションデータ(GFFファイル)情報は、Ensemblから取得。バージョンは年に数回程度以上の頻度で上がっている印象。基本は最新だが、最も重要なのはゲノム配列とアノテーション情報のバージョンが同じであること

http://bacteria.ensembl.org/Lactobacillus\_casei\_12a/Info/Index

EnsemblBacteria | BLAST | Tools | Downloads | Documentation | Website help

Lactobacillus casei 12A (ASM30956v2)

## Lactobacillus casei 12A

Lactobacillus casei 12A

Provider [European Nucleotide Archive](#) | Taxonomy ID [1051650](#)

Search *Lactobacillus casei* 12A...

e.g. *spaB* or Chromosome:502057-502915 or *synthetase*

### About *Lactobacillus casei* 12A

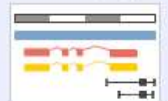
Information and statistics

Genome assembly: [GCA\\_000309565.2](#)

- More information and statistics
- Download DNA sequence (FASTA)
- Convert your data to GCA\_000309565.2 coordinates
- Display your data in Ensembl Bacteria



View karyotype



Example region

### Gene annotation

What can I find? Protein-coding and non-coding genes, splice variants, cDNA and protein sequences, non-coding RNAs.

- More about this genebuild
- Download genes, cDNAs, ncRNA, proteins - FASTA - GFF3
- Update your old Ensembl IDs



Example gene



Example transcript



### Comparative genomics

What can I find? Gene families based on HAMAP and PANTHER classification.

More about comparative analyses

### Variation

This species currently has no variation database. However you can process your own variants using the Variant Effect Predictor:

Variant Effect Predictor

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨン: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# マッピング基礎

QuasRは、**general**なパッケージ。精度云々というよりは、WindowsのR環境でマッピングを可能にしてくれたという点で感謝m(\_ \_)m。前処理、マッピング、QCレポート、カウント情報取得まで可能なので、このあたりの全貌を大まかに理解するうえで便利。

[Bioinformatics](#). 2015 Apr 1;31(7):1130-2. doi: 10.1093/bioinformatics/btu781. Epub 2014 Nov 21.

## QuasR: quantification and annotation of short reads in R.

[Gaidatzis D<sup>1</sup>](#), [Lerch A<sup>1</sup>](#), [Hahne F<sup>2</sup>](#), [Stadler MB<sup>1</sup>](#).

### + Author information

### Abstract

QuasR is a package for the integrated analysis of high-throughput sequencing data in R, covering all steps from read preprocessing, alignment and quality control to quantification. QuasR supports different experiment types (including RNA-seq, CHIP-seq and Bis-seq) and analysis variants (e.g. paired-end, stranded, spliced and allele-specific), and is integrated in Bioconductor so that its output can be directly processed for statistical analysis and visualization.

**AVAILABILITY AND IMPLEMENTATION:** QuasR is implemented in R and C/C++. Source code and binaries for major platforms (Linux, OS X and MS Windows) are available from Bioconductor ([www.bioconductor.org/packages/release/bioc/html/QuasR.html](http://www.bioconductor.org/packages/release/bioc/html/QuasR.html)). The package includes a 'vignette' with step-by-step examples for typical work flows.

**SUPPLEMENTARY INFORMATION:** Supplementary data are available at Bioinformatics online.

© The Author 2014. Published by Oxford University Press.

PMID: 25417205 [PubMed - indexed for MEDLINE] PMCID: PMC4382904 [Free PMC Article](#)

# マッピング基礎

QuasRは、basicなマッピングプログラムであるbowtie (Langmead et al., *Genome Biol.*, 2009)とジャンクションリードをマッピング可能なSpliceMap (Au et al., *Nucleic Acids Res.*, 2010)を選択可能。ここではバクテリア(乳酸菌)ゲノムにマップするので高速な①basic alignerを利用。②例題2

- マッピング | 基礎 (last modified 2013/06/19)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2014/06/19)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis 2015) (last modified 2015/06/19)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis 2015) (last modified 2014/06/19)
- マッピング | paired-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2016/06/19)
- マッピング | paired-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis 2015) (last modified 2016/06/19)
- マッピング | paired-end | トランスクリプトーム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2016/06/19)
- マッピング | paired-end | トランスクリプトーム | splice-aware aligner | QuasR(Gaidatzis 2015) (last modified 2016/06/19)

## マッピング | paired-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis\_2015) NEW

QuasRパッケージを用いてpaired-end RNA-seqデータのリファレンスゲノム配列へのマッピングを行うやり方を示します。basic alignerの1つであるBowtie (Langmead et al., *Genome Biol.*, 2009)を実装した Rbowtieパッケージを内部的に使っています。mapping paired genome1.txtのような2行目の1列目と2列目に「マッピングしたいRNA-seqファイル名1と2」(例: sample\_RNAseq\_1.faとsample\_RNAseq\_2.fa)、そして2行目の3列目に「任意のサンプル名」(例: namae)を記載したタブ区切りテキストファイルを用意した上で行います。1行目の文字列は変えてはいけません(つまり"FileName1", "FileName2",および"SampleName"のままにしておくということです)

### 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

1. mapping paired genome2.txt  
乳酸菌R  
107 bpで  
ている L  
(Lactoba  
シオンは

1. の入力ファイルから5'および3'側を rcode\_20150707\_preprocessing.txt に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092,219 bytes)と SRR616268sub\_trim3\_2.fastq.gz (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されている Lactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus casei 12a.GCA\_000309565.2.25.dna.chromosome.Chromosome.fa)がリファレンス配列です。マッピングオプションはデフォルトです。

in\_f1  
in\_f2

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納(リファレンスゲノム配列ファイル)

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
```

# マッピング基礎

①マップされる側のリファレンス配列。②マップする側はリストファイル(タブ区切りテキストファイル)として与える。リストファイルの中身は、③paired-endの場合はこんな感じ。

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする

1.の入力ファイルから5'および3'側を [rcode\\_20150707\\_preprocessing.txt](#) に書いてある手順でダウンロードして得たファイルは998,521ワードからなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と [SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#) から提供されている [Lactobacillus casei 12A](#) の multi-FASTA形式ゲノム配列ファイル([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)) がリファレンス配列です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #入力ファイル名を指定してin_f2に格納(リファレンス配列)

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
out #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out) #マッピング結果(alignment statistics)の表示。seqlength:

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1]) #Quqlity Controlレポートのpdfファイル名を作成
qQCReport(out, pdfFilename=out_f) #QCレポート結果をファイルに保存
out_f
```

FileName1	FileName2	SampleName
SRR616268sub_trim3_1.fastq.gz	SRR616268sub_trim3_2.fastq.gz	naemae_paired

# マッピング基礎

リストファイルとして与えるメリットは、複数サンプルの場合を考えればよい。①のファイルは2行からなるが、3行目以降に同様な形式で、追加するサンプル数分だけ行を増やしていけばよい。②SampleName列は、カウントデータ(後述)を得るときに、ここに記載されたサンプル名(ここでは`nae_paired`)が列名として使われる。

## 2. `mapping paired genome2.txt`中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする

1.の入力ファイルから5'および3'側を `rcode 20150707 preprocessing.txt` に書いてある手順で得られるpaired-endのファイルです。 `SRR616268sub_trim3_1.fastq.gz` (59,092,219 bytes)と `SRR616268sub_trim3_2.fastq.gz` (54,667,920 bytes)です。 `Ensembl (Flicek et al., 2014)` から提供されている `Lactobacillus casei` ゲノム配列ファイル(`Lactobacillus casei 12a.GCA_000309565.2.25.dna.chromosome.Chromosome1.dna`)をマッピングする。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に代入
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome1.dna"

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
out #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out) #マッピング結果(alignment statistics)の表示。seqlength:

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1]) #Quqlity Controlレポートのpdfファイル名を作成
qQCReport(out, pdfFilename=out_f) #QCレポート結果をファイルに保存
out_f
```



FileName1	FileName2	SampleName
SRR616268sub_trim3_1.fastq.gz	SRR616268sub_trim3_2.fastq.gz	nae_paired





# 解析データ(乳酸菌)

## ■ マップする側 (paired-end RNA-seqデータ; SRR616268)

### □ オリジナルデータ (Illumina HiSeq 2000で取得) の情報

- リード長: forward側は107 bp、reverse側は93 bp
- リード数: とともに134,755,996リード(約1.35億)
- データ量: bzip2圧縮状態で計約15GB。非圧縮FASTQで計約80GB

### □ 下記手順実行後のデータ(計約110MB)をマッピングに利用

1. 最初の100万リードのみ抽出(計200万リード)
2. forward側: 3'側7 bpをトリム後にアダプターを除去。998,649リード
3. reverse側: 3'側2 bpをトリム後にアダプターを除去。999,233リード
4. 両方で存在するリードのみ抽出。998,521 × 2 = 計1,997,042リード

### □ forward側(SRR616268sub\_trim3\_1.fastq.gz)、reverse側(SRR616268sub\_trim3\_2.fastq.gz)

FileName1	FileName2	SampleName
SRR616268sub_trim3_1.fastq.gz	SRR616268sub_trim3_2.fastq.gz	naeae_paired

- [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)
- [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#)

# マッピング基礎

①「デスクトップ - hoge - L.casei\_12A\_genome」フォルダ内に必要なファイルはあります。②作業ディレクトリをそこに変更して、③list.files()。

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合

1.の入力ファイルから5'および3'側をrcode 20からなるpaired-endのファイルです。SRR616268 (54,667,920 bytes)です。Ensembl (Flicek et al)のゲノム配列ファイル(Lactobacillus casei 12a.GCA000309565.2.25.chromosome.Chromosome.gff3)をマッピングオプションはデフォルトです。

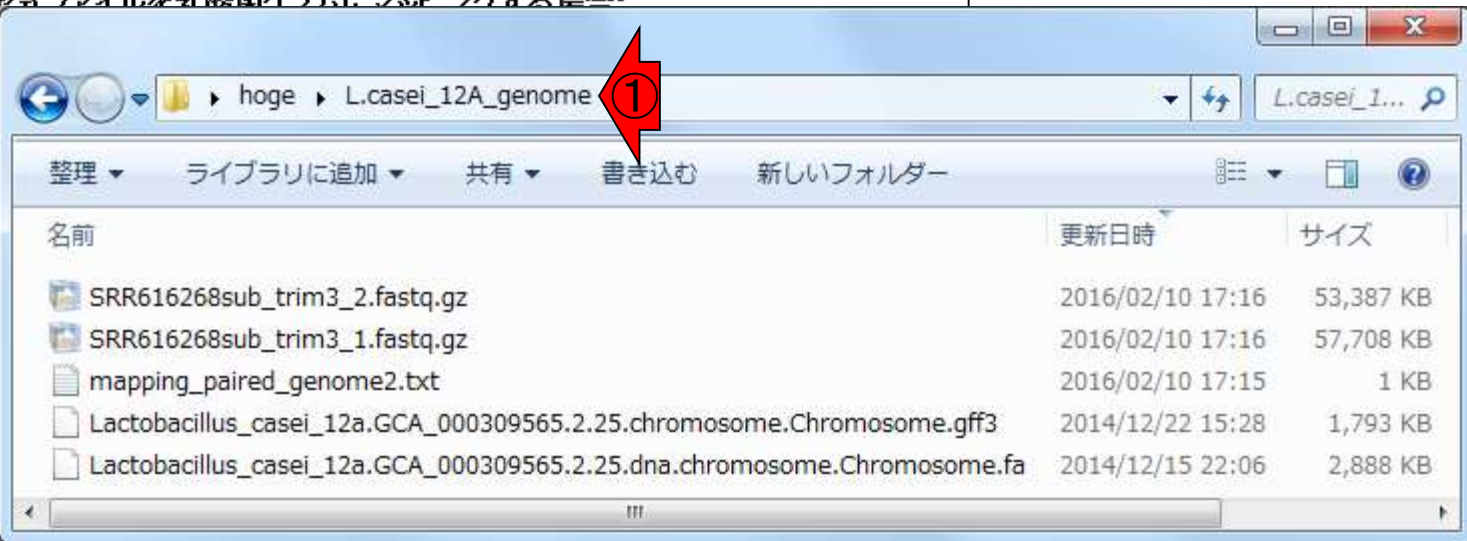
```
in_f1 <- "mapping_paired_genome2.txt"
in_f2 <- "Lactobacillus_casei_12a.fastq.gz"

#必要なパッケージをロード
library(QuasR)
library(GenomicAlignments)

#本番(マッピング)
time_s <- proc.time()
out <- qAlign(in_f1, in_f2)
time_e <- proc.time()
time_e - time_s
out
alignmentStats(out)

#ファイルに保存(QCレポート用のpdfファイル)
out_f <- sub(".bam", "_QC.pdf", out@alignments)
qQCReport(out, pdfFilename=out_f)
out_f

#ファイルに保存(BED形式ファイル)
tmpfname <- out@alignments[,1]
```



#計算時間を計測するため

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_genome"
> list.files()
[1] "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"
[2] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"
[3] "mapping_paired_genome2.txt"
[4] "SRR616268sub_trim3_1.fastq.gz"
[5] "SRR616268sub_trim3_2.fastq.gz"
> ls()
character(0)
> |
```

# マッピング基礎

①赤枠部分が必要最小限。②QuasRパッケージを読み込んで、③主要関数であるqAlignを実行するだけで、BAM形式のマッピング結果ファイルを得ることができる。①の赤枠部分をコピペ。「このプログラムの機能のいくつかがWindows ファイアウォールでブロックされています」というアラートウィンドウが出ることもあるが、④その場合はキャンセルボタンを押す(でないと先に進めない)。約4分

2. [mapping paired genome2.txt](#)中のFASTQ形式ファイルを乳酸菌ク

1.の入力ファイルから5'および3'側を [rcode 20150707 preprocessing.t](#) からなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#)から提供されてい ム配列ファイル([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chrom](#))のマッピングオプションはデフォルトです。

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納(ゲノム配列ファイル)

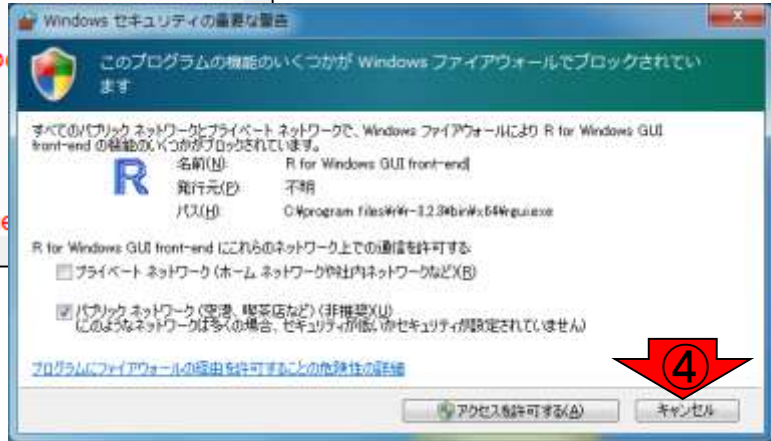
#必要なパッケージをロード
library(QuasR) #パッケージの読み込み
library(GenomicAlignments) #パッケージの読み込み

#本番(マッピング)
time_s <- proc.time()
out <- qAlign(in_f1, in_f2) #計算時間を計測するため
time_e <- proc.time() #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e - time_s #計算時間を計測するため
out #計算時間を表示(一番右側の数字。単位はsecond)
alignmentStats(out) #マッピングに用いたパラメータや入力ファイルの情報などを表示
#マッピング結果(alignment statistics)の表示。seqlength:

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quqlity Controlレポートのpdfファイル名を生成
qQCReport(out, pdfFilename=out_f) #QCレポート結果をファイルに保存
out_f #ファイル名を表示してるだけです

#ファイルに保存(BED形式ファイル)
tmpfname <- out@alignments[,1] #ファイル名(in_f1の1列目に相当)をtmpfname

```



# 解説

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムに

1.の入力ファイルから5'および3'側をrcode 20150707 preprocessing.txtに書  
 かなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092  
 (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されているLa  
 ム配列ファイル(Lactobacillus casei 12a.GCA\_000309565.2.25.dna.chromos  
 マッピングオプションはデフォルトです。

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測する
out <- qAlign(in_f1, in_f2) #マッピングを行うqA
time_e <- proc.time() #計算時間を計測する
time_e - time_s #計算時間を表示(一
out #マッピングに用いた
alignmentStats(out) #マッピング結果(al

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#QuasR
qQCReport(out, pdfFilename=out_f) #QCレポート結果をフ
out_f #ファイル名を表示し

#ファイルに保存(BED形式ファイル)
tmpfname <- out@alignments[,1] #ファイル名(in_f1)
    
```



R Console画面上で見えているのは、①のあたり  
 。②マッピング部分の所要時間は204.96秒。計  
 1,997,042リード中、③693,500個がマップされ、④  
 1,303,542個がマップされなかったことがわかる。

```

> time_e - time_s #計算$
  ユーザ      システム      経過
    0.73      0.61      204.96 ②

> out #マッ$
Project: qProject
Options  : maxHits      : 1
          paired       : fr
          splicedAlignment: FALSE
          bisulfite     : no
          snpFile       : none

Aligner  : Rbowtie v1.10.0 (parameters: -m 1$
Genome   : .../Lactobacillus_casei_12a.GCA_0$

Reads    : 1 pair of files, 1 sample (fastq $
1. SRR6...stq.gz SRR6...stq.gz  namee_pair$

Genome alignments: directory: same as reads
1. SRR616268sub_trim3_1_1be82f4864d3.bam

Aux. alignments: none

> alignmentStats(out) #マッ$
          seqlength mapped unmapped
namee_paired:genome 2907892 693500 1303542
> |
    
```



# 解析データ(乳酸菌)

## ■ マップする側 (paired-end RNA-seq)

- オリジナルデータ (Illumina HiSeq 2000)
  - リード長: forward側は107 bp、reverse側は101 bp
  - リード数: とともに134,755,996リード
  - データ量: bzip2圧縮状態で計約150GB
- 下記手順実行後のデータ (計約110GB)
  1. 最初の100万リードのみ抽出 (計200万リード)
  2. forward側: 3'側7 bpをトリム後にトリム
  3. reverse側: 3'側2 bpをトリム後にトリム
  4. 両方で存在するリードのみ抽出。

□ forward側(SRR616268sub\_trim3\_1.fastq.gz)

## ■ マップされる側 (*Lactobacillus casei*)

- ゲノムサイズ: 2,907,892 bp、遺伝子数: 2,200
- *Lactobacillus\_casei*\_12a.GCA\_000304900.1
- *Lactobacillus\_casei*\_12a.GCA\_000304900.1

```

R Console
> time_e - time_s                                     #計算$
  ユーザ   システム   経過
      0.73      0.61    204.96
> out                                               #マップ$
Project: qProject
Options  : maxHits      : 1
          paired       : fr
          splicedAlignment: FALSE
          bisulfite     : no
          snpFile       : none
Aligner  : Rbowtie v1.10.0 (parameters: -m 1$
Genome   : ../Lactobacillus_casei_12a.GCA_0$

Reads    : 1 pair of files, 1 sample (fastq $
1. SRR6...stq.gz SRR6...stq.gz  nae_pair$

Genome alignments: directory: same as reads
1. SRR616268sub_trim3_1_1be82f4864d3.bam

Aux. alignments: none

> alignmentStats(out)                               #マップ$
                                     seqlength mapped unmapped
nae_paired:genome 2907892 693500 1303542
> |
    
```

# 解説

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムに

1.の入力ファイルから5'および3'側を [rcode\\_20150707\\_preprocessing.txt](#) に書  
 かれるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1\\_fastq.gz](#) (59,092  
 (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#)から提供されている [La](#)  
[ム配列ファイル\(Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromos](#)  
 マッピングオプションはデフォルトです。

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測する
out <- qAlign(in_f1, in_f2) #マッピングを行うqA
time_e <- proc.time() #計算時間を計測する
time_e - time_s #計算時間を表示(一
out #マッピングに用いた
alignmentStats(out) #マッピング結果(al

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#QuasR
qQCReport(out, pdfFilename=out_f) #QCレポート結果をフ
out_f #ファイル名を表示し

#ファイルに保存(BED形式ファイル)
tmpfname <- out@alignments[,1] #ファイル名(in_f1)
    
```



①qAlign関数実行時に、マッピングプログラム (bowtie or SpliceMap)の指定を行わなかった。理由はデフォルトがbasic aligner (unspliced alignerともいう)のbowtieであることを知っていたから。② qAlign関数上では、splicedAlignment = FALSEとして表現される。主なマッピング結果であるBAMファイルは、③拡張子が.bamだが、これはバイナリファイルなので中身を眺めても意味不明(爆)

```

> t
> c
Proc
Options      : maxHits      : 1
              paired      : fr
              splicedAlignment: FALSE
              bisulfite     : no
              snpFile       : none

Aligner      : Rbowtie v1.10.0 (parameters: -m 1$
Genome       : .../Lactobacillus_casei_12a.GCA_0$

Reads        : 1 pair of files, 1 sample (fastq $
              1. SRR6...stq.gz SRR6...stq.gz  namea_pair$

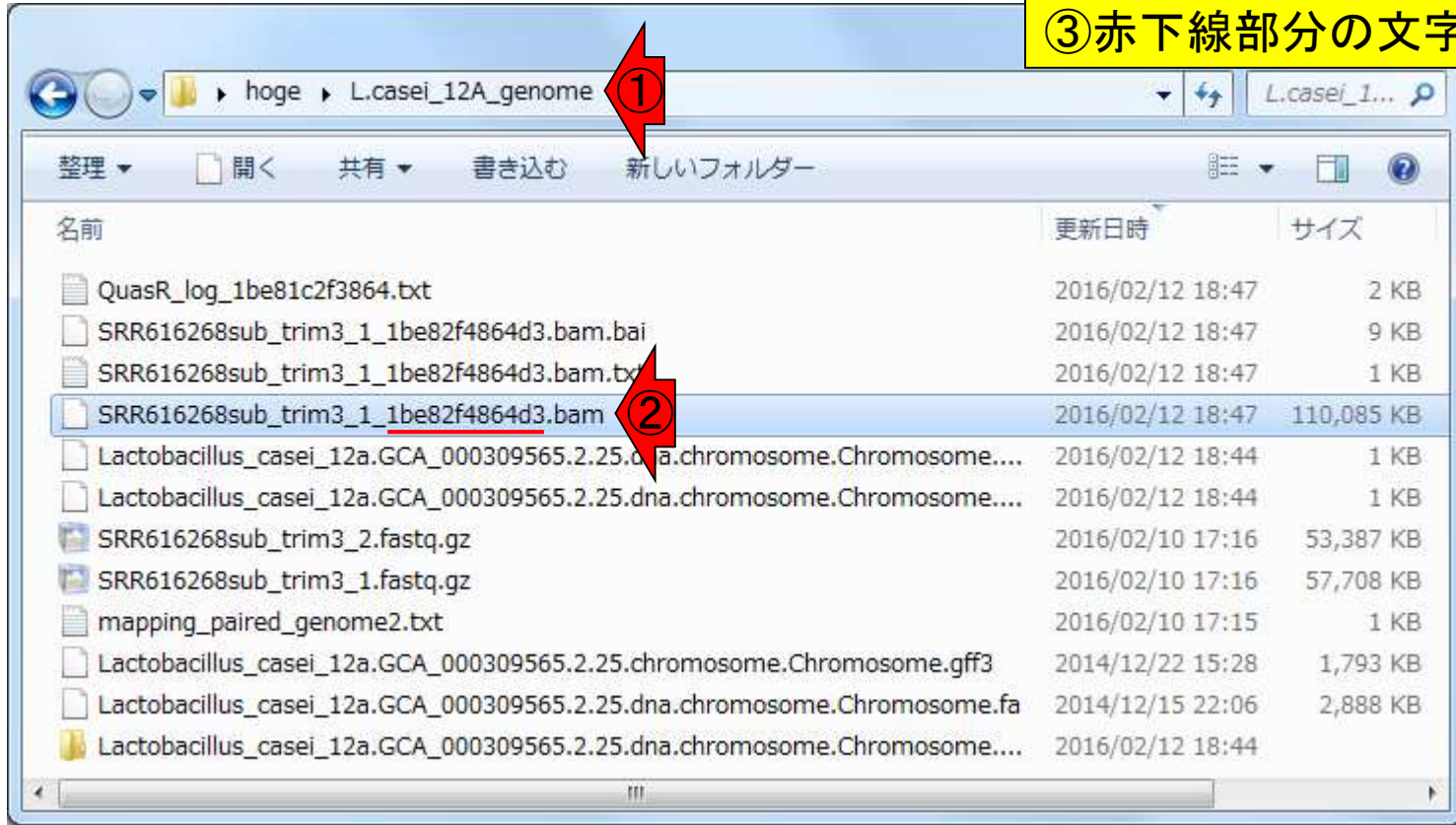
Genome alignments: directory: same as reads
              1. SRR616268sub_trim3_1_1be82f4864d3.bam

Aux. alignments: none

> alignmentStats(out) #マッ$
              seqlength mapped unmapped
namea_paired:genome 2907892 693500 1303542
> |
    
```

# 解説

①作業フォルダを眺めると、②確かに.bamファイルが作成されている。マッピング後の解析は、基本的にBAM形式ファイルを入力として取り扱う。  
③赤下線部分の文字列はヒトそれぞれ。



```
#計算$
204.96
#マツ$
: 1
: fr
gnment: FALSE
: no
: none
.10.0 (parameters: -m 1$
acillus_casei_12a.GCA_0$
files, 1 sample (fastq $
R6...stq.gz  nae_pair$
ectory: same as reads
3_1_1be82f4864d3.bam
```

```
Aux. alignments: none
> alignmentStats(out) #マツ$
                               seqlength mapped unmapped
nae_paired:genome 2907892 693500 1303542
> |
```



# QCレポートPDF

①入力データのクオリティスコアやマッピング結果の概要などをPDFファイルとして出力できます。最後の部分が\_QC.pdfとなるPDFファイルが作成されます。

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピング

1.の入力ファイルから5'および3'側を [rcode\\_20150707\\_preprocessing.txt](#) に書いてある手順でダウンロードして得たファイル(998,521ペア)からなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と [SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#) から提供されている [Lactobacillus casei 12A](#) の multi-FASTA形式ゲノム配列ファイル([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)) がリファレンス配列です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納(リファレンス配列)
```

#必要なパッケージをロード

```
library(QuasR) #パッケージの読み込み
library(GenomicAlignments) #パッケージの読み込み
```

#本番(マッピング)

```
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一秒未満は小数で表示)
out #マッピングに用いたデータ
alignmentStats(out) #マッピング結果(alignmentStats関数)
```

#ファイルに保存(QCレポート用のpdfファイル作成)

```
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#QuasRの出力名を_QC.pdfに変更
qQCReport(out, pdfFilename=out_f) #QCレポート結果をPDFファイルに出力
out_f #ファイル名を表示
```

#ファイルに保存(BED形式ファイル)

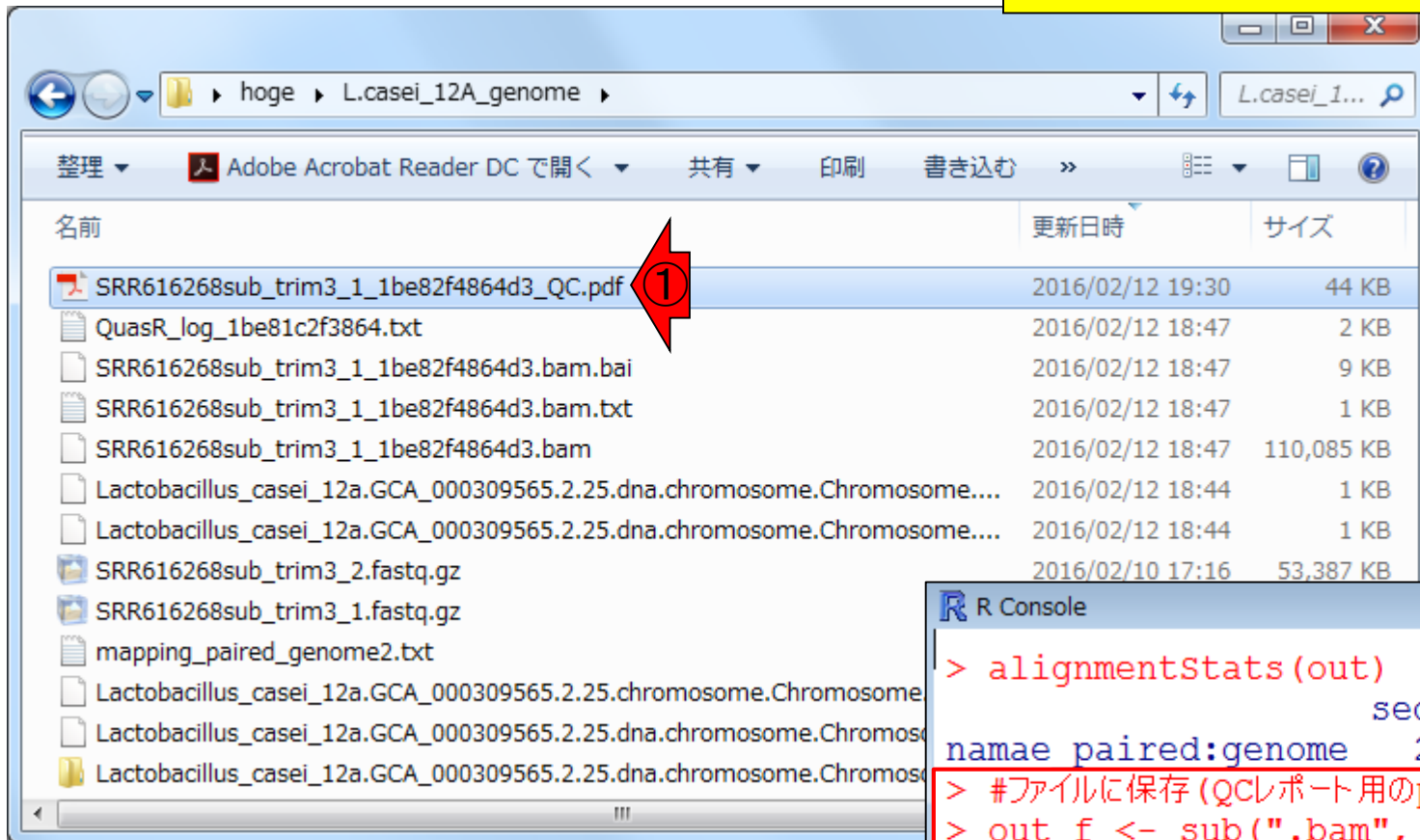
```
tmpfname <- out@alignments[,1] #ファイル名(in_f1のファイル名をtmpfnameに格納)
```

```
R Console
> alignmentStats(out) #マッ$
          seqlength mapped unmapped
name paired:genome 2907892 693500 1303542
> #ファイルに保存(QCレポート用のpdfファイル作$
> out_f <- sub(".bam", "_QC.pdf", out@alignmen$
> qQCReport(out, pdfFilename=out_f) #QCレ$
collecting quality control data
creating QC plots
> out_f #ファ$
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_$
>
```



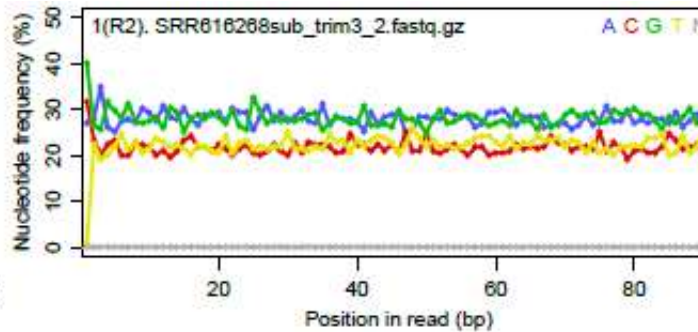
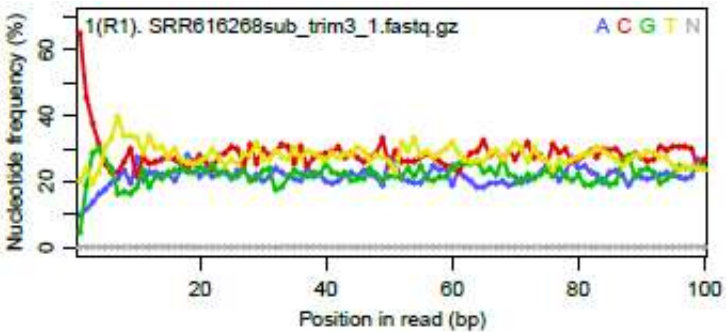
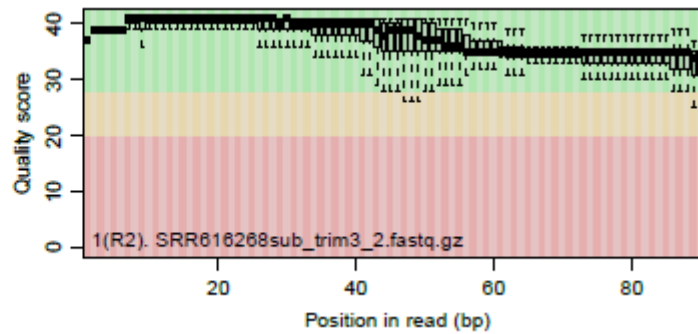
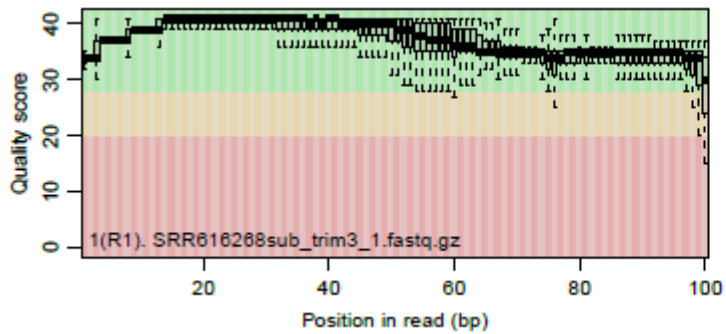
# QCLレポートPDF

①確かに最後の部分が\_QC.pdfとなるPDFファイルが作成されていることが分かります。これを解説します。



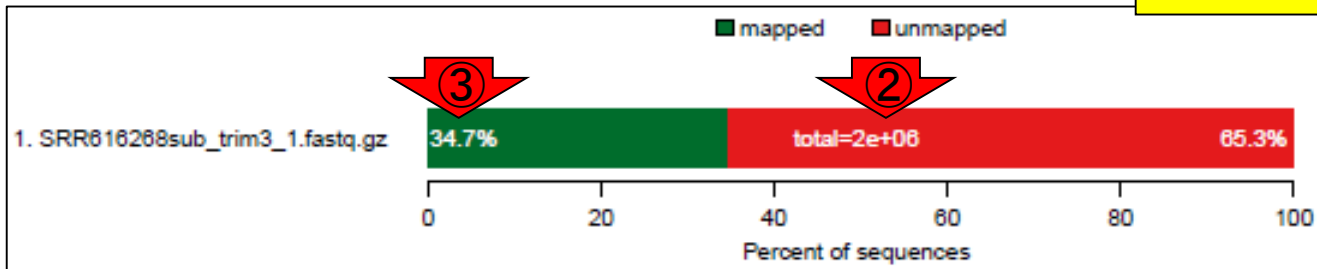
```
R Console
> alignmentStats(out) #マツ$
                               seqlength mapped unmapped
name paired:genome 2907892 693500 1303542
> #ファイルに保存(QCLレポート用のpdfファイル作$
> out_f <- sub(".bam", "_QC.pdf", out@alignmen$
> qQCReport(out, pdfFilename=out_f) #QCL$
collecting quality control data
creating QC plots
> out_f #ファ$
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_$.
> |
```

# PDFの一部を解説



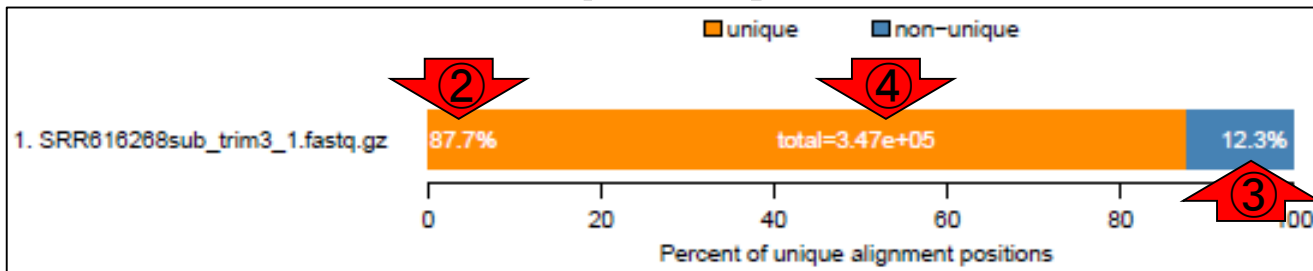
# PDFの一部を解説

① マップされたリードの割合。② 計1,997,042  
リード(2e+06)中、③ 34.7%がマップされた。④  
 $693,500 / 1,997,042 = 0.3473$ なので妥当。



```
R Console
> alignmentStats(out) #マップ$
      seqlength mapped unmapped
name_paired:genome 2907892 693500 1303542
> #ファイルに保存(QCレポート用のpdfファイル作$
> out_f <- sub(".bam", "_QC.pdf", out@alignmen$
> qQCReport(out, pdfFilename=out_f) #QCレ$
collecting quality control data
creating QC plots
> out_f #ファ$
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_$
> |
```

# PDFの一部を解説



① マップされたリードのうち、1か所にのみマップされたリード (uniquely mapped reads) が 87.7%、③ 複数個所にマップされたリード (non-unique) が 12.3%。④  $3.47e+05$  は片側のみで考えているのかもしれない…。いずれにせよ、⑤  $693500 \times 0.877 = 608200$  個程度はゲノム中の1か所にのみマップされたと解釈。

```
R Console
> alignmentStats(out) #マップ$
                        seqlength mapped unmapped
name_paired:genome    2907892 693500 1303542
> #ファイルに保存(QCレポート用のpdfファイル作$
> out_f <- sub(".bam", "_QC.pdf", out@alignmen$
> qQCReport(out, pdfFilename=out_f) #QCレ$
collecting quality control data
creating QC plots
> out_f #ファ$
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_$
> |
```

# BEDファイル作成

BED形式ファイルは、バイナリファイルのため中身を解釈できないBAMファイルを変換して可視化したもの、という理解でよい。①赤枠部分をコピー。②.bedというファイルが作成される。③forというループを回しているのは、複数サンプルに対応するため。

## 2. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピング

1の入力ファイルから5および3'側を [rcode\\_20150707\\_preprocessing.txt](#) に書いてある手順からなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#) から提供されている [Lactobacillus](#) ム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chrom](#)) をマッピングオプションはデフォルトです。

```

out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
out #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out) #マッピング結果(alignment statistics)の表示。seqlength:

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1]) #Quality Controlレポートのpdfファイル名を作成
qQCReport(out, pdfFilename=out_f) #QCレポート結果をファイルに保存
out_f #名前を表示してるだけです

```

```

#ファイルに保存(BED形式ファイル)
tmpfname <- out@alignments[,1] #ファイル名(in_f1)の
for(i in 1:length(tmpfname)){ #サンプル数(ファイル)
  hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込む
  hoge <- as.data.frame(hoge) #データフレーム形式に変換
  tmp <- hoge[, c("seqnames", "start", "end")] #必要な列の抽出
  out_f <- sub(".bam", ".bed", tmpfname[i]) #BED形式ファイル名を作成
  out_f #ファイル名を表示
  write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
}

```

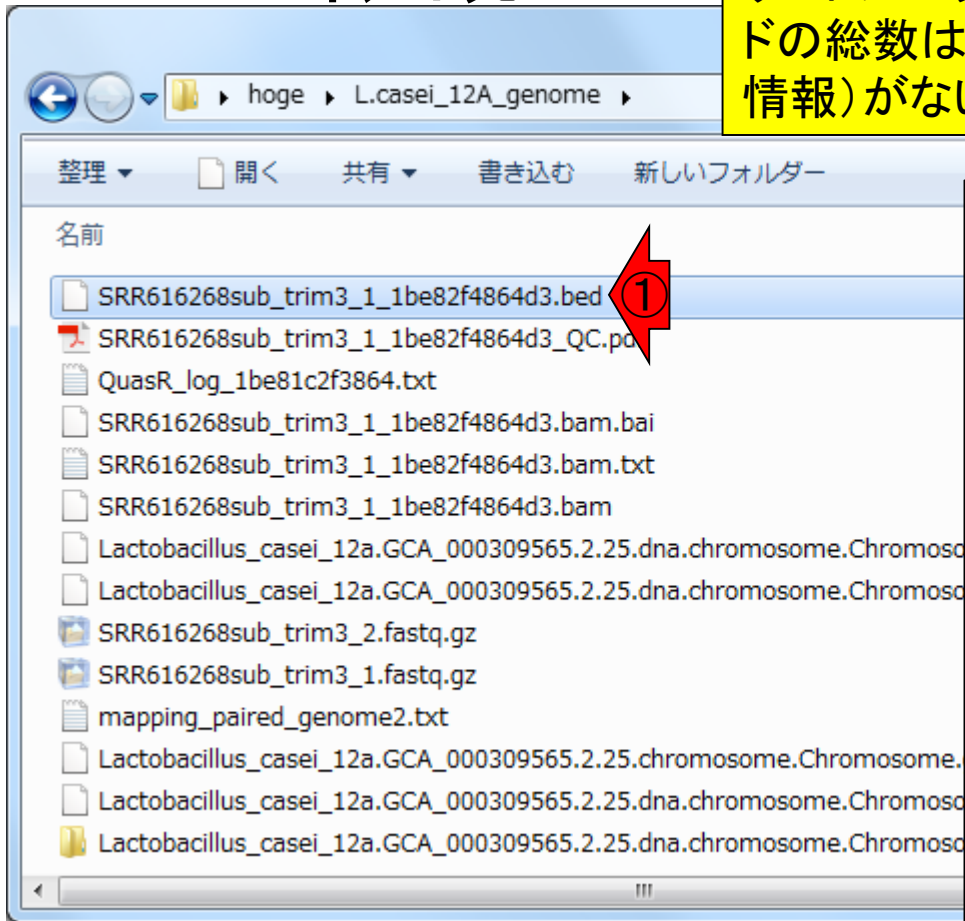
```

R Console
> #ファイルに保存 (BED形式ファイル)
> tmpfname <- out@alignments[,1] #ファイル名(in_f1)の
> for(i in 1:length(tmpfname)){ #サンプル数(ファイル)
+   hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込む
+   hoge <- as.data.frame(hoge) #データフレーム形式に変換
+   tmp <- hoge[, c("seqnames", "start", "end")] #必要な列の抽出
+   out_f <- sub(".bam", ".bed", tmpfname[i]) #BED形式ファイル名を作成
+   out_f #ファイル名を表示
+   write.table(tmp, out_f, sep="\t", append=F, row.names=F)
+ }
> |

```

# BED概観

①作成されたBEDファイルをエクセルで眺める。マッピング結果は、②どの配列上の、③どこから(start)、④どこまで(end)の場所にリードがマップされたかが最低限わかればよい。マップされたリードの総数は693,500個だったので、ヘッダー行(つまり1行目に列名情報)がないこのファイルの場合、⑤693,500行となるのは妥当。



	A	B	C
1	Chromosome	5	95
2	Chromosome	23	112
3	Chromosome	45	134
4	Chromosome	45	135
5	Chromosome	49	139
6	Chromosome	53	143
-	Chromosome	57	144
693496	Chromosome	2907787	2907886
693497	Chromosome	2907787	2907886
693498	Chromosome	2907787	2907886
693499	Chromosome	2907789	2907887
693500	Chromosome	2907789	2907887

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# マッピング応用

QuasRはオプション無指定でもマッピングを行ってくれるが、ログファイル(QuasR\_log\_1be81c2f3864.txt)を眺めても-vで指定する許容するミスマッチ数が何だったのか不明。②例題1。③bowtie (basic alignerの1つ)利用時に"-m 1 --best --strata -v 0"オプションをつけて実行する。

- マッピング | 基礎 (last modified 2013/06/19)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis 2015) (last modified 2014/06/21)
- マッピング | paired-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)
- マッピング | paired-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)
- マッピング | paired-end | トランスクリプトーム | basic aligner(基礎) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)
- マッピング | paired-end | トランスクリプトーム | basic aligner(応用) | QuasR(Gaidatzis 2015) (last modified 2016/02/11)

## マッピング | paired-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis\_2015) NEW

QuasRパッケージを用いてpaired-end RNA-seqデータのリファレンスゲノム配列へのマッピングを行うやり方を示します。basic alignerの1つであるBowtie (Langmead et al., Genome Biol., 2009)を実装したRbowtieパッケージを内部的に使っています。Bowtie自体は、複数個所にマップされるリードの取り扱い(uniuniquely mapped reads or multi-mapped reads)を"-m"オプションで指定したり、許容するミスマッチ数を指定する"-v"などの様々なオプションを利用可能ですが、「基礎」のところではやり方を示しませんでした。ここでは、マッピングのオプションをいくつか変更して挙動を確認したり、複数のRNA-seqファイルを一度にマッピングするやり方を示します。尚、出力ファイルは、"\*\_bam", "\*\_QC.pdf", "\*\_bed"の3つです。それ以外のファイルは基本無視で大丈夫です。「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

### 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

998,521リードからなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092,219 bytes)と SRR616268sub\_trim3\_2.fastq.gz (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus casei 12A.GCA\_000309565.2.25.dna.chromosome.Chromosome.fa)がリファレンス配列です。オプションを"-m 1 --best --strata -v 0"とした例です。-m 1で1か所のみマップされるリード、-v 0で許容するミスマッチ数を0個にしています。-best --strataは、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしませんが、つけておいて悪さをするものではないので、通常は無条件でつけます。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定
```

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

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字、単位はsecond)
```



①以前のマッピング結果が残っている場所で実行してよいが、念のため②オブジェクトの全消去をやっておこう。

# マッピング応用

## 1. `mapping_paired_genome2.txt`中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

998,521リードからなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と [SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)) がリファレンス配列です。 オプションを "`-m 1 --best --strata -v 0`"とした例です。 `-m 1`で1か所のみマップされるリード、`-v 0`で許容するミスマッチ数を0個にしています。 `--best --strata`は、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしません。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #ゲノム配列ファイル
param_mapping <- "-m 1 --best --strata -v 0" #マッピングパラメータ

#必要なパッケージをロード
library(QuasR) #パッケージ
library(GenomicAlignments) #パッケージ

#本番(マッピング)
time_s <- proc.time() #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping) #マッピング
time_e <- proc.time() #計算時間
time_e - time_s #計算時間
out #マッピング結果
alignmentStats(out) #マッピング結果の統計

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments) #ファイル名
qQCReport(out, pdfFilename=out_f) #QCレポート作成
out_f #ファイル名

#ファイルに保存(BED形式ファイル)
```

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_genome"
> list.files()
[1] "Lactobacillus_casei_12a.GCA_000309565.2.25.chr$"
[2] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna$"
[3] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna$"
[4] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna$"
[5] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna$"
[6] "mapping_paired_genome2.txt"
[7] "QuasR_log_1be81c2f3864.txt"
[8] "SRR616268sub_trim3_1.fastq.gz"
[9] "SRR616268sub_trim3_1_1be82f4864d3.bam"
[10] "SRR616268sub_trim3_1_1be82f4864d3.bam.bai"
[11] "SRR616268sub_trim3_1_1be82f4864d3.bam.txt"
[12] "SRR616268sub_trim3_1_1be82f4864d3.bed"
[13] "SRR616268sub_trim3_1_1be82f4864d3_QC.pdf"
[14] "SRR616268sub_trim3_2.fastq.gz"

> rm(list = ls())
> ls()
character(0)
> |
```

# マッピング応用

終了後の状態。①マップされたリードの総数(494,912個)は、デフォルトの結果(693,500個)に比べて減っている。②デフォルトオプション実行時の-vで指定する許容する mismatches 数は、1以上だったのだろうと推測可

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノム

998,521リードからなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092,219 bytes)と SRR616268sub\_trim3\_2.fastq.gz (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供された(Lactobacillus casei 12a GCA\_000309565.2.25.dna.chromosome1.fastq.gz)とした例です。-m 1で1か所のみマップされるリード、-v 0で1以上の場合に効果を発揮します。ここでは意味をなしません

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome1.fastq.gz"
param_mapping <- "-m 1 --best --strata -v 0" #マッピングパラメータ

#必要なパッケージをロード
library(QuasR) #パッケージ
library(GenomicAlignments) #パッケージ

#本番(マッピング)
time_s <- proc.time() #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping) #マッピング
time_e <- proc.time() #計算時間
time_e - time_s #計算時間
out #マッピング結果
alignmentStats(out) #マッピング結果

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1]) #ファイル名
qQCReport(out, pdfFilename=out_f) #QCレポート
out_f #ファイル名

#ファイルに保存(BED形式ファイル)

```

```

R Console
> alignmentStats(out) #マッピング$
      seqlength mapped unmapped
nameae Paired:genome 2907892 494912 1502130
>
> #ファイルに保存(QCレポート用のpdfファイル作成)
> out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])
> qQCReport(out, pdfFilename=out_f) #QCレポート$
collecting quality control data
creating QC plots
> out_f #ファイル名$
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_genome$
>
> #ファイルに保存(BED形式ファイル)
> tmpfname <- out@alignments[,1] #ファイル名$
> for(i in 1:length(tmpfname)) { #サンプル数$
+   hoge <- readGAlignments(tmpfname[i]) #BAM形式フ$
+   hoge <- as.data.frame(hoge) #データフレ$
+   tmp <- hoge[, c("seqnames", "start", "end")] #必要$
+   out_f <- sub(".bam", ".bed", tmpfname[i]) #BED形$
+   out_f #ファイル名$
+   write.table(tmp, out_f, sep="\t", append=F, quot$
+ }
> |

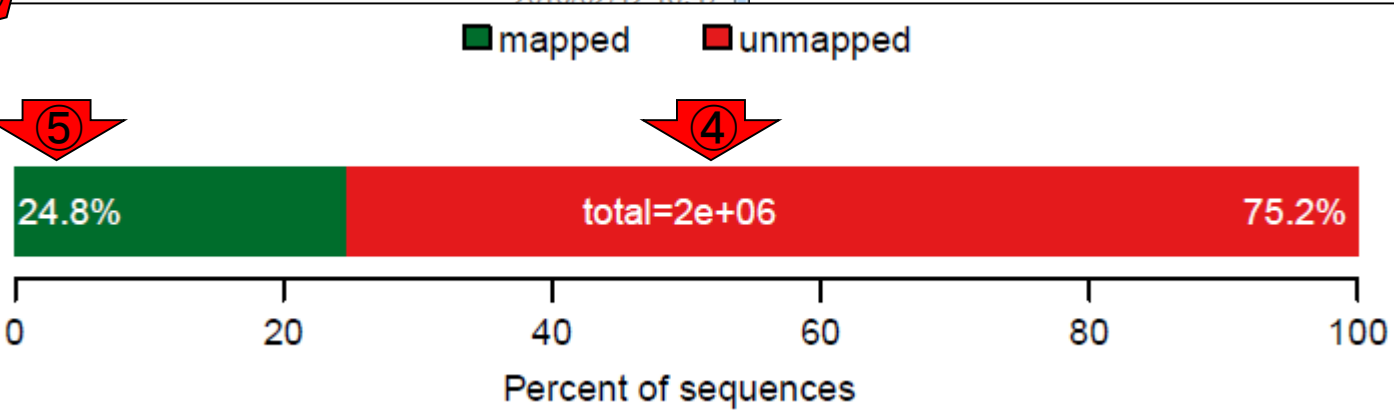
```

# マッピング応用

①赤枠部分が”-m 1 --best --strata -v 0”オプションをつけて実行した結果ファイル。②QCレポートPDF中の③マップされたリードの割合。④計1,997,042リード(2e+06)中、⑤24.8%がマップされた。494,912/1,997,042 = 0.2478なので妥当。

File Explorer window showing a directory of files. A red box highlights a group of files including QC reports and BAM files. Red arrows point from the text in the yellow box to these files.

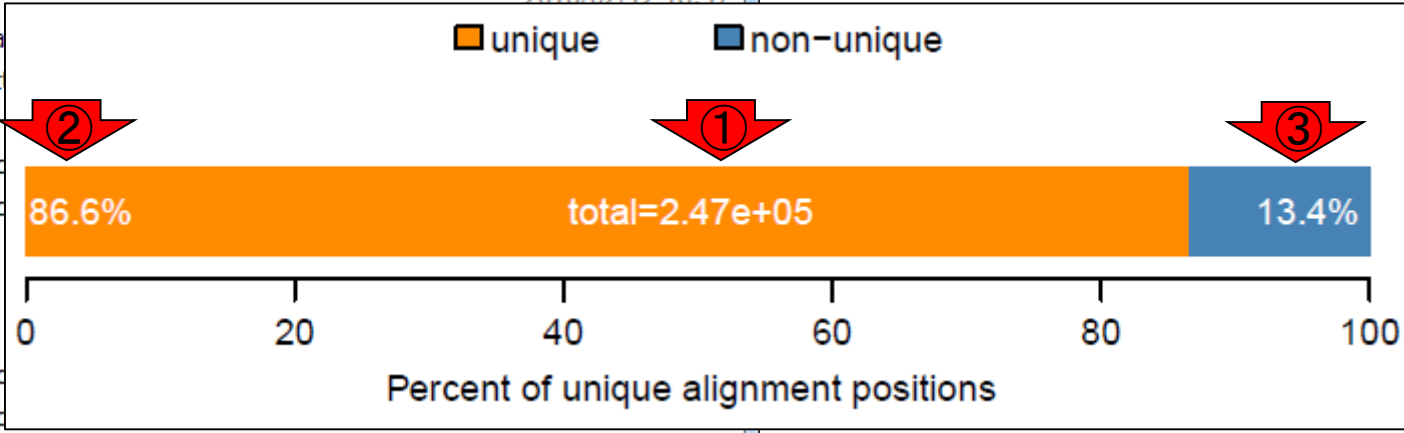
名前	更新日時
SRR616268sub_trim3_1_19cc24fc2749.bed	2016/02/13 14:25
SRR616268sub_trim3_1_19cc24fc2749_QC.pdf	2016/02/13 14:25
QuasR_log_19cc422f24eb.txt	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam.bai	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam.txt	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam	2016/02/13 14:24
SRR616268sub_trim3_1_1be82f4864d3.bed	2016/02/12 20:58
SRR616268sub_trim3_1_1be82f4864d3_QC.pdf	2016/02/12 19:30
QuasR_log_1be81c2f3864.txt	2016/02/12 18:47
SRR616268sub_trim3_1_1be82f4864d3.bam.ba	
SRR616268sub_trim3_1_1be82f4864d3.bam.tx	
SRR616268sub_trim3_1_1be82f4864d3.bam	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
SRR616268sub_trim3_2.fastq.gz	
SRR616268sub_trim3_1.fastq.gz	
mapping_paired_genome2.txt	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.Rbowtie	2016/02/12 18:44



# マッピング応用

①マップされたリード(片側247,456リードで計494,912リード)のうち、②1か所にのみマップされたリード(uniqely mapped reads)が86.6%、③複数個所にマップされたリード(non-unique)が13.4%。この結果は直感的にオカシイ。理由は“-m 1”として1か所にのみマップされるリードを出力しているつもりだから。

名前	更新日時
SRR616268sub_trim3_1_19cc24fc2749.bed	2016/02/13 14:25
SRR616268sub_trim3_1_19cc24fc2749_QC.pdf	2016/02/13 14:25
QuasR_log_19cc422f24eb.txt	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam.bai	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam.txt	2016/02/13 14:24
SRR616268sub_trim3_1_19cc24fc2749.bam	2016/02/13 14:24
SRR616268sub_trim3_1_1be82f4864d3.bed	2016/02/12 20:58
SRR616268sub_trim3_1_1be82f4864d3_QC.pdf	2016/02/12 19:30
QuasR_log_1be81c2f3864.txt	2016/02/12 18:47
SRR616268sub_trim3_1_1be82f4864d3.bam.ba	
SRR616268sub_trim3_1_1be82f4864d3.bam.tx	
SRR616268sub_trim3_1_1be82f4864d3.bam	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
SRR616268sub_trim3_2.fastq.gz	
SRR616268sub_trim3_1.fastq.gz	
mapping_paired_genome2.txt	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.d	
Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.Rbowtie	2016/02/12 18:44



# ?関数名

Bowtie実行時のオプションを眺めるべく、① alignmentParameterのところを詳細に調査。② qAlign関数の詳細を調べたいときは、③ ?qAlignと打つ。数秒後にhtmlマニュアルが開く

1. `mappingpaired_genome2.txt`中のFASTQ形式ファイルを乳酸菌ゲノムにマ

998,521リードからなるpaired-endのファイルです。 [SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と [SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。 [Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#)) がリファレンス配列です。オプションを `"-m 1 --best --strata -v 0"`とした例です。 `-m 1`で1か所のみマップされるリード、 `-v 0`で許容するミスマッチ数を0個にしています。 `--best --strata`は、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなませんが、つけておいて悪さをするものではないので、通常は無条件でつけます。

```

in_f1 <- "mappingpaired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定

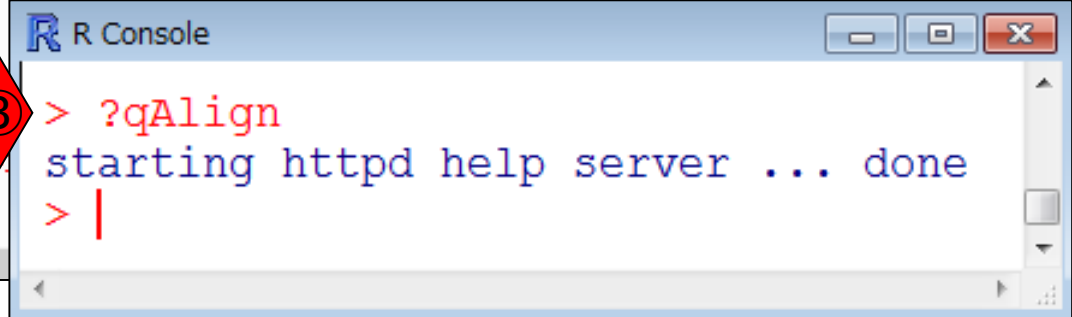
#必要なパッケージをロード
library(QuasR) #パッケージの読み込み
library(GenomicAlignments) #パッケージの読み込み

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
out #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out) #マッピング結果(alignment statistics)の表示。seqlength:リファレンス配列の

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])
qQCReport(out, pdfFilename=out_f) #QCレポート結果をファイル名を保存
out_f

#ファイルに保存(BED形式ファイル)

```



# 関数マニュアル

qAlign関数のhtmlマニュアル。ページ上部。①Usage (基本的な利用法)、②Arguments(オプションの説明)、③Details(詳細情報)、④Value(返り値;どんな結果が返ってくるか)などの情報が見られる。これはかなり難しい例なので、最初のうちはsum, mean, alphabetFrequencyなど、挙動を完全に把握できている関数のマニュアルを眺めて慣れておくとよい

qAlign {QuasR}

Align reads

## Description

Create read alignments against reference genome and optional auxiliary targets if not yet existing. If necessary, also build target indices for the aligner.

## Usage



```
qAlign(sampleFile,
       genome,
       auxiliaryFile=NULL,
       aligner="Rbowtie",
       maxHits=1,
       paired=NULL,
       splicedAlignment=FALSE,
       snpFile=NULL,
       bisulfite="no",
       alignmentParameter=NULL,
       projectName="qProject",
       alignmentsDir=NULL,
       lib.loc=NULL,
       cacheDir=NULL,
       clobj=NULL,
       checkOnly=FALSE)
```

## Arguments



sampleFile

the name of a text file listing input sequence files and sample names (see 'Details').

genome

the reference genome for primary alignments, one of:

- a string referring to a "BSgenome" package (e.g. "BSgenome.Hsapiens.UCSC.hg19"), which will be downloaded automatically from Bioconductor if not present
- the name of a fasta sequence file containing one

# 関数マニュアル

## Arguments

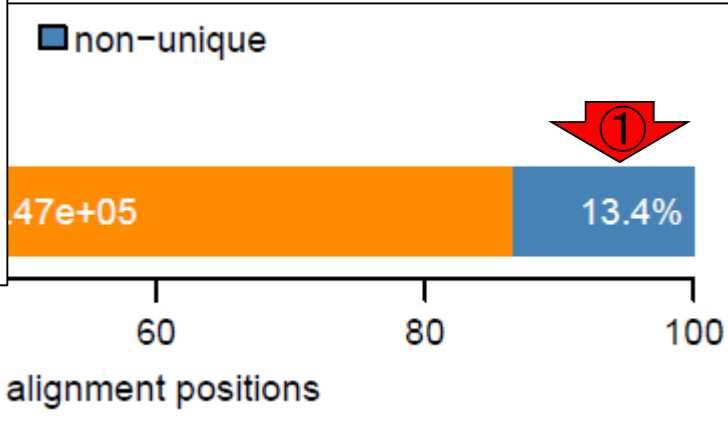
### alignmentParameter

a optional string containing command to be used for the aligner, to overrule the alignment parameters used by QuasR. Use with caution; some alignment parameters may contradict assumptions made by QuasR. Default parameters are listed in 'Details'.

調べたいalignmentParameterのところをつぎはぎで表示。結論として、“-m 1”として1か所へのみマップされるリードを出力しているつもりなのに、なぜ① non-uniqueが13.4%含まれるという結果になるのか理解できない。もしかしたら、②複数個所にマップされるリードはランダムにどこか1か所の結果が返されるということなのだろうか?少なくとも門田はマニュアルの説明だけでは挙動を完全にイメージできない。プログラムのバグかもしれないし、門田の勘違いかもしれないが、これ以上は深追いしない

## Details

If alignmentParameter is NULL (recommended), qAlign will select default parameters that are suitable for the experiment type. Please note that for bisulfite or allele-specific experiments, each read is aligned multiple times, and resulting alignments need to be combined. This requires special settings for the alignment parameters that are not recommended to be changed. For 'simple' experiments (neither bisulfite, allele-specific, nor spliced), alignments are generated using the parameters `-m maxHits --best --strata`. This will align reads with up to “maxHits” best hits in the genome and selects one of them randomly.



# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウンT情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)



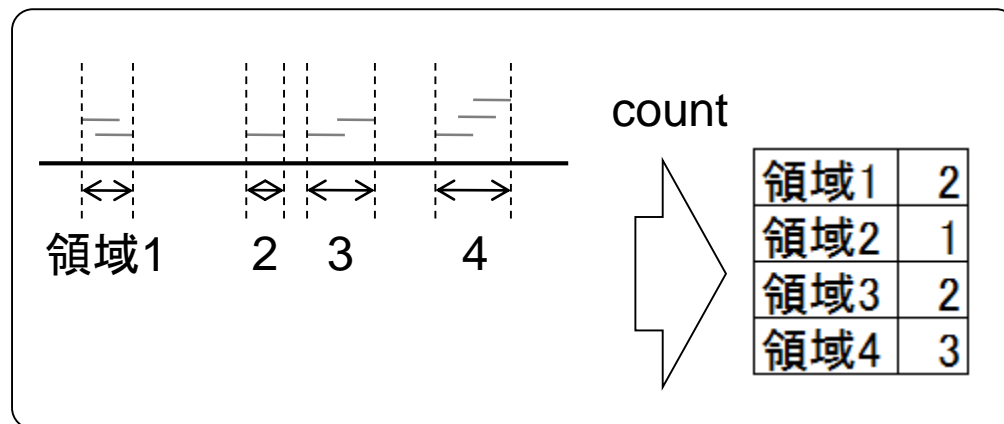
# カウント情報取得

## ■ アノテーション情報を利用する場合

- UCSC known Genes, Ensembl Genesなど様々なテーブル名を指定可能
- **gene**, **exon**, **promoter**, **junction**など様々なレベルを指定可能

## ■ アノテーション情報がない場合

- マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
- BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



# カウント情報取得

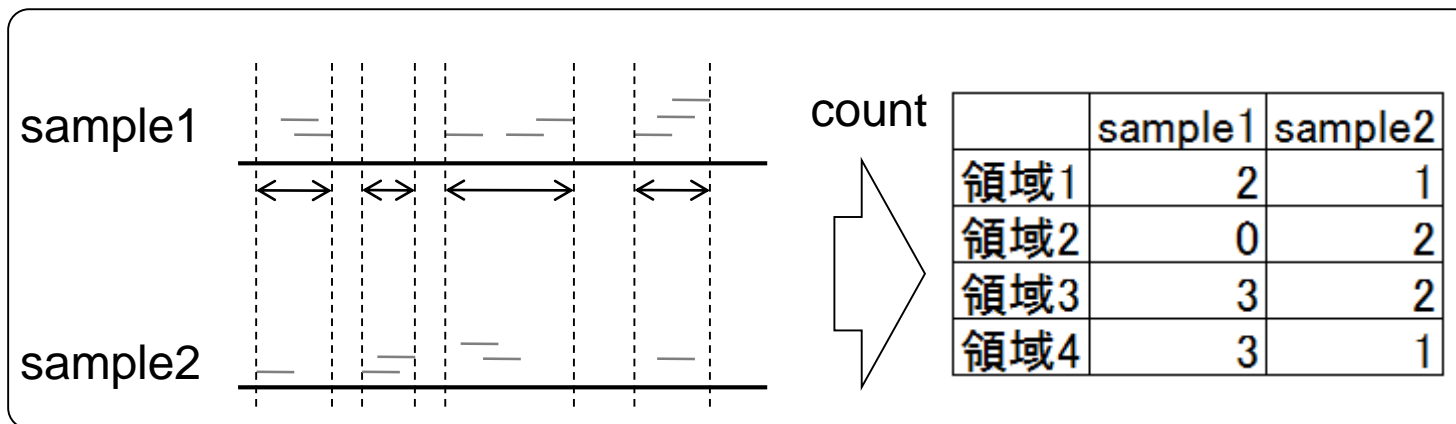
アノテーション情報がない場合の戦略は、複数サンプルの場合には領域が変わりうる。Cufflinksを知っているヒトはcuffmergeと同じイメージだと思えばよい

## ■ アノテーション情報を利用する場合

- UCSC known Genes, Ensembl Genesなど様々なテーブル名を指定可能
- gene, exon, promoter, junctionなど様々なレベルを指定可能

## ■ アノテーション情報がない場合

- マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
- BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



# カウント情報取得1

- マップ後 | 出力ファイルの読み込み | [htSeqTools\(Planet 2012\)](#) (last modified 2013/06/19)
- マップ後 | [カウント情報取得](#) | [について](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション無 | [QuasR\(Gaidatzis 2015\)](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | トランスクリプトーム | [QuasR\(Gaidatzis 2015\)](#) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | トランスクリプトーム | [BEDファイルから](#) (last modified 2014/06/21)

## マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有 | [QuasR\(Gaidatzis\\_2015\)](#) NEW

正規化 | 基礎 | [RP](#) [QuasR](#)パッケージを用いたpaired-end RNA-seqデータのリファレンスゲノム配列への[Bowtie](#)によるマッピングから、カウントデータ取得までの一連の流れを示します。

正規化 | 基礎 | [RP](#) 「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。



### ② 1. [mapping paired genome2.txt](#)中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル([SRR616268sub\\_1.fastq.gz](#)と[SRR616268sub\\_2.fastq.gz](#))から5および3'側を[rcode\\_20150707\\_preprocessing.txt](#)に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と[SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。[Ensembl \(Flicek et al., 2014\)](#)から提供されている[Lactobacillus casei 12A](#)のmulti-FASTA形式ゲノム配列ファイル([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#))とGFF3形式のアノテーションファイル([Lactobacillus casei 12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#))を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"#入力ファイル名を指定してin_f3に格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定: "gene", "exon", "promoter", "junction"

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

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです
```

「マッピング基礎」の例題2と基本的に同じ。違いは、①GFF形式のアノテーションファイルの指定、②カウントデータ取得のレベルを指定、③アノテーションファイルの読み込み…

# 違いを説明

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノム

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル([SRR616268sub\\_1.fastq.gz](#)と[SRR616268sub\\_2.fastq.gz](#))から5'および3'側を [rcode\\_20150707\\_preprocessing.txt](#)に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。  
[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と[SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#))とGFF3形式のアノテーションファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#))を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus casei 12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #入力ファイル名を指定してin
in_f3 <- "Lactobacillus casei 12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #①入力ファイル名を指定してin_f
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #② #カウントデータ取得時のレベルを指定："gene", "exon", "promoter", "junction"
```

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

```
#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto") #txdbオブジェクトの作成 #③
txdb #確認してるだけです
```

```
#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2) #マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
out #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out) #マッピング結果(alignment statistics)の表示。seqlength: リファレンス配列
```

# 違いを説明

①マッピング結果outとアノテーション情報txdbを読み込んで、param\_reportlevel(この場合geneレベル)で指定したレベルのカウントデータを取得するqCount実行部分、および②結果の保存の部分。

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノム

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル([SRR616268sub\\_1.fastq.gz](#)と[SRR616268sub\\_2.fastq.gz](#))から右および3'側を [rcode\\_20150707\\_preprocessing.txt](#)に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。  
[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と[SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#))とGFF3形式のアノテーションファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#))を読み込むやり方です。マッピングオプションはデフォルトです。

### #本番(マッピング)

```
time_s <- proc.time()
out <- qAlign(in_f1, in_f2)
time_e <- proc.time()
time_e - time_s
out
alignmentStats(out)
```

#計算時間を計測するため  
 #マッピングを行うqAlign関数を実行した結果をoutに格納  
 #計算時間を計測するため  
 #計算時間を表示(一番右側の数字。単位はsecond)  
 #マッピングに用いたパラメータや入力ファイルの情報などを表示  
 #マッピング結果(alignment statistics)の表示。seqlength: リファレンス配列

### #本番(カウントデータ取得)

```
count <- qCount(out, txdb, reportLevel=param_reportlevel)#カウントデータ行列を取得してcountに格納
dim(count)
head(count)
```

#行数と列数を表示  
 #確認してるだけです



### #ファイルに保存

```
tmp <- cbind(rownames(count), count) #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
```



### #ファイルに保存(QCレポート用のpdfファイル作成)

```
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quality Controlレポートのpdfファイル名を作成した結果をout_fに格納
qQCReport(out, pdfFilename=out_f)
out_f
```

#QCレポート結果をファイルに保存  
 #ファイル名を表示してるだけです

同じデフォルトオプションで実行した、「マッピング基礎」の例題2の結果がある①作業ディレクトリ上で、②念のためオブジェクトの全消去を行ってからコピペ。1分弱

# コピペで実行

## 1. `mapping paired genome2.txt`中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル(`SRR616268sub_1.fastq.gz`と `SRR616268sub_2.fastq.gz`)から5'および3'側を `rcode_20150707_preprocessing.txt`に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。  
`SRR616268sub_trim3_1.fastq.gz` (59,092,219 bytes)と `SRR616268sub_trim3_2.fastq.gz` (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されている `Lactobacillus casei 12A`の multi-FASTA形式ゲノム配列ファイル (`Lactobacillus casei 12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa`)とGFF3形式のアノテーションファイル (`Lactobacillus casei 12a.GCA_000309565.2.25.chromosome.Chromosome.gff3`)を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #入力ファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #入力ファイル名を指定してin_f3に格納
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定: "gene", "exon", "promoter", "junction"
```

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

```
#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto") #txdbを作成
txdb #確認して
```

```
#本番(マッピング)
time_s <- proc.time() #計算時間
out <- qAlign(in_f1, in_f2) #マッピング
time_e <- proc.time() #計算時間
time_e - time_s #計算時間
out #マッピング結果
alignmentStats(out) #マッピング結果
```

```
R Console
> getwd
function ()
.Internal(getwd())
<bytecode: 0x000000001785b630>
<environment: namespace:base>
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/L.casei_12A_genome"
> rm(list = ls())
> ls()
character(0)
> |
```

# 無駄を省く

1分弱で終わる理由は、マッピングを行っていないから。QuasRはまずログファイルを調べる。そして以前に同じ入力ファイル、同じオプションで実行した結果を見つけたら(QuasR\_log\_1be81c2f3864.txt)、①「(あなたが実行したい)全てのマッピング結果が見つかったよ」となり、以前の結果を読み込んでくれる

1. [mapping paired genome2.txt](#)中のFASTQ形式ファイルを乳酸菌ゲノムにマ  
 乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル  
 よび3'側を [rcode 20150707 preprocessing.txt](#)に書いてある手順でトリムして得  
[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と [SRR616268sub\\_trim3\\_2.f](#)  
 提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル  
[\(Lactobacillus casei\\_12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa\)](#)とGFF3形式のアノテーションファイル  
[\(Lactobacillus casei\\_12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3\)](#)を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #
out_f <- "hoge1.txt" #
param_reportlevel <- "gene" #

#必要なパッケージをロード
library(QuasR) #
library(GenomicFeatures) #

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="gff3") #
txdb #

#本番(マッピング)
time_s <- proc.time() #
out <- qAlign(in_f1, in_f2) #
time_e <- proc.time() #
time_e - time_s #
out #
alignmentStats(out) #
```

```
R Console
> #本番(マッピング)
> time_s <- proc.time() #計算時間を計測す$
> out <- qAlign(in_f1, in_f2) #マッピングを行う$
all necessary alignment files found ①
> time_e <- proc.time() #計算時間を計測す$
> time_e - time_s #計算時間を表示($
   ユーザ   システム   経過
   0.13    0.02    0.14
> out #マッピングに用い$
Project: qProject
Options  : maxHits      : 1
          paired       : fr
          splicedAlignment: FALSE
          bisulfite    : no
          snpFile      : none
Aligner  : Rbowtie v1.10.0 (parameters: -m 1 --best --st$
Genome   : .../Lactobacillus_casei_12a.GCA_000309565.2.2$
Reads    : 1 pair of files, 1 sample (fastq format):
```

# 無駄を省く

①「マッピング基礎」の例題2の結果と同じく、693,500リードがマップされた②outオブジェクトを利用可能なので、③qCount関数でカウントデータを得ることができる

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル(SRR616268sub\_1.fastq.gzとSRR616268sub\_2.fastq.gz)から5'および3'側を rcode\_20150707\_preprocessing.txtに書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092,219 bytes)とSRR616268sub\_trim3\_2.fastq.gz (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus casei\_12a.GCA\_000309565.2.25.dna.chromosome.Chromosome.fa)とGFF3形式のアノテーションファイル(Lactobacillus casei\_12a.GCA\_000309565.2.25.chromosome.Chromosome.gff3)を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #
out_f <- "hoge1.txt" #
param_reportlevel <- "gene" #

#必要なパッケージをロード
library(QuasR) #
library(GenomicFeatures) #

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="gff3") #
txdb #

#本番(マッピング)
time_s <- proc.time() #
out <- qAlign(in_f1, in_f2) #
time_e <- proc.time() #
time_e - time_s #
out #
alignmentStats(out) #
```

```
R Console

1. SRR61626....fastq.gz SRR61626....fastq.gz namee_pa$
Genome alignments: directory: same as reads
1. SRR616268sub_trim3_1_1be82f4864d3.bam
Aux. alignments: none

> alignmentStats(out) #マッピング結果 (a$
      seqlength mapped unmapped
namee_pa$genome 2907892 693500 1303542

> #本番(カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=param_reportlevel$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done

> dim(count) #行数と列数を表示
[1] 2727 2

> head(count) #確認してるだけで$
```



# qCount

①qCount関数実行によって得られたカウントデータ情報を含むcountオブジェクトの、②行数と列数は2727行×2列。③countオブジェクトの最初の6行分をhead関数で表示

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル([SRR616268sub\\_1.fastq.gz](#)と[SRR616268sub\\_2.fastq.gz](#))が5つのよび3'側を [rcode\\_20150707\\_preprocessing.txt](#)に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と[SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。[Ensembl \(Flicek et al., 2014\)](#)から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#))とGFF3形式のアノテーションファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#))を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #
out_f <- "hoge1.txt" #
param_reportlevel <- "gene" #

#必要なパッケージをロード
library(QuasR) #
library(GenomicFeatures) #

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="gff3") #
txdb #

#本番(マッピング)
time_s <- proc.time() #
out <- qAlign(in_f1, in_f2) #
time_e <- proc.time() #
time_e - time_s #
out #
alignmentStats(out) #
```

```
R Console
> # (カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=param_reportlevel$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行数と列数を表示
[1] 2727 2 #②
> head(count) #確認してるだけで$
      width namee_paired
LCA12A_0001    549      40
LCA12A_0002   1719      99
LCA12A_0003   1209   479
LCA12A_0004   1029   303
LCA12A_0005   1014   108
LCA12A_0006    354    22
>
> #ファイルに保存
> tmp <- cbind(rownames(count), count) #保存したい情報を$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row$
```

# qCount

①1列目は配列長、②2列目が目的のカウント情報。③param\_reportlevelで指定していたのは④gene。それゆえ⑤行名は、遺伝子名の通し番号のようにになっているのだろう。

## 1. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル(SRR616268sub\_1.fastq.gzとSRR616268sub\_2.fastq.gz)が5つのよび3'側を rcode\_20150707\_preprocessing.txtに書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。SRR616268sub\_trim3\_1.fastq.gz (59,092,219 bytes)とSRR616268sub\_trim3\_2.fastq.gz (54,667,920 bytes)です。Ensembl (Flicek et al., 2014)から提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus casei\_12a.GCA\_000309565.2.25.dna.chromosome.Chromosome.fa)とGFF3形式のアノテーションファイル(Lactobacillus casei\_12a.GCA\_000309565.2.25.chromosome.Chromosome.gff3)を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #
out_f <- "hoge1.txt" #
param_reportlevel <- "gene" # ④

#必要なパッケージをロード
library(QuasR)
library(GenomicFeatures)

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="gff3")
txdb

#本番(マッピング)
time_s <- proc.time()
out <- qAlign(in_f1, in_f2)
time_e <- proc.time()
time_e - time_s
out
alignmentStats(out)
```

```
R Console
> #本番(カウントデータ取得) ③
> count <- qCount(out, txdb, reportLevel=param_reportlevel$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行数と列数を表示
[1] 2727 ① ②
> head(count) #確認してるだけで$
      width name paired
⑤ LCA12A_0001    549     40
   LCA12A_0002   1719     99
   LCA12A_0003   1209   479
   LCA12A_0004   1029   303
   LCA12A_0005   1014   108
   LCA12A_0006    354    22
>
> #ファイルに保存
> tmp <- cbind(rownames(count), count) #保存したい情報を$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row$
```

# qCount

①遺伝子名の通し番号っぽいものは、②txdbオブジェクトの元情報である③in\_f3で読み込んだアノテーションファイル中に書かれている筈

## 1. mapping paired\_genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル([SRR616268sub\\_1.fastq.gz](#)と[SRR616268sub\\_2.fastq.gz](#))から5および3'側を [rcode\\_20150707\\_preprocessing.txt](#)に書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。  
[SRR616268sub\\_trim3\\_1.fastq.gz](#) (59,092,219 bytes)と[SRR616268sub\\_trim3\\_2.fastq.gz](#) (54,667,920 bytes)です。Ensembl ([Flicek et al., 2014](#))から提供されている [Lactobacillus casei 12A](#)の multi-FASTA形式ゲノム配列ファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.dna.chromosome.Chromosome.fa](#))とGFF3形式のアノテーションファイル ([Lactobacillus casei 12a.GCA\\_000309565.2.25.chromosome.Chromosome.gff3](#))を読み込むやり方です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome2.txt" #
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa" #
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3" #
out_f <- "hoge1.txt" #
param_reportlevel <- "gene" #

#必要なパッケージをロード
library(QuasR) #
library(GenomicFeatures) #

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="gff3") #
txdb #

#本番(マッピング)
time_s <- proc.time() #
out <- qAlign(in_f1, in_f2) #
time_e <- proc.time() #
time_e - time_s #
out #
alignmentStats(out) #
```

```
R Console
> #本番(カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=param_reportlevel$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行数と列数を表示
[1] 2727 2
> head(count) #確認してるだけで$
      width name paired
1 LCA12A_0001    549    40
2 LCA12A_0002   1719    99
3 LCA12A_0003   1209   479
4 LCA12A_0004   1029   303
5 LCA12A_0005   1014   108
6 LCA12A_0006    354    22
>
> #ファイルに保存
> tmp <- cbind(rownames(count), count) #保存したい情報を$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row$
```



# GFFを眺めて確認

GFFファイルをエクセルで眺めている。①の文字列が②out\_fで指定した出力ファイル(hoge1.txt)中の③行名として使われているのだろう

```
##gff-version 3
##sequence-region Chromosome 1 2907892
Chromosome ena gene 1 1350 . + . ID=gene:LCA12A_0617;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 1523 2662 . + . ID=gene:LCA12A_0618;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 3240 3452 . + . ID=gene:LCA12A_0619;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 3449 4564 . + . ID=gene:LCA12A_0620;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 4817 6778 . + . ID=gene:LCA12A_0621;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 6840 9461 . + . ID=gene:LCA12A_0622;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 9566 10270 . - . ID=gene:LCA12A_0623;assembly_name=GCA_000309565.2;bi
Chromosome ensembl CDS 1 1350 . + . 0 ID=CDS:EKP96483;Parent=transcript:EKP96483;assembly_na
```

```
collapsing counts by query name...done
> dim(count) #行数と列数を表示
[1] 2727 2
> head(count) #確認してるだけで$
      width name_paired
LCA12A_0001 549 40
LCA12A_0002 1719 99
LCA12A_0003 1209 479
LCA12A_0004 1029 303
LCA12A_0005 1014 108
LCA12A_0006 354 22
>
> #③に保存
> tmp <- cbind(rownames(count), count) #保存したい情報を$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row$
```

# GFFを眺めて確認

基本テクニックを駆使して、①配列長1350 bpのLCA12A\_0617を②確認。同じ配列長になっており妥当。完全に欠番のない通し番号になっていれば③count行列の617行目という指定でもイケる…わけではない。④tailで納得

```
##gff-version 3
##sequence-region Chromosome 1 2907892
Chromosome ena gene 1 1350 . + . ID=gene:LCA12A_0617;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 1523 2662 . + . ID=gene:LCA12A_0618;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 3240 3452 . + . ID=gene:LCA12A_0619;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 3449 4564 . + . ID=gene:LCA12A_0620;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 4817 6778 . + . ID=gene:LCA12A_0621;assembly_name=GCA_000309565.2;bi
Chromosome ena gene 6840
Chromosome ena gene 956
Chromosome ensembl CDS 1
```

```
R Console
> count["LCA12A_0617", ]
      width nameae_paired
      1350           363
> count[617, ]
      width nameae_paired
      858           27
> tail(count)
      width nameae_paired
transcript:LCA12A_2828  405           15
transcript:LCA12A_2835 1816          302
transcript:LCA12A_2837 1538          110
transcript:LCA12A_2840  630           24
transcript:LCA12A_2843  993           25
transcript:LCA12A_2845 1495           32
> dim(count)
[1] 2727  2
> |
```

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# カウント情報取得2

- マップ後 | 出力ファイルの読み込み | htSeqTools(Planet 2012) (last modified 2013/06/19)
- マップ後 | カウント情報取得 | について (last modified 2014/12/17)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | single-end | ゲノム | アノテーション無 | QuasR(Gaidatzis 2015) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション無 | QuasR(Gaidatzis 2015) (last modified 2014/12/17)
- マップ後 | カウント情報取得 | paired-end | トランスクリプトーム | QuasR(Gaidatzis 2015) (last modified 2014/12/17)



## マップ後 | カウント情報取得 | paired-end | ゲノム | アノテーション有 | QuasR(Gaidatzis 2015) NEW

QuasRパッケージを用いたpaired-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。

「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. mapping paired\_genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分のpaired-endファイル(SRR616268sub\_1.fastq.gzとSRR616268sub\_2.fastq.gz)から50万リードをrcode\_20150707\_preprocessing.txtに書いてある手順でトリムして得られた998,521リードからなるpaired-endのファイルです。



### 4. mapping paired\_genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

3と基本的に同じですが、マッピング時のオプションを"-m 1 --best --strata -v 0"とした例です。-m 1で1か所にもみマップされるリード、-v 0で許容するミスマッチ数を0個にしています。--best --strataは、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしません。つけておいて悪さをするものではないので、通常は無条件につけます。

in\_f1 <-  
in\_f2 <-  
in\_f3 <-  
out\_f1 <-  
out\_f2 <-  
param\_reportlevel <-  
param\_mapping <-  
  
#必要な  
library  
library  
  
#前処理  
txdb <-  
txdb

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"#入力ファイル名を指定してin_f3に格納
out_f1 <- "hoge4_count.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_length.txt" #出力ファイル名を指定してout_f2に格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定:"gene", "exon", "promoter", "junction"
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定

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

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトの作成
    
```

# ① カウント情報取得2

①例題4は「マッピング応用」の例題1と同じくマッピング時のオプションを明示的に指定している。②許容する mismatches 数を0(-v 0)にしたときのマップされたリード数(494,912個)は、デフォルトの結果(693,500個)よりも少なかった。カウント行列データの数値は全体的に少なくなると予想できるので、それを確認するのが主目的。

## 4. mapping paired genome2.txt 中のFASTQ形式ファイルを乳酸菌

3.と基本的に同じですが、マッピング時のオプションを"-m 1 --best --strata"にする mismatches 数を0個にしています。--best --strataは、許容する mismatches が、つけておいて悪さをするものではないので、通常は無条件でつ

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"#入力ファイル名を指定してin_f3に格納
out_f1 <- "hoge4_count.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_length.txt" #出力ファイル名を指定してout_f2に格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定："gene", "exon", "promoter", "junction"
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定

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

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実行した結果をoutに格納
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
    
```



②



コピーで実行し、全体像を見られるところを表示。②  
 マップされたリード数は、確かに494,912個だった。③  
 カウント行列データの数値は確かに全体的に少ない

# ① カウント情報取得2

4. `mapping_paired_genome2.txt`中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

3と基本的に同じですが、マッピング時のオプションを"`-m 1 --best --strata -v 0`"とした例です。`-m 1`で1か所のみマップされるリード、`-v 0`で許容するミスマッチ数を0個にしています。`--best --strata`は、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしません  
 が、つけておいて悪さをするものではないので、通常は無条件でつけます。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosomes.fasta" #ゲノムファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosomes.gff" #ゲノムアノテーションファイル名を指定してin_f3に格納
out_f1 <- "hoge4_count.txt" #出力ファイル名を指定
out_f2 <- "hoge4_length.txt" #出力ファイル名を指定
param_reportlevel <- "gene" #カウントデータ取得時の報告レベルを指定
param_mapping <- "-m 1--best --strata -v 0"#マッピング時のオプション

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

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトを作成
txdb #確認してるだけです

#本番(マッピング)
time_s <- proc.time() #計算時間を計測するた
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping) #マッピングを実行
time_e <- proc.time() #計算時間を計測するた
time_e - time_s #計算時間を表示(一番)
```

```
R Console
> alignmentStats(out) #マ$
          seqlength mapped unmapped
name_paired:genome 2907892 494912 1502130
>
> #本番(カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=par$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行$
[1] 2727 2
> head(count) #確$
      width name_paired
LCA12A_0001 549 27
LCA12A_0002 1719 60
LCA12A_0003 1209 350
LCA12A_0004 1029 217
LCA12A_0005 1014 65
LCA12A_0006 354 11
>
```

①許容するミスマッチ数を0(-v 0)にしたときの結果のほうが、確かに②デフォルトの結果に比べて少ない

# 比較



```
R Console
> alignmentStats(out) #マ$
      seqlength mapped unmapped
name_paired:genome 2907892 693500 13035
>
> #本番 (カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=pa
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行$
[1] 2727 2
> head(count) #確$
      width name_paired
LCA12A_0001 549 40
LCA12A_0002 1719 99
LCA12A_0003 1209 479
LCA12A_0004 1029 303
LCA12A_0005 1014 108
LCA12A_0006 354 22
>
```

```
R Console
> alignmentStats(out) #マ$
      seqlength mapped unmapped
name_paired:genome 2907892 494912 1502130
>
> #本番 (カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=par$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行$
[1] 2727 2
> head(count) #確$
      width name_paired
LCA12A_0001 549 27
LCA12A_0002 1719 60
LCA12A_0003 1209 350
LCA12A_0004 1029 217
LCA12A_0005 1014 65
LCA12A_0006 354 11
>
```

例題1と①例題4のもう一つの違いは、②配列長(hoge4\_length.txt)と③カウント情報(hoge4\_count.txt)を別々に出力している点

# 出力ファイル

①

## 4. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

3.と基本的に同じですが、マッピング時のオプションを"-m 1 --best --strata -v 0"とした例です。-m 1で1か所のみマップされるリード、-v 0で許容するミスマッチ数を0個にしています。--best --strataは、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしません。つけておいて悪さをするものではないので、通常は無条件でつけます。

```

in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25
out_f1 <- "hoge4_count.txt" #出力ファイル名
out_f2 <- "hoge4_length.txt" #出力ファイル名
param_reportlevel <- "gene" #カウントデータ
param_mapping <- "-m 1 --best --strata -v 0"#マッピング

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

#前処理(アノテーション情報を取得)
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトを作成
txdb #確認してるだけ

#本番(マッピング)
time_s <- proc.time() #計算時間を計測
out <- qAlign(in_f1, in_f2, alignmentParameter=param) #マッピング実行
time_e <- proc.time() #計算時間を計測
time_e - time_s #計算時間を表示
    
```

```

R Console
> alignmentStats(out) #マ$
          seqlength mapped unmapped
nameae_paired:genome 2907892 494912 1502130
>
> #本番(カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=par$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> dim(count) #行$
[1] 2727      2      3
> head(count) #確$
      width nameae_paired
LCA12A_0001 549          27
LCA12A_0002 1719          60
LCA12A_0003 1209         350
LCA12A_0004 1029         217
LCA12A_0005 1014          65
LCA12A_0006 354           11
    
```



# 出力ファイル

②カウント情報ファイル(hoge4\_count.txt)の中身はこんな感じ。配列情報の列を含まないものが一般的なカウントデータ以降の統計解析の入力ファイルとして利用される。

## 4. mapping paired genome2.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピング

3.と基本的に同じですが、マッピング時のオプションを"-m 1 --best --strata -v 0"とした例です。-m 1で1か所のみマッピングされるリード、-v 0で許容するミスマッチ数を0個にしています。--best --strataは、許容するミスマッチ数が1以上の場合に効果を発揮します。ここでは意味をなしません。つけておいて悪さをするものではないので、通常は無条件でつけます。

```
in_f1 <- "mapping_paired_genome2.txt" #入力ファイル名を指定してin_f1に格納(RNA-seqリストファイル)
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指定してin_f2に格納
in_f3 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"#入力ファイル名を指定してin_f3に格納
out_f1 <- "hoge4_count.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_length.txt" #出力ファイル名を指定してout_f2に格納
param_reportlevel <- "gene" #カウントデータ取得時のレベルを指定："gene", "exon", "promoter", "junction"
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定
```

#必要なパッケージをロード

```
library(QuasR) #パッケージの読み込み
library(GenomicFeatures) #パッケージの読み込み
library(GenomicAlignments) #パッケージの読み込み
```

#前処理(アノテーション情報を取得)

```
txdb <- makeTxDbFromGFF(in_f3, format="auto")#txdbオブジェクトの作成
txdb #確認してるだけです
```

#本番(マッピング)

```
time_s <- proc.time() #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数
time_e <- proc.time() #計算時間を計測するため
time_e - time_s #計算時間を表示(一番右側の数字。単位はsecond)
```

rownames(data)	count[, -1]
LCA12A_0001	27
LCA12A_0002	60
LCA12A_0003	350
LCA12A_0004	217
LCA12A_0005	65
LCA12A_0006	11
LCA12A_0007	95
LCA12A_0008	159
LCA12A_0009	21
LCA12A_0011	46

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# カウントデータ解析

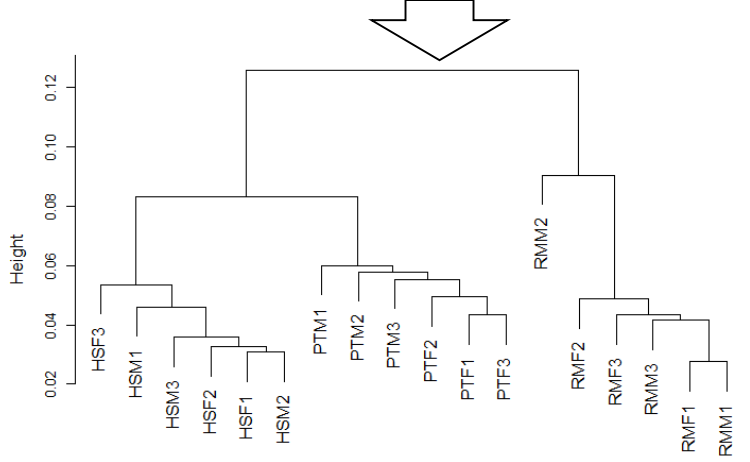
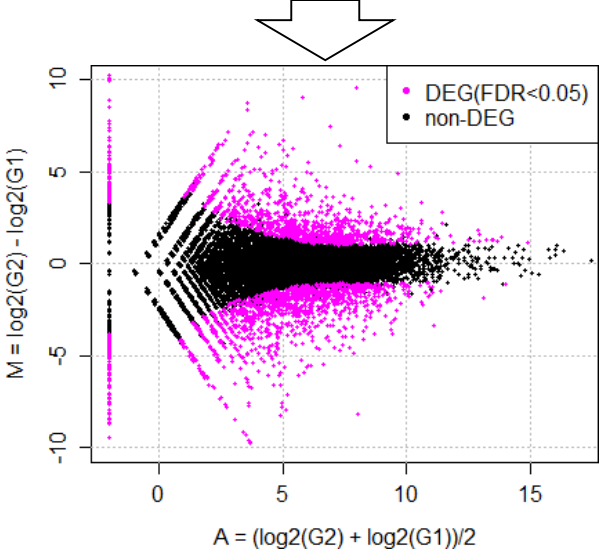
カウントデータ

このデータ(Blekhman et al., 2010)は、3種類の生物種間比較。20,689 genes × 18samples。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)。生物種ごとにメス3匹、オス3匹。

20,689 genes	ヒト ( <i>Homo sapiens</i> , HS)			チンパンジー ( <i>Pan troglodytes</i> , PT)			アカゲザル ( <i>Rhesus macaque</i> , RM)											
	メス(Female)			オス(Male)			メス			オス								
	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3
ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
ENSG0000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
ENSG0000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
ENSG0000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
ENSG0000000000938	44	65	210	73	43	65	84	104	76	198	31	58	73	28	54	80	34	72
ENSG0000000000971	4765	7225	3405	3600	6383	5546	5382	8331	4335	2568	5019	2653	13566	9964	18247	14236	5196	11834
ENSG0000000001036	297	251	189	200	234	249	305	301	313	254	151	331	292	106	379	201	88	140
ENSG0000000001084	630	737	306	336	984	459	417	328	885	298	569	218	1062	786	1110	873	664	1752
ENSG0000000001167	36	30	36	29	33	28	63	80	25	69	74	41	62	34	108	97	35	61
ENSG0000000001460	3	1	5	1	4	2	0	1	1	1	1	3	1	1	1	0	1	3
ENSG0000000001461	49	37	34	28	62	32	75	69	40	90	69	60	210	92	176	247	81	117
ENSG0000000001467	117	82	89	80	131	110	125	89	75	109	130	131	139	95	197	137	159	179

発現変動遺伝子(DEG)同定

サンプル間クラスタリング



# サンプル間クラスタリング

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 (Blekhman et al., 2010)のサンプル間クラスタリング。データの取得や整形については、2015.07.29の講義資料を参考。

- ・ 解析 | 発現量推定(トランスクリプトーム配列を利用) (last modified 2014/07/09)
- ・ 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 (last modified 2014/02/05)

## 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。

「ファイル名」を「デフォルト」に変更して解析したいファイル名を置いてあるディレクトリに移動し、以下をコピー

### 8. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

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

Neyret-  
ンゲル

in\_f  
out\_f  
param

#必要  
libra

#入力  
data  
dim(d

#本番  
out <

```

in_f <- "sample_blekhman_18.txt"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.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を読み込み
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設定
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

```

# 入力ファイル



デスクトップ上のhogeフォルダ中に① sample\_blekhman\_18.txtが存在するはず。ファイルが存在しないヒトは、②右クリック、③対象をファイルに保存、でデスクトップ上のhogeに保存

## 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

```
in_f <- "sample_blekhman_18.txt" #入力ファイル
out_f <- "hoge8.png" #出力ファイル
param_fig <- c(700, 400) #パラメータ
```

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

```
#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1)
dim(data) #次元
```

```
#本番実行
out <- heatmap.2(data, col="red", row.names=1, #ファイル名指定
```

開く(O)  
 新しいタブで開く(W)  
 新しいウィンドウで開く(N)  
 対象をファイルに保存(A)... ③ (hogeフォルダに保存)  
 対象を印刷(P)  
 切り取り  
 コピー(C)  
 ショートカットのコピー(T) (Ctrl+C)  
 貼り付け(P)

	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3
ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
ENSG000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
ENSG000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
ENSG000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
ENSG000000000938	44	65	210	73	43	65	84	104	78	188	84	58	78	88	54	88	84	72
ENSG000000000971	4765	7225	3405	3600	6383	5546	5382	8331	78	188	84	58	78	88	54	88	84	72
ENSG000000001036	297	251	189	200	234	249	305	301	78	188	84	58	78	88	54	88	84	72
ENSG000000001084	630	737	306	336	984	459	417	328	78	188	84	58	78	88	54	88	84	72
ENSG000000001167	36	30	36	29	33	28	63	80	78	188	84	58	78	88	54	88	84	72
ENSG000000001460	3	1	5	1	4	2	0	1	78	188	84	58	78	88	54	88	84	72
ENSG000000001461	49	37	34	28	62	32	75	69	78	188	84	58	78	88	54	88	84	72
ENSG000000001487	117	93	88	80	131	110	125	88	78	188	84	58	78	88	54	88	84	72

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="blekh")
[1] "sample_blekhman_18.txt"
> |
```



# 入力ファイル

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)。生物種ごとにメス3匹、オス3匹。雄雌を考慮しなければbiological replicates (生物学的な反復)は6。

	ヒト ( <i>Homo sapiens</i> ; HS)						チンパンジー ( <i>Pan troglodytes</i> ; PT)						アカゲザル ( <i>Rhesus macaque</i> ; RM)					
	メス(Female)			オス(Male)			メス			オス			メス			オス		
	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3
ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
ENSG000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
ENSG000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
ENSG000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
ENSG000000000938	44	65	210	73	43	65	84	104	76	198	31	58	73	28	54	80	34	72
ENSG000000000971	4765	7225	3405	3600	6383	5546	5382	8331	4335	2568	5019	2653	13566	9964	18247	14236	5196	11834
ENSG000000001036	297	251	189	200	234	249	305	301	313	254	151	331	292	106	379	201	88	140
ENSG000000001084	630	737	306	336	984	459	417	328	885	298	569	218	1062	786	1110	873	664	1752
ENSG000000001167	36	30	36	29	33	28	63	80	25	69	74	41	62	34	108	97	35	61
ENSG000000001460	3	1	5	1	4	2	0	1	1	1	1	3	1	1	1	0	1	3
ENSG000000001461	49	37	34	28	62	32	75	69	40	90	69	60	210	92	176	247	81	117
ENSG000000001487	117	93	88	80	131	110	125	98	75	108	130	131	138	95	187	137	158	172

20,689 genes

# コピーで実行

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。

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。

```

in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.png" #出力ファイル名を指定してout_fに格納
param_fig <- c(700, 400) #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, as.is=TRUE)
dim(data)

```

```

#本番
out <- clusterSample(data, distance="euclidean",
                     hclust.method="ward.D2")

```

```

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2],
     par(mar=c(0, 4, 1, 0)))
plot(out, sub="", xlab="", cex=1.3, main="", ylab="Heatmap of gene expression")
dev.off()

```

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

R Console

詳しくは 'contributors()' と入力してください。  
 また、R や R のパッケージを出版物で引用する際の形\$  
 'citation()' と入力してください。

'demo()' と入力すればデモ  
 'help()' とすればオンライン  
 'help.start()' で HTML  
 'q()' と入力すれば R を終

[以前にセーブされたワークスペース]

```

> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="blekh")
[1] "sample_blekhman_18.txt"
> |

```

- コピー Ctrl+C
- ペースト **②** Ctrl+V
- コマンドのペースト
- コピー&ペースト Ctrl+X
- ウインドウの消去 Ctrl+L
- 全て選択
- バッファに出力 Ctrl+W
- ウインドウを常にトップに置く

# 実行結果

エラーなく実行できると右下のような画面になっているはずですが。①入力ファイル情報を格納した行列dataの行数が20,689、列数が18となっていることがわかります。

## 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。

```

in_f <- "sample_blekhman_18.txt"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.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, #オブジェクト
dim(data)

#本番
out <- clusterSample(data, dist.method="spearman",
hclust.method="average", unique.patter

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2],
par(mar=c(0, 4, 1, 0)) #下、左、上、右の余白を指定
plot(out, sub="", xlab="", cex.lab=1.2, #樹形図(デフォルト)
cex=1.3, main="", ylab="Height") #樹形図(デフォルト)
dev.off() #おまじない
    
```

```

R Console
> dim(data) #オブジェ$
[1] 20689 18
>
> #本番
> out <- clusterSample(data, dist.method="spearman"$
+ hclust.method="average", unique.patte$
>
> #ファイルに保存
> png(out_f, pointsize=13, width=param_fig[1], heig$
> par(mar=c(0, 4, 1, 0)) #下、左、$
> plot(out, sub="", xlab="", cex.lab=1.2, #樹形図(デ$
+ cex=1.3, main="", ylab="Height") #樹形図(デ$
> dev.off() #おまじない
null device
      1
> |
    
```

①出力ファイルのサイズを指定しているの、こんな感じになる

# 出力ファイル

## 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。

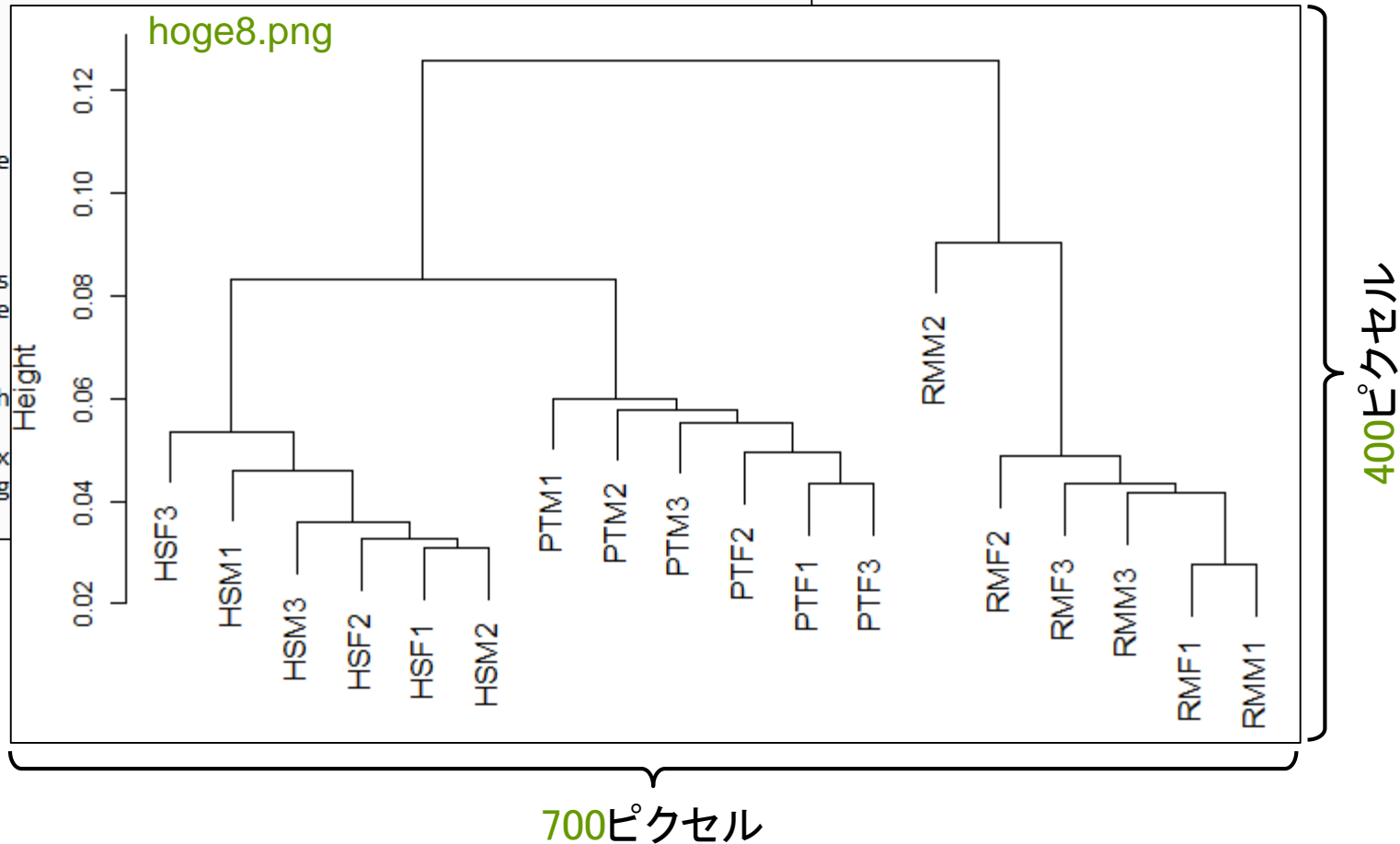
```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.png" #出力ファイル名を指定してout_fに格納
param_fig <- c(700, 400) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
```

```
#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, as.is=TRUE)
dim(data)

#本番
out <- clusterSample(data, distance="euclidean", hclust.method="ave")

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2],
    par(mar=c(0, 4, 1, 0)))
plot(out, sub="", xlab="", cex=1.3, main="", ylab="Height",
    dev.off())
```



3生物種間全体で眺めると、①ヒト(HS)とチンパンジー(PT)はよく似ている。

# 結果の解釈

## ■ ヒト(HS)

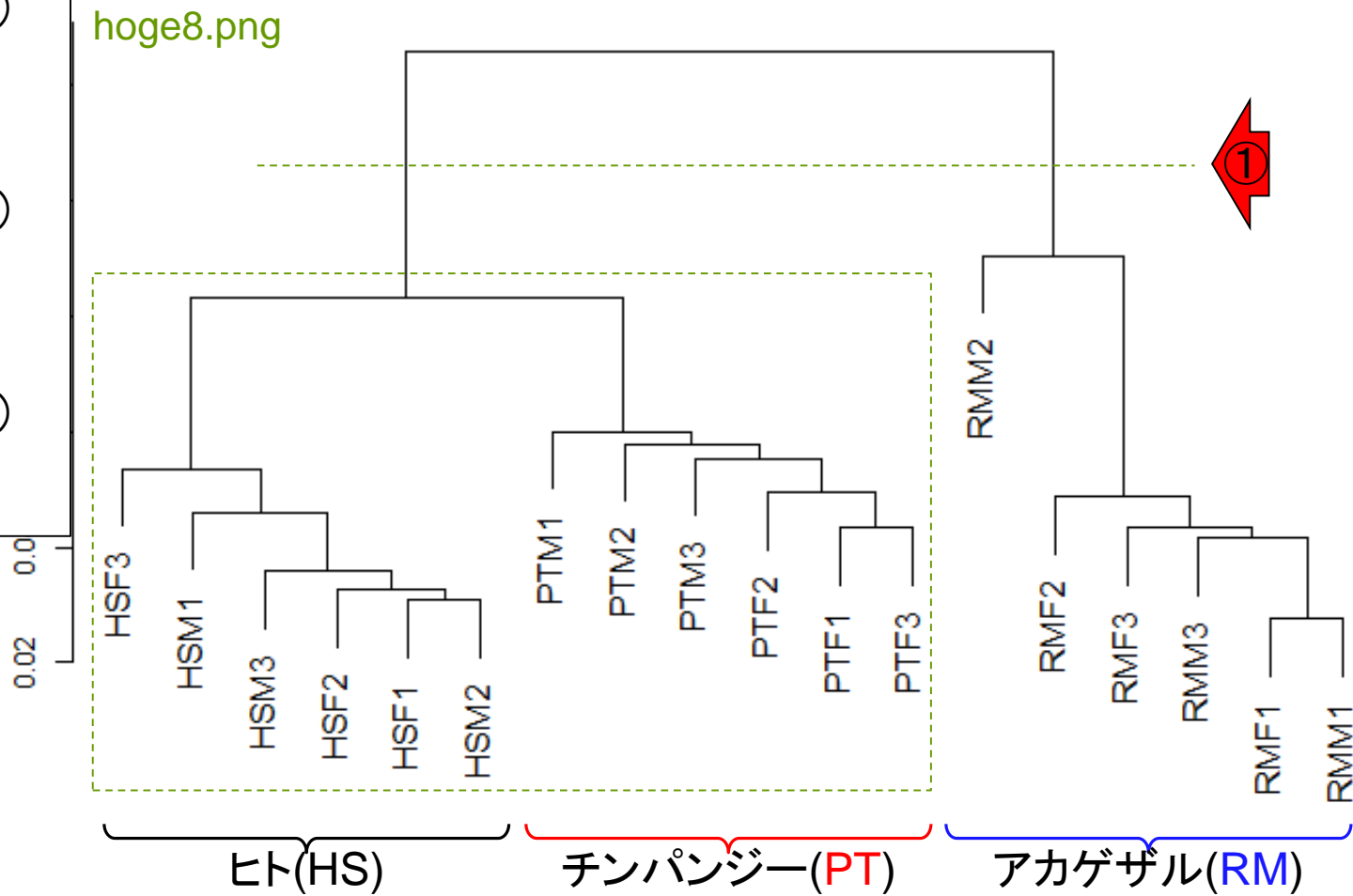
- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

## ■ 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)

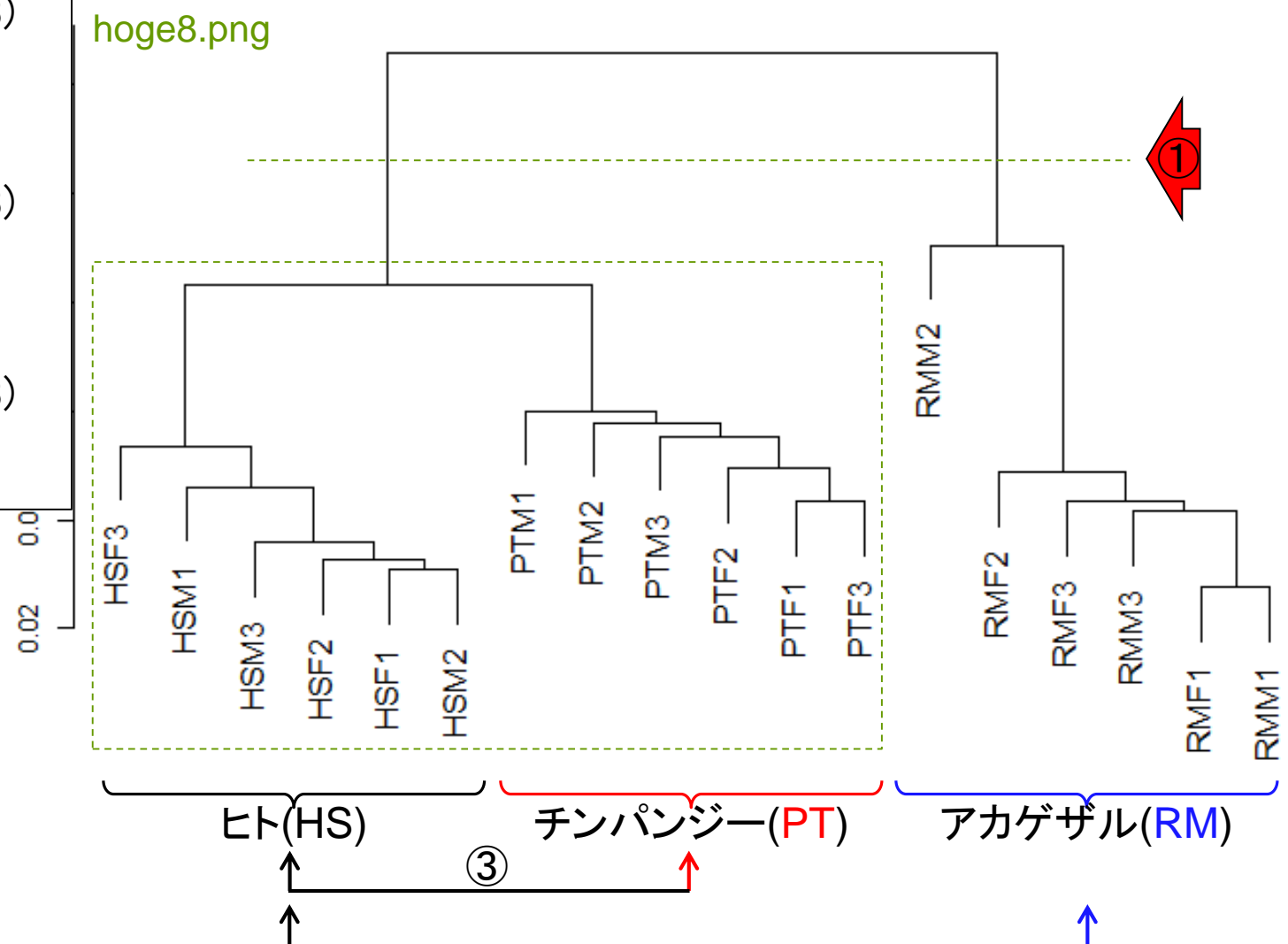
- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)



2群間比較(発現変動遺伝子検出; DEG検出)を行ったときに、②「HS vs. RMで得られるDEG数」のほうが③「HS vs. PTで得られるDEG数」よりも多そう。

# 結果の解釈

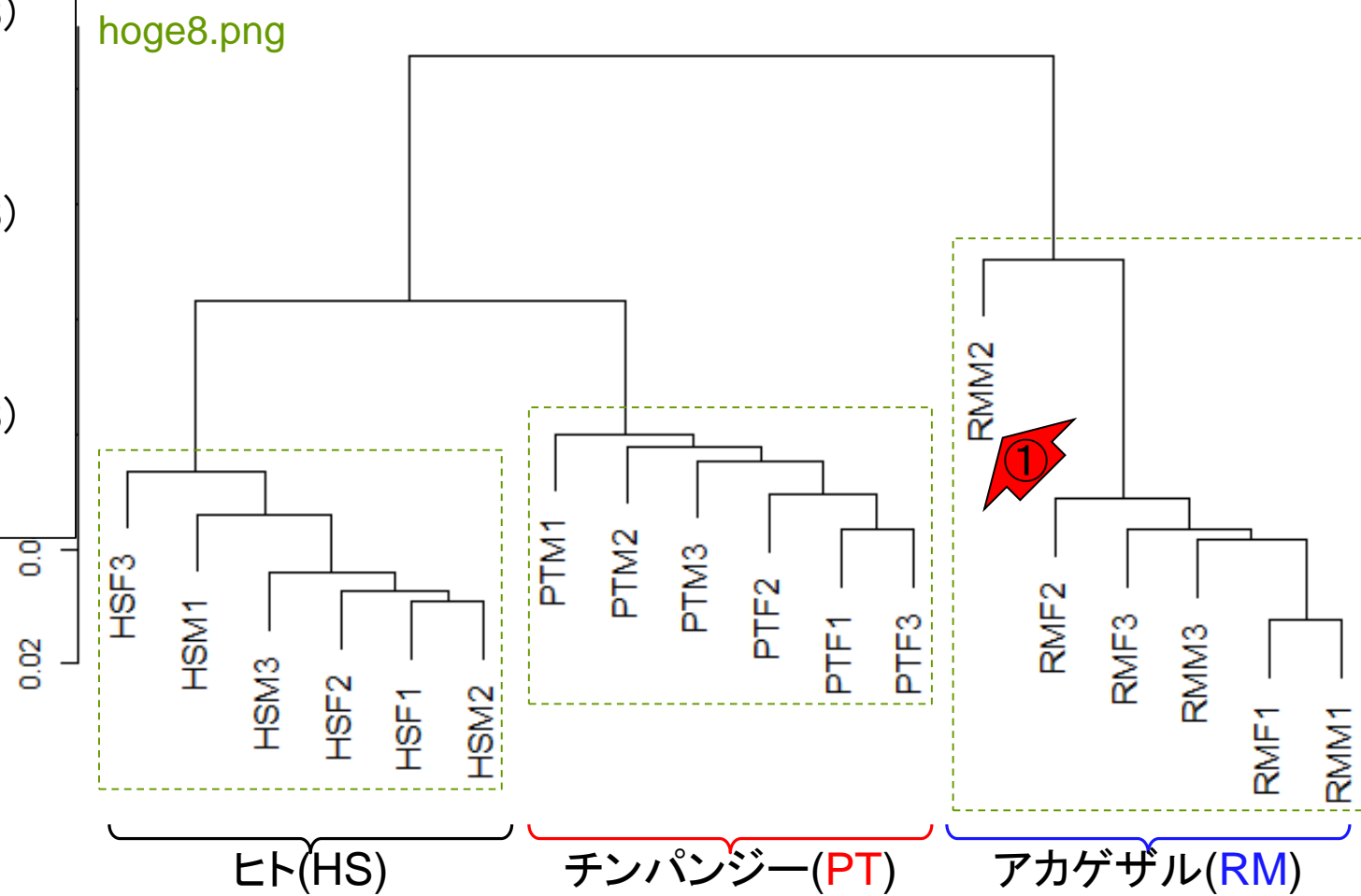
- ヒト(HS)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)
- チンパンジー(PT)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)
- アカゲザル(RM)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)



同一生物種でクラスターを形成している。  
①RMM2は「外れサンプル」っぽい。

# 結果の解釈

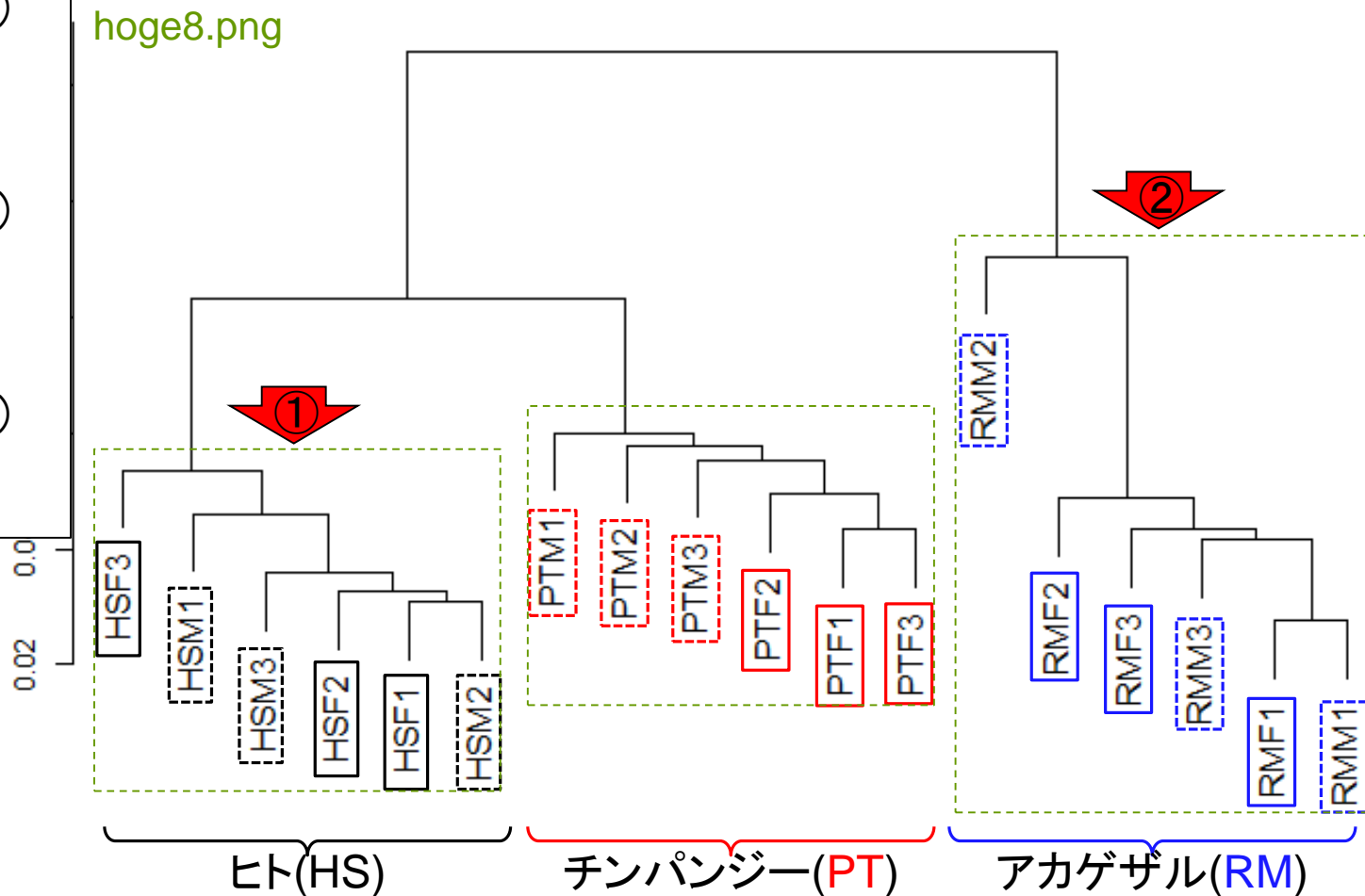
- ヒト(HS)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)
- チンパンジー(PT)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)
- アカゲザル(RM)
  - オス3匹(M1, M2, M3)
  - メス3匹(F1, F2, F3)



# 結果の解釈

①ヒト(HS)と②アカゲザル(RM)は、メスとオスのサンプルが入り混じっている。これらの生物種内で、「メス群 vs. オス群」の2群間比較を行ってもDEGはほとんど検出されないだろう。

- 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)

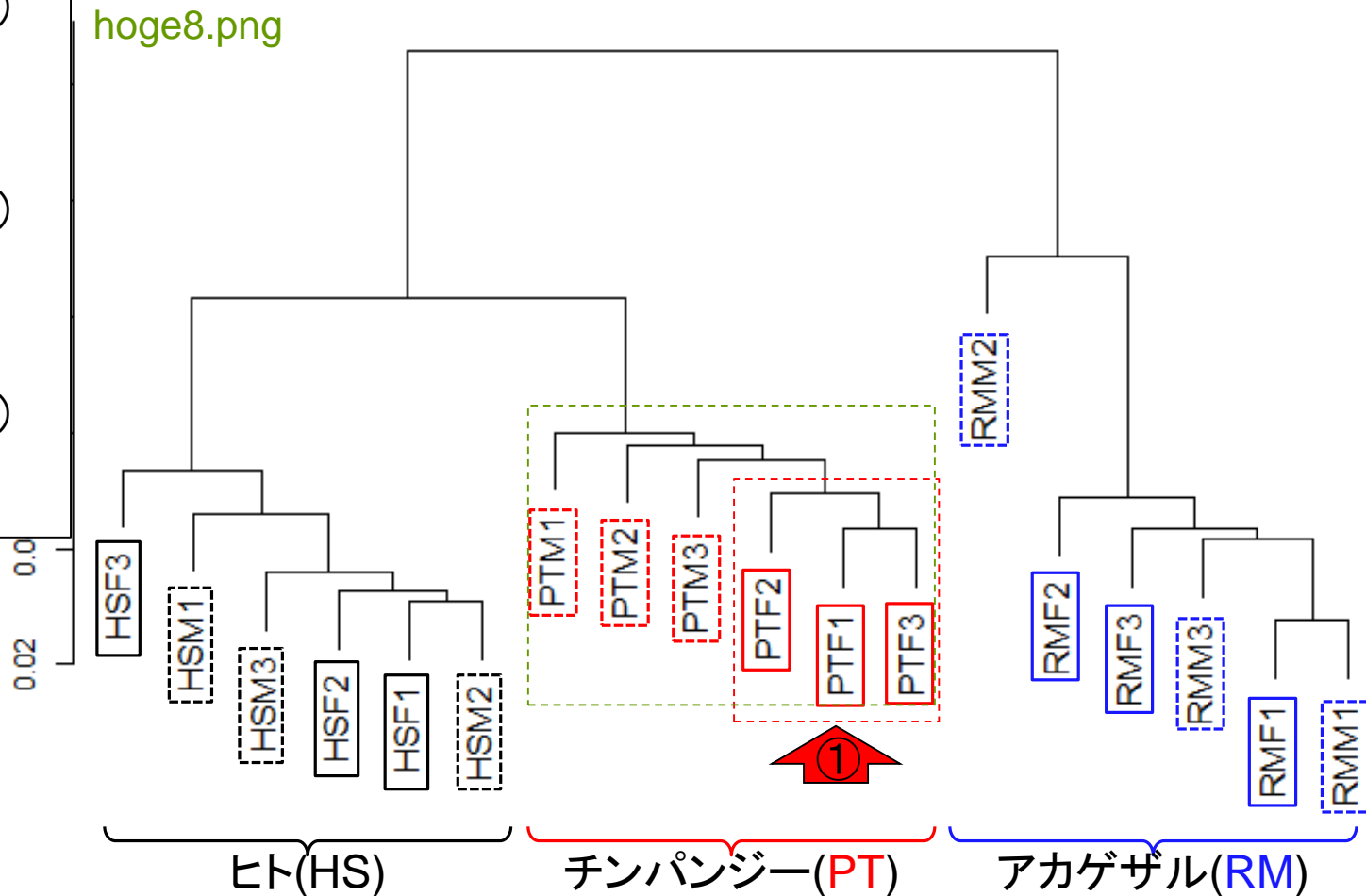




チンパンジー(PT)に限っていえば、①メス3匹がクラスターを形成しているのので、「メス群 vs. オス群」の2群間比較結果として、多少なりともDEGが検出されるだろう。

# 結果の解釈

- 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)



# この解釈の仕方は

原著論文あり。但し、TCCパッケージが提供する clusterSample 関数を用いた結果以外では責任を持ちません。また、運悪く例外データセットもあるかも…

BMC Bioinformatics. 2015 Nov 4;16:361. doi: 10.1186/s12859-015-0794-7.

## Evaluation of methods for differential expression analysis on multi-group RNA-seq count data.

Tang M<sup>1</sup>, Sun J<sup>2</sup>, Shimizu K<sup>3</sup>, Kadota K<sup>4</sup>.

### Author information

#### Abstract

**BACKGROUND:** RNA-seq is a powerful tool for measuring transcriptomes, especially for identifying differentially expressed genes or transcripts (DEGs) between sample groups. A number of methods have been developed for this task, and several evaluation studies have also been reported. However, those evaluations so far have been restricted to two-group comparisons. Accumulations of comparative studies for multi-group data are also desired.

**METHODS:** We compare 12 pipelines available in nine R packages for detecting differential expressions (DE) from multi-group RNA-seq count data, focusing on three-group data with or without replicates. We evaluate those pipelines on the basis of both simulation data and real count data.

**RESULTS:** As a result, the pipelines in the TCC package performed comparably to or better than other pipelines under various simulation scenarios. TCC implements a multi-step normalization strategy (called DEGES) that internally uses functions provided by other representative packages (edgeR, DESeq2, and so on). We found considerably different numbers of identified DEGs (18.5 ~ 45.7% of all genes) among the pipelines for the same real dataset but similar distributions of the classified expression patterns. We also found that DE results can roughly be estimated by the hierarchical dendrogram of sample clustering for the raw count data.

**CONCLUSION:** We confirmed the DEGES-based pipelines implemented in TCC performed well in a three-group comparison as well as a two-group comparison. We recommend using the DEGES-based pipeline that internally uses edgeR (here called the EEE-E pipeline) for count data with replicates (especially for small sample sizes). For data without replicates, the DEGES-based pipeline with DESeq2 (called SSS-S) can be recommended.

# 論文レベルの図です

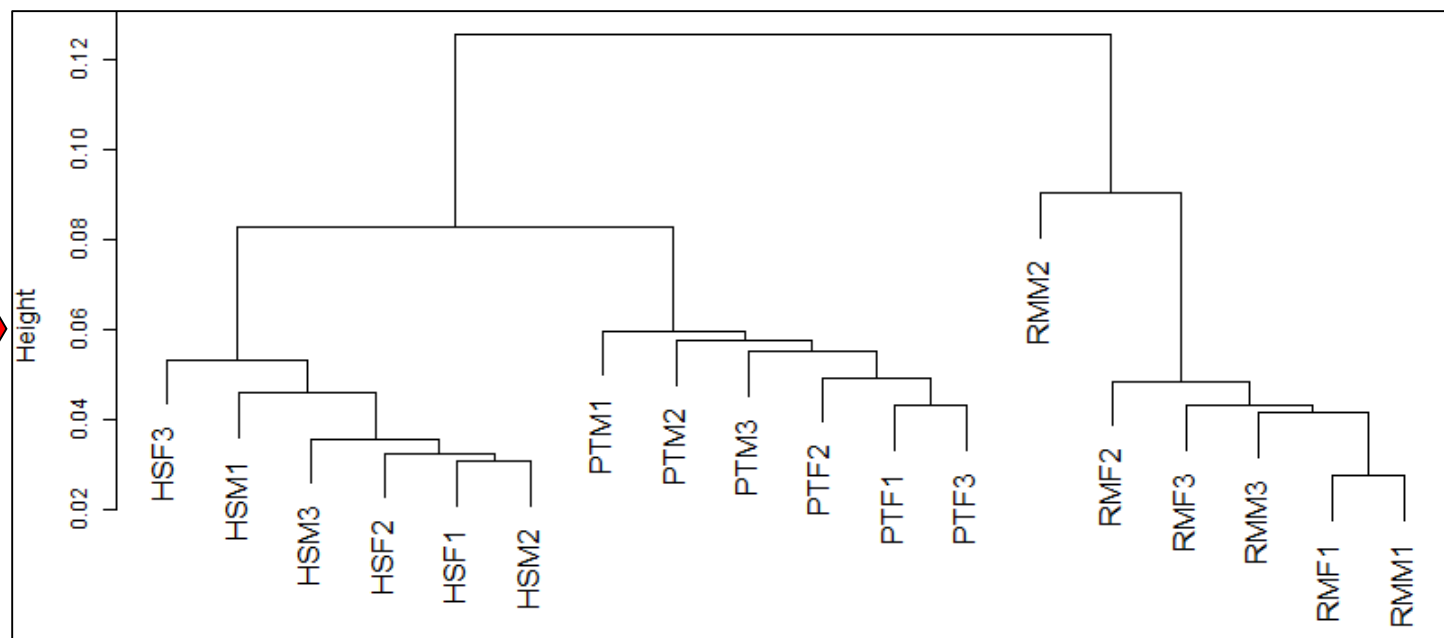
手法比較原著論文(Tang et al., 2015)の① Additional file 6に②全く同じ図もあり。つまり、コピペで作成したクラスタリング結果がそのまま論文の図に使えるということ。③書き方は赤下線のような感じでよい。

①

Additional file 6: Dendrogram of average-linkage hierarchical clustering for the Blekhman's count data. Results of sample clustering are shown: (a) a raw count dataset consisting of 36 samples, (b) a collapsed data consisting of 18 samples, and (c) the same data as (b) but with different sample labels. The clustering was performed using the "clusterSample" function with default options provided in TCC. (PPTX 62 kb)

③

②

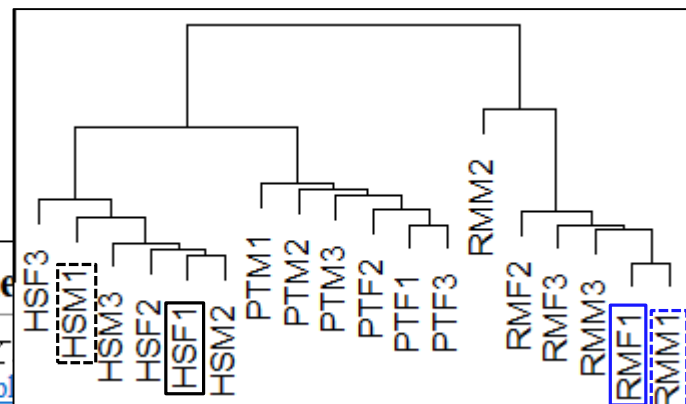


# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# HS vs. RM



## 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | Ble

[Blekhman et al., Genome Res., 2010](#)の公共カウントデータ解析に特化させてサンプルデータ42の20,689 genes×18 samplesのリアルカウントデータ ([sample](#)

ス3サンプル(HSF1-3)とオス3サンプル(HSM1-3), チンパンジー (Pan troglodytes; PT)のメス3サンプル(PTF1-3)とオス3サンプル(PTM1-3), アカゲザル (Rhesus macaque; RM)のメス3サンプル(RMF1-3)とオス3サンプル(RMM1-3)の並びになっています。つまり、以下のような感じです。FはFemale(メス)、MはMale(オス)を表します。

ヒト(1-6列目): HSF1, HSF2, HSF3, HSM1, HSM2, and HSM3

チンパンジー(7-12列目): PTF1, PTF2, PTF3, PTM1, PTM2, and PTM3

アカゲザル(13-18列目): RMF1, RMF2, RMF3, RMM1, RMM2, and RMM3

「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. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```
in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)
```

#必要なパッケージをロード

```
library(TCC)
```

```
#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
#出力ファイル名を指定してout_f2に格納
#取り扱いたいサブセット情報を指定
#G1群のサンプル数を指定
#G2群のサンプル数を指定
#DEG検出時のfalse discovery rate (FDR)閾値を指定
#ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#下、左、上、右の順で余白を指定(単位は行)
```

#パッケージの読み込み

# サブセット抽出

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

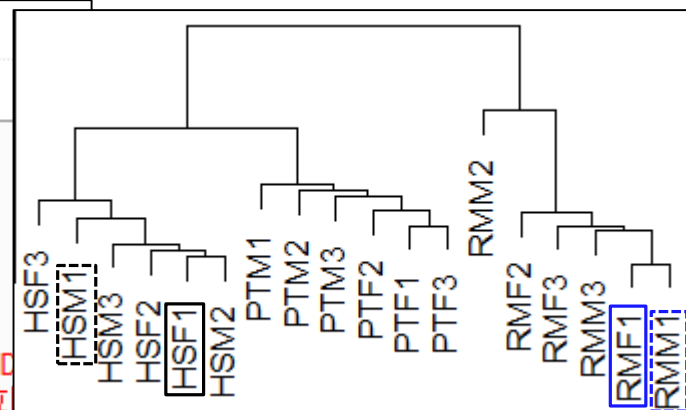
1, 4, 13, 16 列目のデータのみ抽出しています。

```
in_f <- "sample_blekman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge1.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1.png" #出力ファイル名を指定してout_f2に格納
param_subset <- c(1, 4, 13, 16) #取り扱いたいサブセット情報を指定
param_G1 <- 2 #G1群のサンプル数を指定
param_G2 <- 2 #G2群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)を指定
param_fig <- c(430, 350) #ファイル出力時の横幅と縦幅を指定(単位はpixel)
param_mar <- c(4, 4, 0, 0) #下、左、上、右の順番で余白を指定(単位はpixel)
```

```
#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")

#前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
data <- data[,param_subset] #param_subsetで指定した列のみを抽出
data.cl <- c(rep(1, param_G1), rep(2, param_G2)) #G1群を1、G2群を2で指定
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトの作成
dim(data) #行数と列数を表示
head(data) #最初の6行分を表示
```



```
R Console
> #前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
> data <- data[,param_subset] # $
> data.cl <- c(rep(1, param_G1), rep(2, param_G2)) # $
> tcc <- new("TCC", data, data.cl) # $
> dim(data) # $
[1] 20689 4
> head(data) # $
      HSF1 HSM1 RMF1 RMM1
ENSG000000000003 329 121 511 424
ENSG000000000005 0 0 0 2
ENSG000000000419 81 39 67 49
ENSG000000000457 91 114 89 117
ENSG000000000460 6 15 4 7
ENSG000000000938 44 73 73 80
>
```

# サブセット抽出

①ここで取得したいサブセットの列番号やグループ情報を指定。②発現変動解析に用いるサブセットは20,689 genes × 4 samplesのデータ。③正しくヒト(HS) vs. アカゲザル(RM)になっている

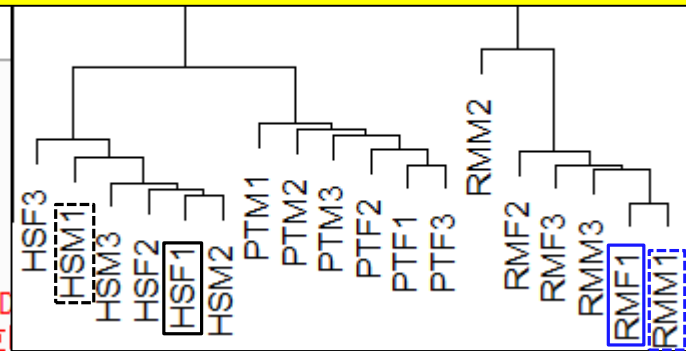
1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1

1, 4, 13, 16 列目のデータのみ抽出しています。

```
in_f <- "sample_blekman_18.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)
```



#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\_f1に格納  
 #出力ファイル名を指定してout\_f2に格納  
 #取り扱いたいサブセット情報を指定  
 #G1群のサンプル数を指定  
 #G2群のサンプル数を指定  
 #DEG検出時のfalse discovery rate (FDR)を指定  
 #ファイル出力時の横幅と縦幅を指定(単位はpx)  
 #下、左、上、右の順番で余白を指定(単位はpx)



```
#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=" ")

#前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
data <- data[,param_subset]
data.cl <- c(rep(1, param_G1), rep(2, param_G2))
tcc <- new("TCC", data, data.cl)
dim(data)
head(data)
```

#パッケージの読み込み  
 #param\_subsetで指定した列番号で抽出  
 #G1群を1、G2群を2で指定  
 #TCCクラスオブジェクトの作成  
 #行数と列数を表示  
 #最初の6行分を表示

```
R Console
> #前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
> data <- data[,param_subset]
> data.cl <- c(rep(1, param_G1), rep(2, param_G2))
> tcc <- new("TCC", data, data.cl)
> dim(data)
[1] 20689 4
> head(data)
      HSF1 HSM1 RMF1 RMM1
ENSG000000000003 329 121 511 424
ENSG000000000005 0 0 0 2
ENSG000000000419 81 39 67 49
ENSG000000000457 91 114 89 117
ENSG000000000460 6 15 4 7
ENSG000000000938 44 73 73 80
>
```



# サブセット抽出

入力ファイル(sample\_blekhman\_18.txt)を眺めるなどして、①該当サンプルの列の位置を把握していることが前提。

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```
in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)
```



```
#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_f1に格納
#出力ファイル名を指定してout_f2に格納
#取り扱いたいサブセット情報を指定
#G1群のサンプル数を指定
#G2群のサンプル数を指定
#DEG検出時のfalse discovery rate (FDR)を指定
#ファイル出力時の横幅と縦幅を指定(単位は行)
#下、左、上、右の順で余白を指定(単位は行)
```

```
#必要なパッケージをロード
library(TCC)
```

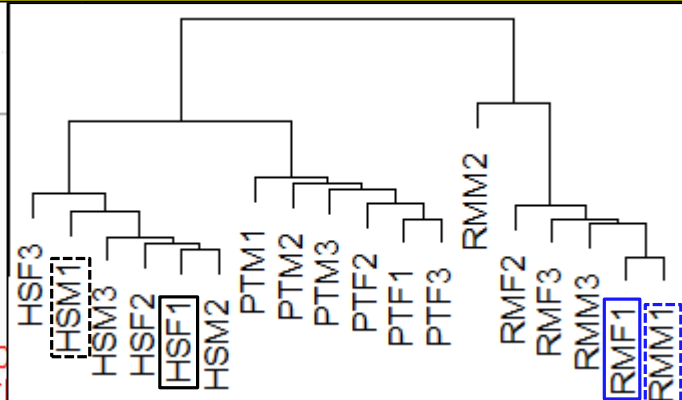
```
#パッケージの読み込み
```

```
#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
```

```
#前処理
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
data	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	
data	ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
tcc <	ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
dim(d	ENSG000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
head(c	ENSG000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
<	ENSG000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
	ENSG000000000938	44	65	210	73	43	65	84	104	76	198	31	58	73	28	54	80	34	72
	ENSG000000000971	4765	7225	3405	3600	6383	5546	5382	8331	4335	2568	5019	2653	13566	9064	18247	14236	5196	11834





# FDR

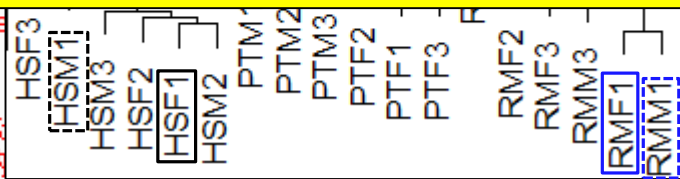
赤枠部分までをコピー。①  $q < 0.05$  を満たす遺伝子数は 2,488 個。False discovery rate (FDR) = 0.05 は、この閾値を満たす 2,488 個を発現変動遺伝子 (Differentially Expressed Genes; DEGs) とみなすと、 $2,488 * 0.05 = 124.4$  個は偽物であることを意味する。有意水準 (false positive rate; FPR) 5% と同じような位置づけであり、FDR 5% というのは、「許容する偽物 (non-DEG) 混入割合」に相当する。

1. ヒト 2 サンプル (G1 群: HSF1 と HSM1) vs. アカゲザル 2 サンプル (G2 群: RMF1 と RMM1)

1, 4, 13, 16 列目のデータのみ抽出しています。

```
#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", iteration=3, FDR=0.1, flog=2)
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalized

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edgeR", FDR=param_FDR) #DEG検出を実行した
result <- getResult(tcc, sort=FALSE) #p値などの結果をした結果をresultに格納
sum(tcc$stat$q.value < param_FDR) #FDR < param_FDR を満たす遺伝子数を表示
```



```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result) #正規化後のデータとDEG検出結果を結合
tmp <- tmp[order(tmp$rank),] #発現変動順にソート
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=FALSE)

#ファイルに保存(M-A plot)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])
par(mar=param_mar) #余白を指定
plot(tcc, FDR=param_FDR, xlim=c(-2, 17), ylim=c(-10, 10))
cex=0.9, cex.lab=1.2, #param_FDRで指定
cex.axis=1.2, main="", #param_FDRで指定
xlab="A = (log2(G2) + log2(G1))/2", #param_FDRで指定
```

```
R Console
+ iteration=3, FDR=$
TCC::INFO: Calculating normalization factors
TCC::INFO: (iDEGES pipeline : tmm - [ edgeR ]
TCC::INFO: Done.
> normalized <- getNormalizedData(tcc) # $
>
> #本番 (DEG検出)
> tcc <- estimateDE(tcc, test.method="edgeR")
TCC::INFO: Identifying DE genes using edgeR
TCC::INFO: Done.
> result <- getResult(tcc, sort=FALSE) # $
> sum(tcc$stat$q.value < param_FDR) # $
[1] 2488
>
> |
```



# FDR

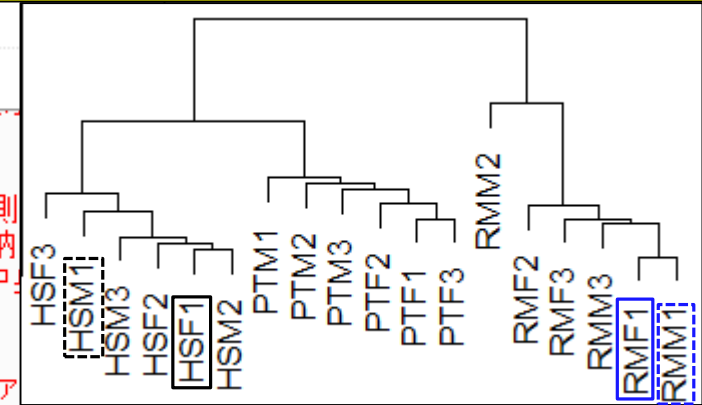
最後までコピー。①  $q < 0.30$ を満たす遺伝子数は  
 ② 4,786個。FDR = 0.30なので、 $4,786 * 0.30 = 1,435.8$ 個は偽物で残りの70%は本物だと判断する

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F)#tmpの中

#ファイルに保存(M-A plot)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
par(mar=param_mar) #余白を指定
plot(tcc, FDR=param_FDR, xlim=c(-2, 17), ylim=c(-10, 10), #param_FDRで指定
      cex=0.8, cex.lab=1.2, #param_FDRで指定
      xlab="A = (log2(G2) + log2(G1))/2", #param_FDRで指定
      ylab="M = log2(G2) - log2(G1)" #param_FDRで指定
     , legend("topright", c(paste("DEG(FDR<", param_FDR, ")"), #凡例
                             col=c("magenta", "black"), pch=20)
     , dev.off() #おまじない
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を満たす遺伝子数
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を満たす遺伝子数
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を満たす遺伝子数
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を満たす遺伝子数
```



```
R Console
+ ylab="M = log2(G2) - log2(G1)" # $
> legend("topright", c(paste("DEG(FDR<", p$
+ col=c("magenta", "black"), pch=20)
> dev.off() # $
null device
1
> sum(tcc$stat$q.value < 0.05) # $
[1] 2488
> sum(tcc$stat$q.value < 0.10) # $
[1] 3122
> sum(tcc$stat$q.value < 0.20) # $
[1] 4049
> sum(tcc$stat$q.value < 0.30) # $
[1] 4786
```

FDR閾値が比較的緩めのところを眺め、①20,689 genes中3,300個程度が本物のDEGと判断する。

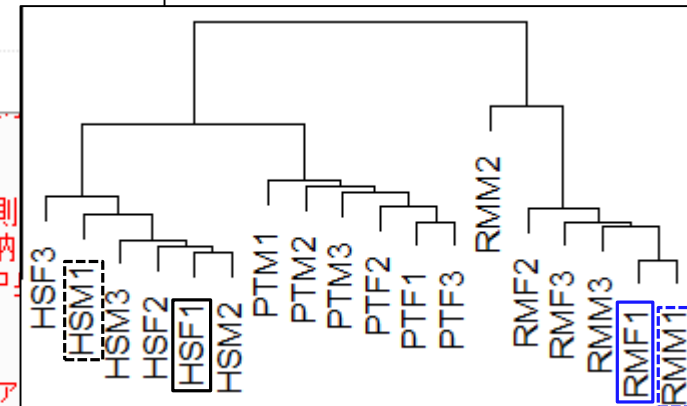
# DEG数の見積もり

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側
tmp <- tmp[order(tmp$rank),]#発現変動順にソートした結果をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F)#tmpの中

#ファイルに保存(M-A plot)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
par(mar=param_mar)#余白を指定
plot(tcc, FDR=param_FDR, xlim=c(-2, 17), ylim=c(-10, 10),#param_FDRで指定した閾値を
      cex=0.8, cex.lab=1.2,#param_FDRで指定した閾値を満たすDEGをマゼンタ
      cex.axis=1.2, main="",#param_FDRで指定した閾値を満たすDEGをマゼンタ
      xlab="A = (log2(G2) + log2(G1))/2",#param_FDRで指定した閾値を満たすDEGをマゼンタ
      ylab="M = log2(G2) - log2(G1)"#param_FDRで指定した閾値を満たすDEGをマゼンタ
      legend("topright", c(paste("DEG(FDR<", param_FDR, ")"), "non-DEG"),#凡例を
      col=c("magenta", "black"), pch=20, cex=1.2)#凡例を作成
dev.off()#おまじない
sum(tcc$stat$q.value < 0.05)#FDR < 0.05を満たすDEG数
sum(tcc$stat$q.value < 0.10)#FDR < 0.10を満たすDEG数
sum(tcc$stat$q.value < 0.20)#FDR < 0.20を満たすDEG数
sum(tcc$stat$q.value < 0.30)#FDR < 0.30を満たすDEG数
```



```
R Console
> 2488*(1 - 0.05)
[1] 2363.6
> 3122*(1 - 0.10)
[1] 2809.8
> 4049*(1 - 0.20)
[1] 3239.2
> 4786*(1 - 0.30)
[1] 3350.2
> |
```



# 樹形図と一致

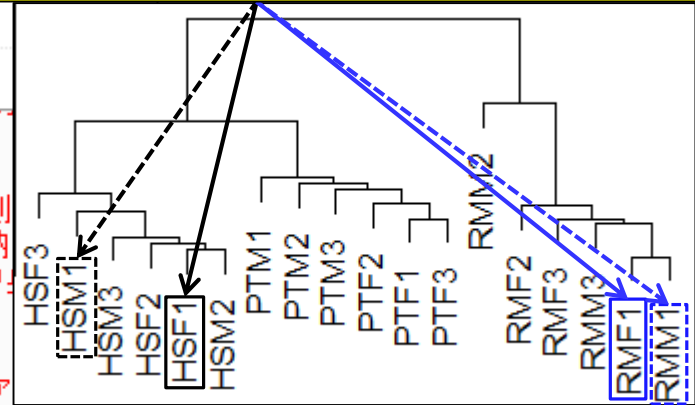
今比較しているのはHS vs. RM。クラスタリング結果からも、これらの発現プロファイルの類似度が低い(距離が遠い)ので妥当

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F)#tmpの中

#ファイルに保存(M-A plot)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
par(mar=param_mar) #余白を指定
plot(tcc, FDR=param_FDR, xlim=c(-2, 17), ylim=c(-10, 10),#param_FDRで指定した閾値
      cex=0.8, cex.lab=1.2, #param_FDRで指定した閾値を満たすDEGをマゼンタ
      cex.axis=1.2, main="", #param_FDRで指定した閾値を満たすDEGをマゼンタ
      xlab="A = (log2(G2) + log2(G1))/2",#param_FDRで指定した閾値を満たすDEGをマゼンタ
      ylab="M = log2(G2) - log2(G1)" #param_FDRで指定した閾値を満たすDEGをマゼンタ
      legend("topright", c(paste("DEG(FDR<", param_FDR, ")"), "non-DEG"),#凡例を
      col=c("magenta", "black"), pch=20, cex=1.2)#凡例を作成
dev.off() #おまじない
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を満たすDEGの数
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を満たすDEGの数
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を満たすDEGの数
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を満たすDEGの数
```



```
R Console
> 2488*(1 - 0.05)
[1] 2363.6
> 3122*(1 - 0.10)
[1] 2809.8
> 4049*(1 - 0.20)
[1] 3239.2
> 4786*(1 - 0.30)
[1] 3350.2
> |
```

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# M-A plot

これがM-A plot。発現変動遺伝子 (DEG) と判定されたものが多数存在することがわかる。param\_FDRで指定した閾値(0.05)を満たす遺伝子群がマゼンタ色で表示されている。

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とF

1, 4, 13, 16 列目のデータのみ抽出しています。

```
in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)
```

#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\_f1に格納  
 #出力ファイル名を指定してout\_f2に格納  
 #取り扱いたいサブセット情報を指定  
 #G1群のサンプル数を指定  
 #G2群のサンプル数を指定  
 #DEG検出時のfalse discovery rate (FDR)  
 #ファイル出力時の横幅と縦幅を指定(単位はピクセル)  
 #下、左、上、右の順で余白を指定(単位は行)

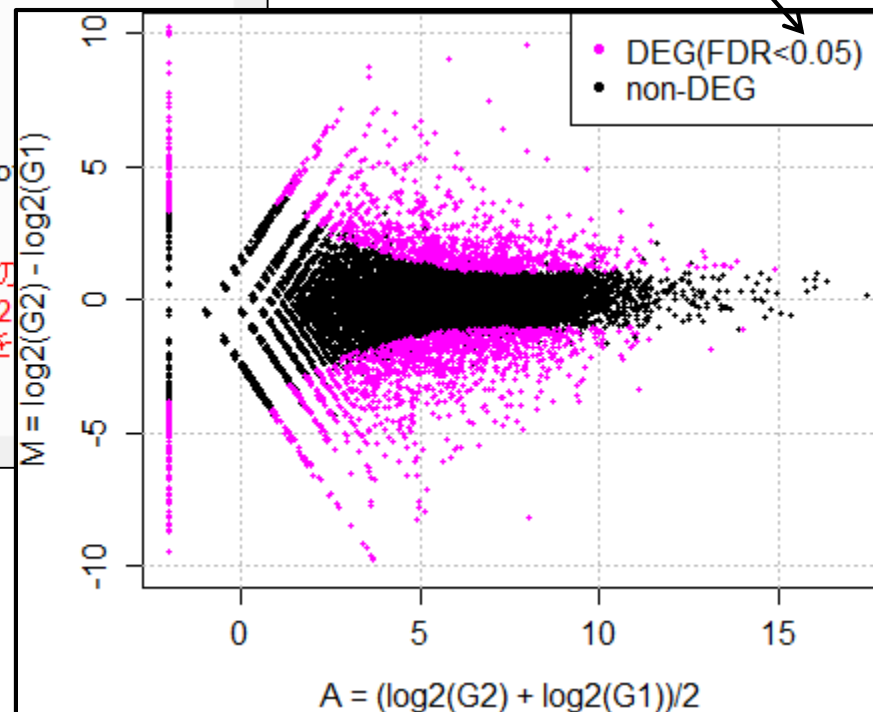
```
#必要なパッケージをロード
library(TCC)

#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", quo

#前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
data <- data[,param_subset]
data.cl <- c(rep(1, param_G1), rep(2, param_G2)) #G1群を1, G2群を2
tcc <- new("TCC", data, data.cl)
dim(data)
head(data)
```

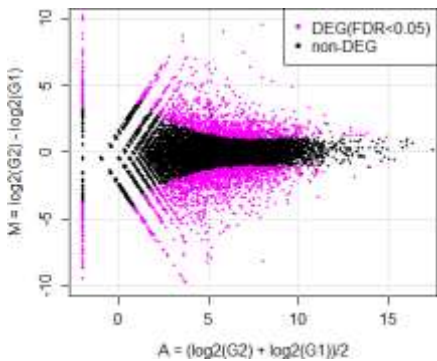
#param\_subsetで指定した列の抽出  
 #TCCクラスオブジェクトtccを作成  
 #行数と列数を表示  
 #最初の6行分を表示



DEGが存在しないデータのM-A plotを眺めることで、縦軸の閾値のみに相当する倍率変化を用いたDEG同定の危険性が分かります

# M-A plot

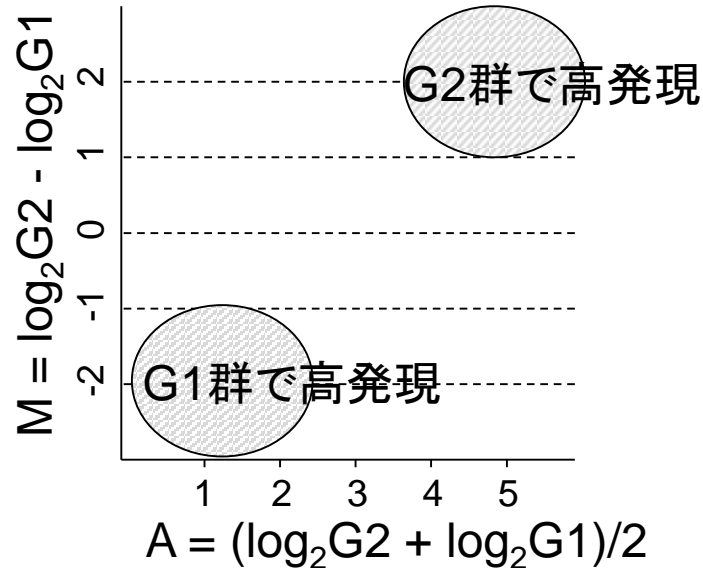
- 2群間比較用
- 横軸が全体的な発現レベル、縦軸がlog比からなるプロット
- 名前の由来は、おそらく対数の世界での縦軸が引き算 (Minus)、横軸が平均 (Average)



G1群 < G2群

G1群 = G2群

G1群 > G2群



低発現 ← 全体的に → 高発現

# DEG検出結果

1. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```

in_f <- "sample_blekman_18.txt"
out_f1 <- "hogel.txt"
out_f2 <- "hogel.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)

#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
    
```

#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\_f1に格納  
 #出力ファイル名を指定してout\_f2に格納  
 #取り扱いたいサブセット情報を指定  
 #G1群のサンプル数を指定  
 #G2群のサンプル数を指定  
 #DEG検出時のfalse discovery rate (FDR)  
 #ファイル出力時の横幅と縦幅を指定(単位はピクセル)  
 #下、左、上、右の順で余白を指定(単位は行)

#パッケージの読み込み

rownames(tcc\$count)	HSF1	HSM1	RMF1	RMM1	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000208570	0.0	0.0	1346.8	1477.0	ENSG00000208570	-2.04	11.29	4.41E-53	9.13E-49	1	1
ENSG00000220191	2.3	2.5	1394.7	1171.1	ENSG00000220191	5.79	9.06	4.54E-47	4.70E-43	2	1
ENSG00000106366	4421.9	4411.0	23.1	8.3	ENSG00000106366	8.04	-8.14	2.46E-45	1.70E-41	3	1
ENSG00000209449	0.0	0.0	644.5	713.1	ENSG00000209449	-2.04	10.23	3.29E-44	1.70E-40	4	1
ENSG00000218007	0.0	0.0	616.1	606.7	ENSG00000218007	-2.04	10.08	1.74E-43	7.21E-40	5	1
ENSG00000070985	0.0	0.0	528.2	650.8	ENSG00000070985	-2.04	10.03	4.68E-42	1.61E-38	6	1
ENSG00000209007	0.0	0.0	615.2	479.6	ENSG00000209007	-2.04	9.92	1.24E-40	3.67E-37	7	1
ENSG00000182327	367.5	363.9	0.9	0.0	ENSG00000182327	3.67	-9.69	1.52E-38	3.93E-35	8	1
ENSG00000156222	367.5	301.5	0.9	0.0	ENSG00000156222	3.61	-9.56	1.09E-36	2.51E-33	9	1
ENSG00000165272	404.8	429.0	2.6	0.9	ENSG00000165272	4.02	-7.59	5.45E-36	1.12E-32	10	1

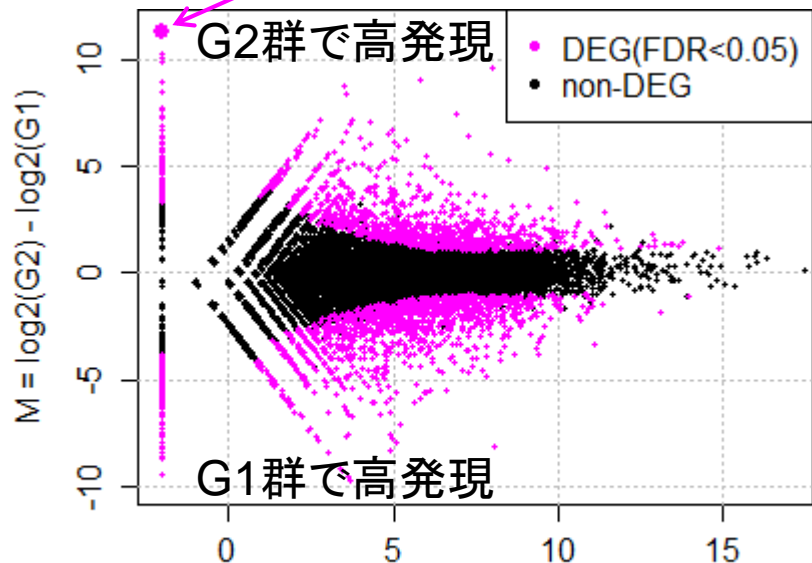


# DEG検出結果

G1(HS)群    G2(RM)群

p-valueとその順位

rownames(tcc\$count)	HSF1	HSM1	RMF1	RMM1	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000208570	0.0	0.0	1346.8	1477.0	ENSG00000208570	-2.04	11.29	4.41E-53	9.13E-49	1	1
ENSG00000220191	2.3	2.5	1394.7	1171.1	ENSG00000220191	5.79	9.06	4.54E-47	4.70E-43	2	1
ENSG00000106366	4421.9	4411.0	23.1	8.3	ENSG00000106366	8.04	-8.14	2.46E-45	1.70E-41	3	1
ENSG00000209449	0.0	0.0	644.5	713.1	ENSG00000209449	-2.04	10.23	3.29E-44	1.70E-40	4	1
ENSG00000218007	0.0	0.0	616.1	606.7	ENSG00000218007	-2.04	10.08	1.74E-43	7.21E-40	5	1
ENSG00000070985	0.0	0.0	528.2	650.8	ENSG00000070985	-2.04	10.03	4.68E-42	1.61E-38	6	1
ENSG00000209007	0.0	0.0	615.2	479.6	ENSG00000209007	-2.04	9.92	1.24E-40	3.67E-37	7	1
ENSG00000182327	367.5	363.9	0.9	0.0	ENSG00000182327	3.67	-9.69	1.52E-38	3.93E-35	8	1
ENSG00000156222	367.5	301.5	0.9	0.0	ENSG00000156222	3.61	-9.56	1.09E-36	2.51E-33	9	1
ENSG00000165272	404.9	429.0	2.6	0.9	ENSG00000165272	4.02	-7.59	5.45E-36	1.12E-32	10	1



M-A plotのA値とM値

q-value

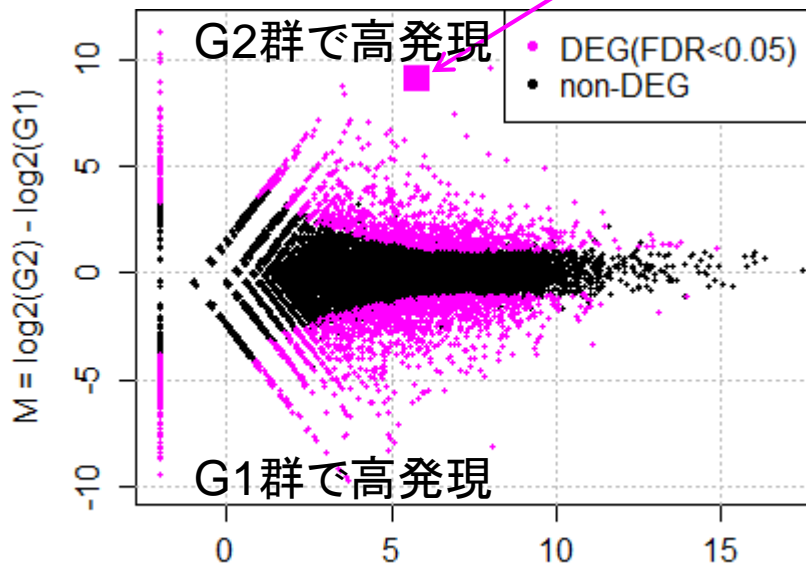
FDR閾値判定結果。q-value < 0.05を満たすDEGが1、non-DEGが0。

# DEG検出結果

G1(HS)群    G2(RM)群

p-valueとその順位

rownames(tcc\$count)	HSF1	HSM1	RMF1	RMM1	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000208570	0.0	0.0	1346.8	1477.0	ENSG00000208570	-2.04	11.29	4.41E-53	9.13E-49	1	1
ENSG00000220191	2.3	2.5	1394.7	1171.1	ENSG00000220191	5.79	9.06	4.54E-47	4.70E-43	2	1
ENSG00000106366	4421.9	4411.0	23.1	8.3	ENSG00000106366	8.04	-8.14	2.46E-45	1.70E-41	3	1
ENSG00000209449	0.0	0.0	644.5	713.1	ENSG00000209449	-2.04	10.23	3.29E-44	1.70E-40	4	1
ENSG00000218007	0.0	0.0	616.1	606.7	ENSG00000218007	-2.04	10.08	1.74E-43	7.21E-40	5	1
ENSG00000070985	0.0	0.0	528.2	650.8	ENSG00000070985	-2.04	10.03	4.68E-42	1.61E-38	6	1
ENSG00000209007	0.0	0.0	615.2	479.6	ENSG00000209007	-2.04	9.92	1.24E-40	3.67E-37	7	1
ENSG00000182327	367.5	363.9	0.9	0.0	ENSG00000182327	3.67	-9.69	1.52E-38	3.93E-35	8	1
ENSG00000156222	367.5	301.5	0.9	0.0	ENSG00000156222	3.61	-9.56	1.09E-36	2.51E-33	9	1
ENSG00000165272	404.9	429.0	2.6	0.9	ENSG00000165272	4.02	-7.59	5.45E-36	1.12E-32	10	1



M-A plotのA値とM値

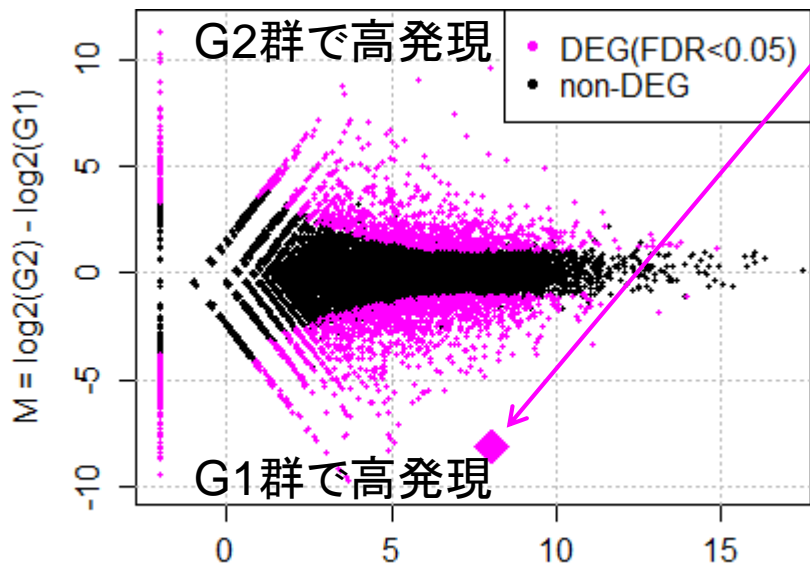
q-value

FDR閾値判定結果。q-value < 0.05を満たすDEGが1、non-DEGが0。



# DEG検出結果

rownames(tcc\$count)	G1(HS)群		G2(RM)群		gene_id	a.value	m.value	p-valueとその順位		rank	estimatedDEG
	HSF1	HSM1	RMF1	RMM1				p.value	q.value		
ENSG00000208570	0.0	0.0	1346.8	1477.0	ENSG00000208570	-2.04	11.29	4.41E-53	9.13E-49	1	1
ENSG00000220191	2.3	2.5	1394.7	1171.1	ENSG00000220191	5.79	9.06	4.54E-47	4.70E-43	2	1
ENSG00000106366	4421.9	4411.0	23.1	8.3	ENSG00000106366	8.04	-8.14	2.46E-45	1.70E-41	3	1
ENSG00000209449	0.0	0.0	644.5	713.1	ENSG00000209449	-2.04	10.23	3.29E-44	1.70E-40	4	1
ENSG00000218007	0.0	0.0	616.1	606.7	ENSG00000218007	-2.04	10.08	1.74E-43	7.21E-40	5	1
ENSG00000070985	0.0	0.0	528.2	650.8	ENSG00000070985	-2.04	10.03	4.68E-42	1.61E-38	6	1
ENSG00000209007	0.0	0.0	615.2	479.6	ENSG00000209007	-2.04	9.92	1.24E-40	3.67E-37	7	1
ENSG00000182327	367.5	363.9	0.9	0.0	ENSG00000182327	3.67	-9.69	1.52E-38	3.93E-35	8	1
ENSG00000156222	367.5	301.5	0.9	0.0	ENSG00000156222	3.61	-9.56	1.09E-36	2.51E-33	9	1
ENSG00000165272	404.9	429.0	2.6	0.9	ENSG00000165272	4.02	-7.59	5.45E-36	1.12E-32	10	1



M-A plotのA値とM値

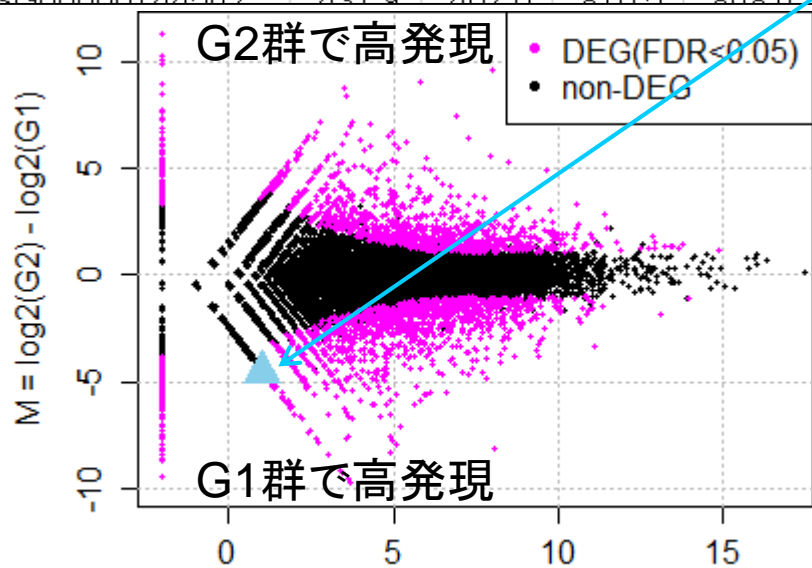
q-value

FDR閾値判定結果。q-value < 0.05  
を満たすDEGが1、non-DEGが0。

指定したFDR閾値(0.05)をギリギリ満たす2,488位の遺伝子

# DEG検出結果

rownames(tcc\$count)	G1(HS)群		G2(RM)群		gene_id	a.value	m.value	p-valueとその順位				estimatedDEG
	HSF1	HSM1	RMF1	RMM1				p.value	q.value	rank		
ENSG00000180672	9.0	8.9	0.9	1.7	ENSG00000180672	1.76	-2.82	0.00596	0.04967	2484	1	
ENSG00000159899	161.7	89.1	47.9	60.7	ENSG00000159899	6.37	-1.21	0.00597	0.0497	2485	1	
ENSG00000110442	108.5	103.1	214.0	219.4	ENSG00000110442	7.24	1.03	0.00599	0.04987	2486	1	
ENSG00000105327	5.7	24.2	1.8	2.5	ENSG00000105327	2.50	-2.80	0.006	0.04989	2487	1	
ENSG00000139445	17.0	2.5	0.0	0.8	ENSG00000139445	1.01	-4.55	0.006	0.04989	2488	1	
ENSG00000105321	61.1	128.5	14.2	47.4	ENSG00000105321	5.76	-1.62	0.00602	0.05008	2489	0	
ENSG00000118017	1.1	2.5	13.3	10.0	ENSG00000118017	2.21	2.66	0.00603	0.05009	2490	0	
ENSG00000110917	768.8	591.6	1440.9	1334.8	ENSG00000110917	9.92	1.03	0.00603	0.05011	2491	0	
ENSG00000119630	19.2	12.7	34.6	55.7	ENSG00000119630	4.75	1.50	0.00604	0.05011	2492	0	
ENSG00000144567	421.9	402.0	810.5	888.6	ENSG00000144567	9.21	1.01	0.00605	0.05019	2493	0	



M-A plotのA値とM値

q-value

FDR閾値判定結果。q-value < 0.05を満たすDEGが1、non-DEGが0。

# 様々なM-A plot

- 解析 | 発現変動 | 1について (last modified 2014/07/10)
- 解析 | 発現変動 | 2群間 | 対応なし | 1について (last modified 2015/06/02)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun\_2013) (last modified 2015/07/07)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | Blekhmanデータ | TCC(Sun\_2013) **1**
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq(Li\_2013) (last modified 2015/07/07)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR(Robinson\_2010) (last modified 2015/07/07)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | WAD(Kadota\_2008) (last modified 2015/07/07)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun\_2013) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun\_2013) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun\_2013) NEW

## 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | Blekhmanデータ | TCC(Sun\_2013) NEW

[Blekhman et al., Genome Res., 2010](#)の公共カウントデータ解析に特化させて、[TCC](#)を用いた様々な例題を示します。入力は全て [サンプルデータ42](#)の 20,689 genes×18 samplesのリアルカウントデータ ([sample blekhman 18.txt](#))です。ヒトHomo sapiens; HS)のメス3サンプル(HSF1-3)とオス3サンプル(HSM1-3), チンパンジー(Pan troglodytes; PT)のメス3サンプル(PTF1-3)とオス3サンプル(PTM1-3), アカゲザル(Rhesus macaque; RM)のメス3サンプル(RMF1-3)とオス3サンプル(RMM1-3)の並びになっています。つまり、以下のような感じです。FはFemale(メス)、MはMale(オス)を表します。

ヒト(1-6列目): HSF1, HSF2, HSF3, HSM1, HSM2, and HSM3

チンパンジー(7-12列目): PTF1, PTF2, PTF3, PTM1, PTM2, and PTM3

アカゲザル(13-18列目): RMF1, RMF2, RMF3, RMM1, RMM2, and RMM3

「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. ヒト2サンプル(G1群:HSF1とHSM1) vs. アカゲザル2サンプル(G2群:RMF1とRMM1)の場合:

1, 4, 13, 16 列目のデータのみ抽出しています。

```

in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_subset <- c(1, 4, 13, 16)
param_G1 <- 2
param_G2 <- 2
param_FDR <- 0.05
param_fig <- c(430, 350)
param_mar <- c(4, 4, 0, 0)

#必要なパッケージをロード
library(TCC)

```

#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\_f1に格納  
#出力ファイル名を指定してout\_f2に格納  
#取り扱いたいサブセット情報を指定  
#G1群のサンプル数を指定  
#G2群のサンプル数を指定  
#DEG検出時のfalse discovery rate (FDR)閾値を指定  
#ファイル出力時の横幅と縦幅を指定(単位はピクセル)  
#下、左、上、右の順で余白を指定(単位は行)

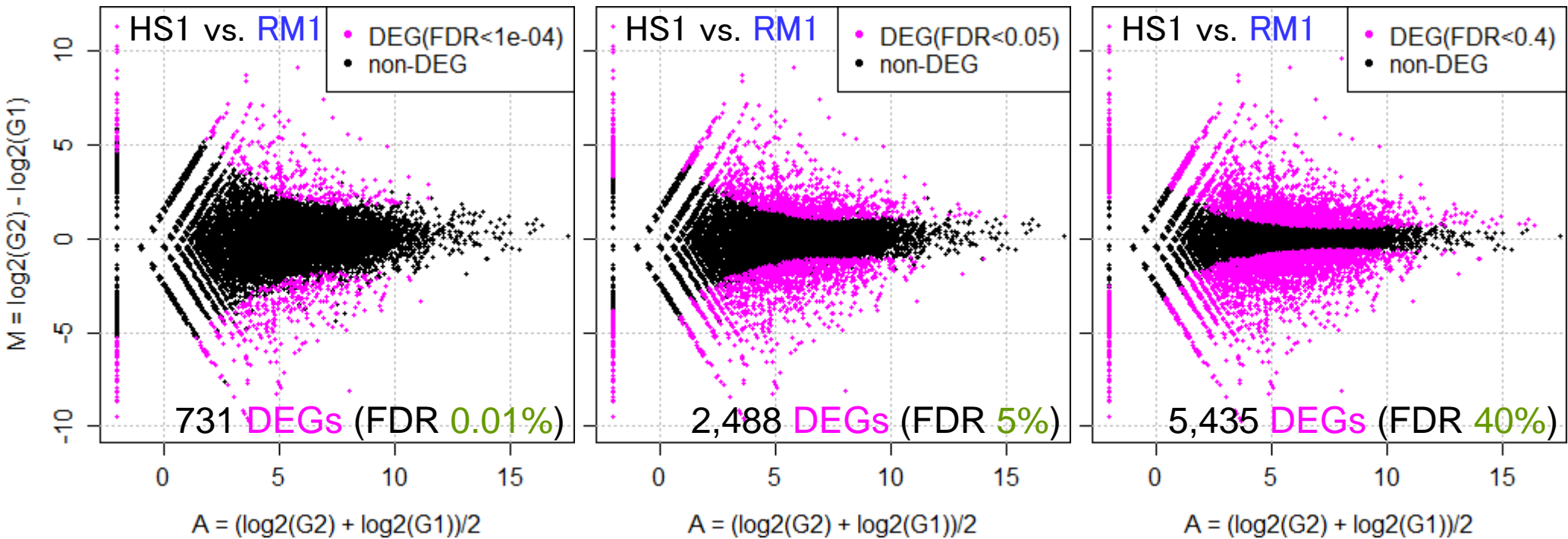
# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

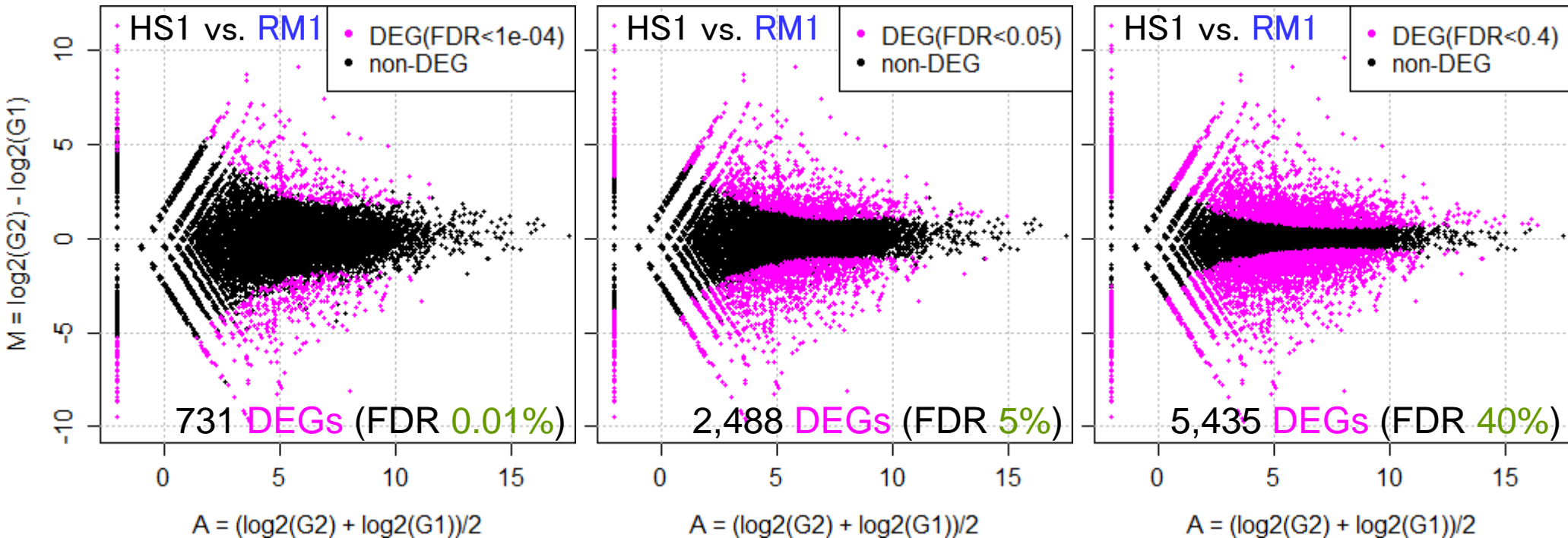
# 分布やモデル

(当たり前だが)FDR閾値を緩めると得られるDEG数は増える傾向にあることがわかる。例題6のコピペで作成。



厳しい ← FDR閾値 → 緩い  
 少ない ← DEG数 → 多い

# 分布やモデル



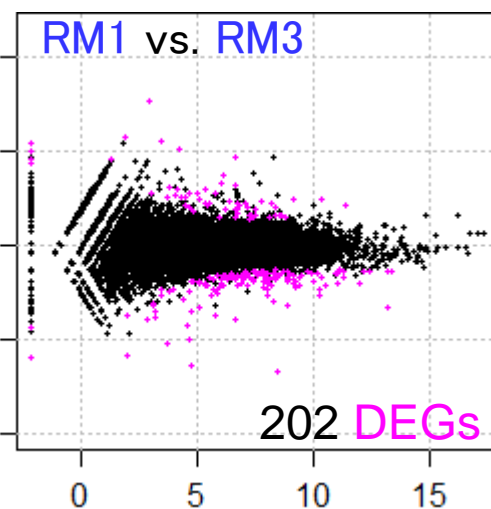
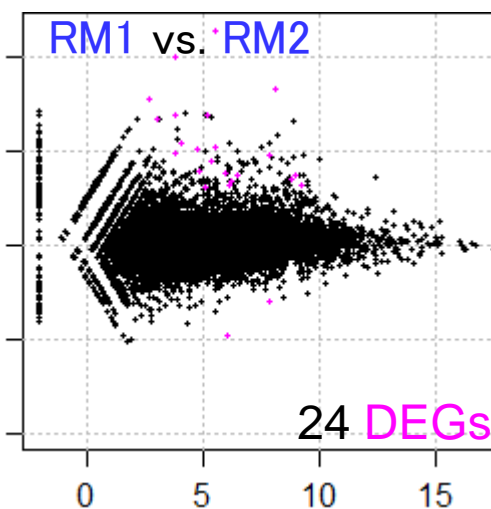
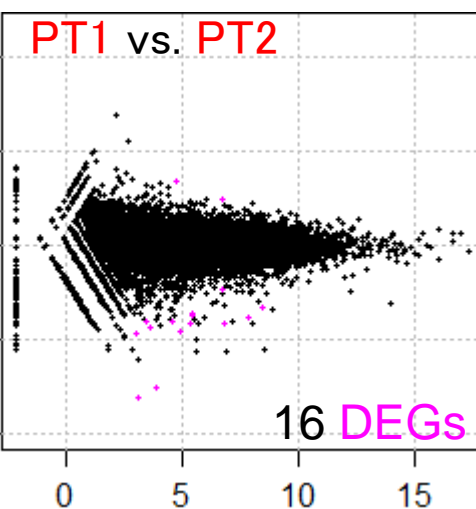
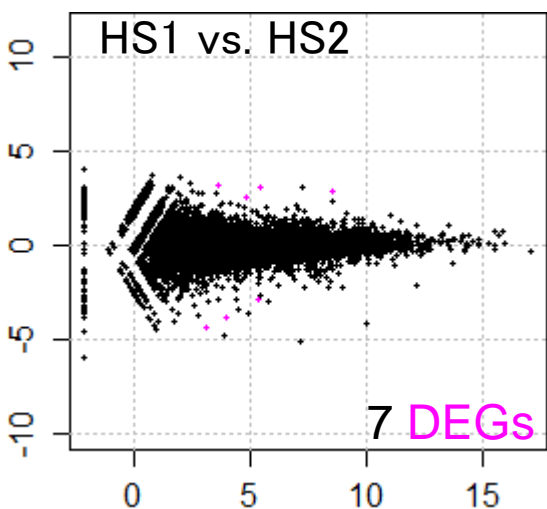
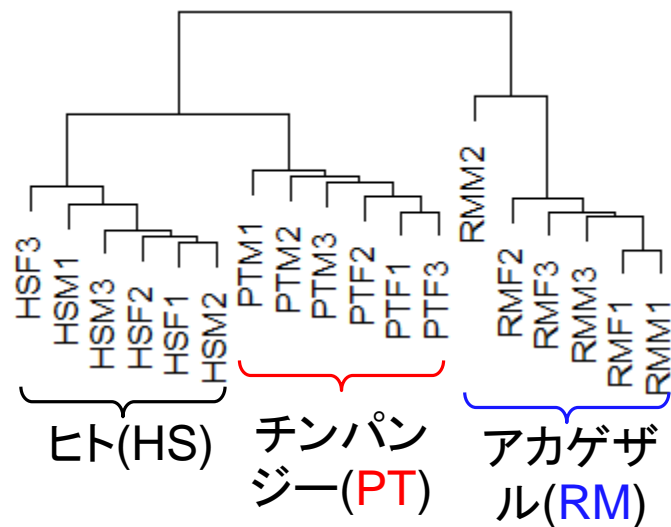
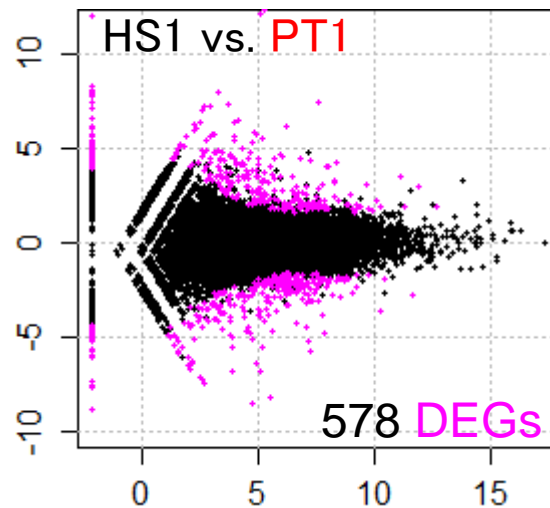
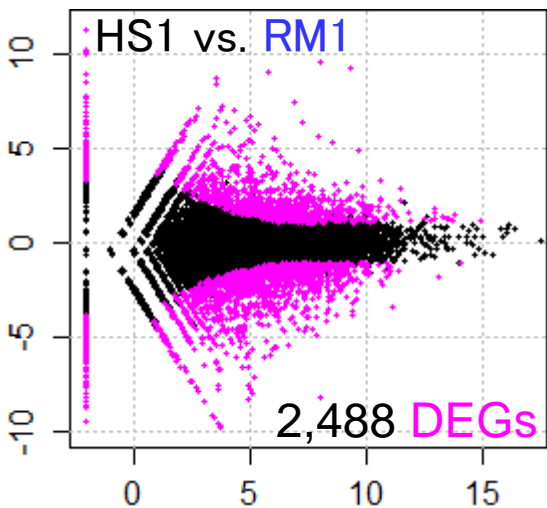
厳しい ← FDR閾値 → 緩い  
 少ない ← DEG数 → 多い





# 統計的手法とは...

同一群(下段)の分布は、異なる群(上段)の non-DEG分布とよく一致する。同一群内のばらつきの分布 (non-DEG分布) 以外のものが **DEG**と判定されるのが統計的手法の結果

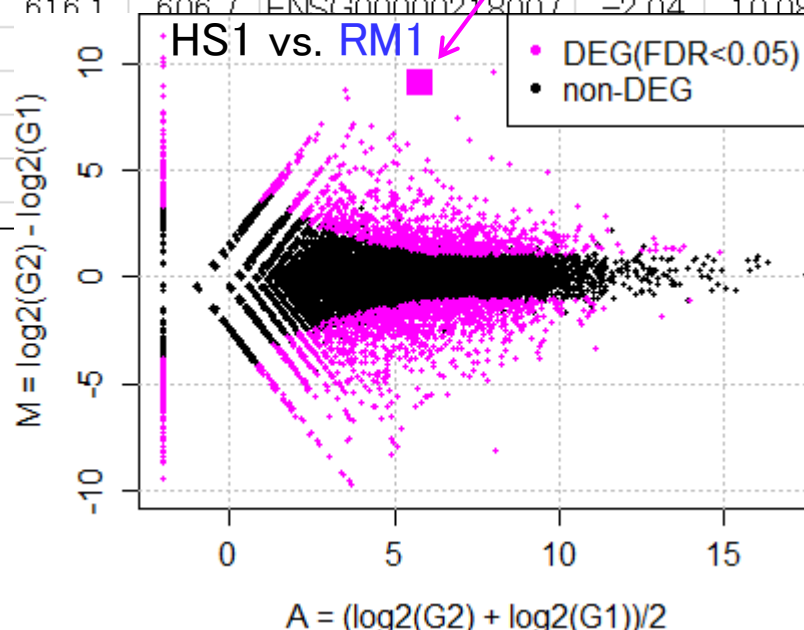


同一群内のばらつきの分布 (non-DEG分布) から遠く離れたところに位置するものは、0に近いp-value

# 統計的手法とは

- 同一群内の遺伝子のばらつきの程度を把握し、帰無仮説に従う分布の全体像を把握しておく (モデル構築)
  - non-DEGのばらつきの程度を把握しておくことと同義
- 実際に比較したい2群の遺伝子のばらつきの程度がnon-DEG分布のどのあたりに位置するかを評価 (検定)

rownames(tcc\$count)	HSF1	HSM1	RMF1	RMM1	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000208570	0.0	0.0	1346.8	1477.0	ENSG00000208570	-2.04	11.29	4.41E-53	9.13E-49	1	1
ENSG00000220191	2.3	2.5	1394.7	1171.1	ENSG00000220191	5.79	9.06	4.54E-47	4.70E-43	2	1
ENSG00000106366	4421.9	4411.0	23.1	8.3	ENSG00000106366	8.04	-8.14	2.46E-45	1.70E-41	3	1
ENSG00000209449	0.0	0.0	644.5	713.1	ENSG00000209449	-2.04	10.23	3.29E-44	1.70E-40	4	1
ENSG00000218007	0.0	0.0	616.1	606.7	ENSG00000218007	-2.04	10.08	1.74E-43	7.21E-40	5	1
ENSG00000070985	0.0	0.0						4.68E-42	1.61E-38	6	1
ENSG00000209007	0.0	0.0						1.24E-40	3.67E-37	7	1
ENSG00000182327	367.5	363.9						1.52E-38	3.93E-35	8	1
ENSG00000156222	367.5	301.5						1.09E-36	2.51E-33	9	1
ENSG00000165272	404.9	429.0						5.45E-28	1.12E-22	10	1

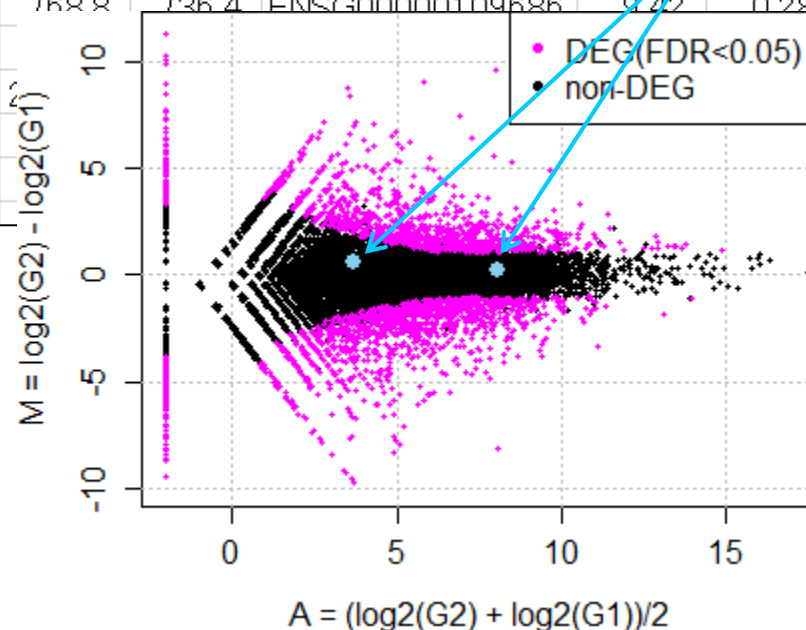


同一群内のばらつきの分布 (non-DEG分布) の、ど真ん中に位置するものは、1に近いp-value

# 統計的手法とは

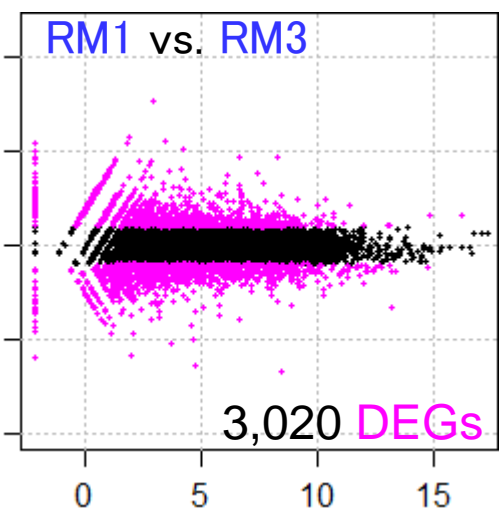
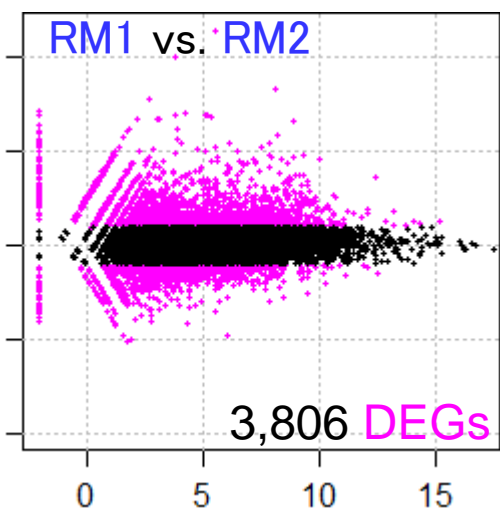
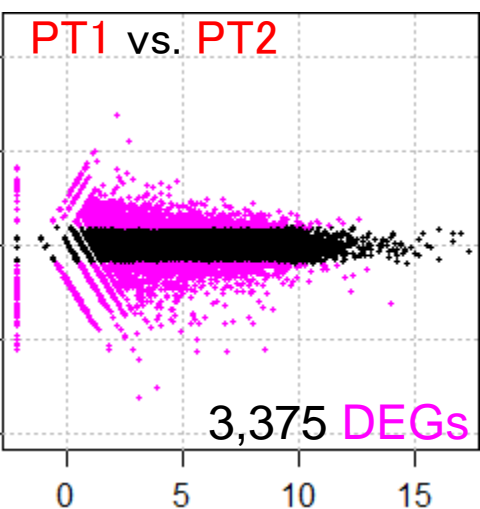
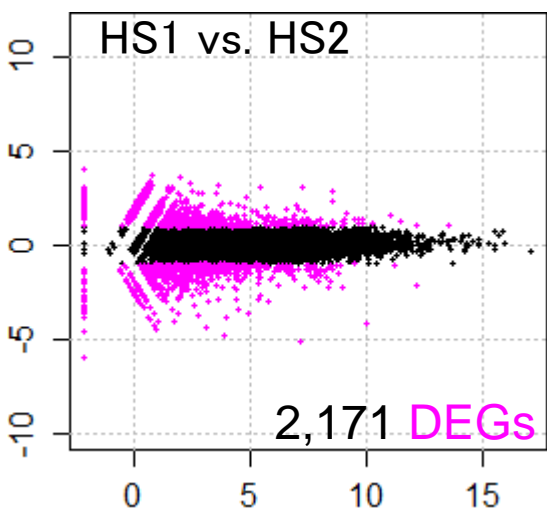
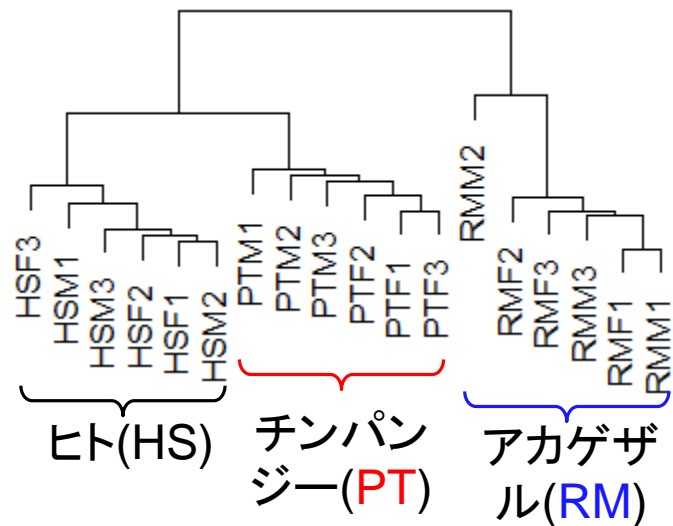
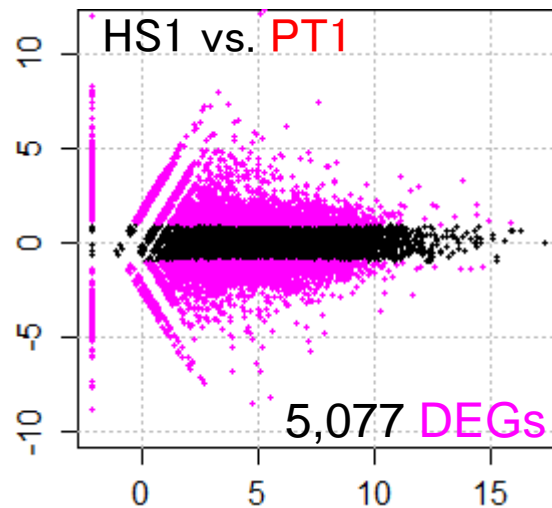
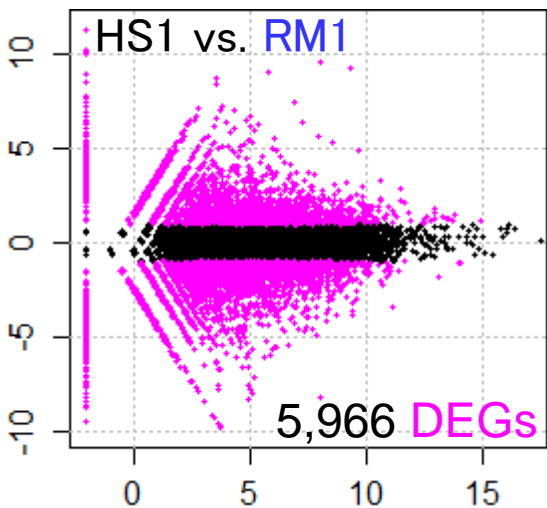
- 同一群内の遺伝子のばらつきの程度を把握し、帰無仮説に従う分布の全体像を把握しておく(モデル構築)
  - non-DEGのばらつきの程度を把握しておくことと同義
- 実際に比較したい2群の遺伝子のばらつきの程度がnon-DEG分布のどのあたりに位置するかを評価(検定)

rownames(tcc\$count)	HSF1	HSM1	RMF1	RMM1	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000047578	69.0	131.0	98.5	148.8	ENSG00000047578	6.80	0.31	0.4906	1	9727	0
ENSG00000115325	3.4	17.8	16.9	14.1	ENSG00000115325	3.68	0.55	0.49087	1	9728	0
ENSG00000122257	256.7	234.1	253.9	334.1	ENSG00000122257	8.07	0.26	0.49092	1	9729	0
ENSG00000090861	603.8	339.7	542.4	601.8	ENSG00000090861	9.02	0.28	0.491	1	9730	0
ENSG00000109686	451.1	792.6	768.8	736.4	ENSG00000109686	9.42	0.28	0.49109	1	9731	0
ENSG00000032389	53.1	36.9						0.49115	1	9732	0
ENSG00000125844	2299.7	3137.5						0.49127	1	9733	0
ENSG00000180190	52.0	28.0						0.4913	1	9734	0
ENSG00000100351	2.3	12.7						0.49134	1	9735	0
ENSG00000160554	72.5	122.6						0.49139	1	9736	0



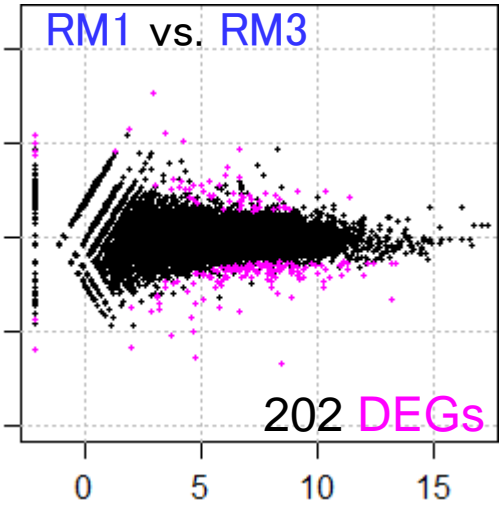
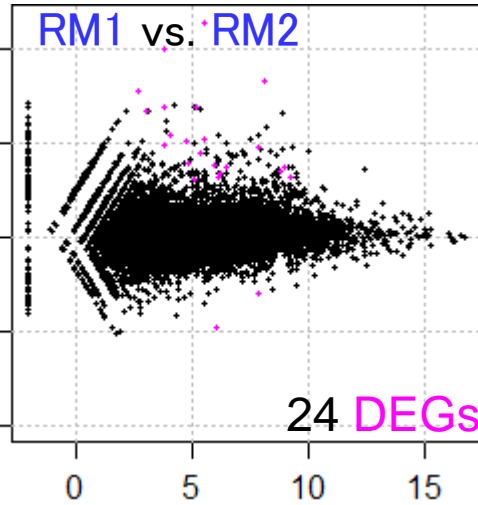
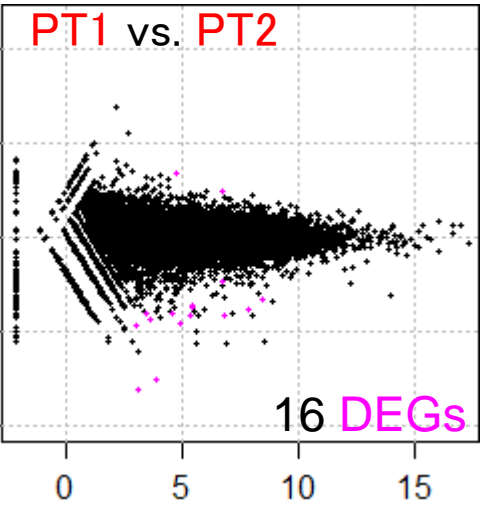
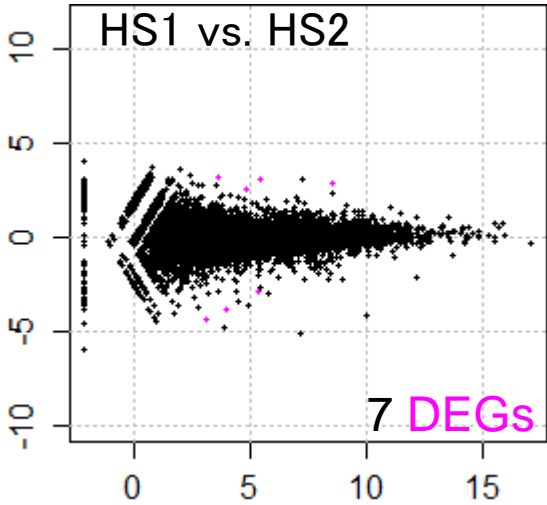
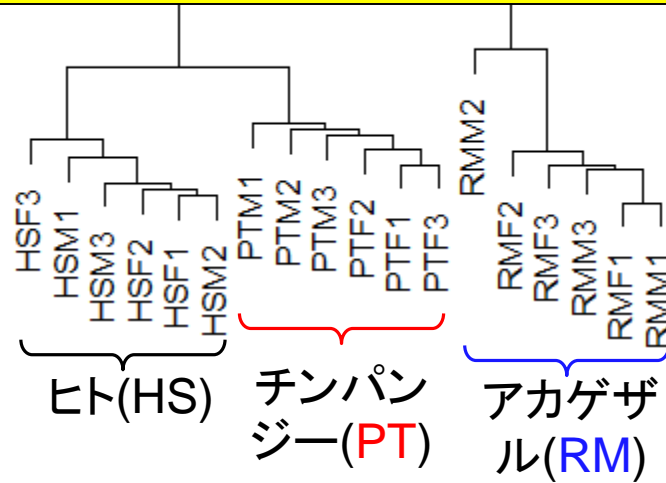
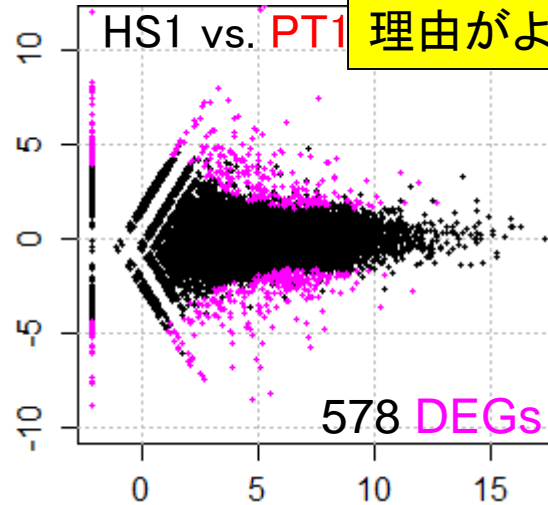
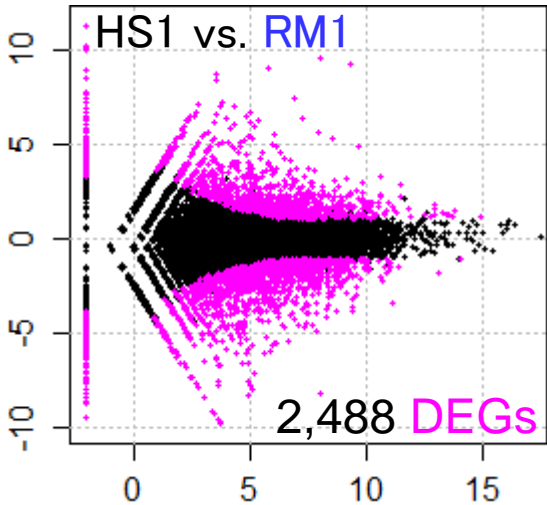
# 結果の比較(2倍変化)

倍率変化(fold-change; FC)でのDEG検出結果。下段の同一群内比較でも多数の偽陽性が検出されている。例題13をベースに作成



統計的手法(TCC)も多少偽陽性が存在するが、倍率変化(FC)ほど凶悪ではないことがわかる。また高発現側のDEGは、FCと比較的よく一致していることがわかる。先人がFCのみで比較的信頼性の高い結果を得てきた理由がよくわかる(高発現側を信頼するという経験則)。

# 結果の比較(FDR)



# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# 3群間比較

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)。どこかの群間で発現変動している遺伝子を検出するやり方を示す。

	ヒト ( <i>Homo sapiens</i> ; HS)						チンパンジー ( <i>Pan troglodytes</i> ; PT)						アカゲザル ( <i>Rhesus macaque</i> ; RM)					
	メス(Female)			オス(Male)			メス			オス			メス			オス		
	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3
ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
ENSG000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
ENSG000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
ENSG000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
ENSG000000000938	44	65	210	73	43	65	84	104	76	198	31	58	73	28	54	80	34	72
ENSG000000000971	4765	7225	3405	3600	6383	5546	5382	8331	4335	2568	5019	2653	13566	9964	18247	14236	5196	11834
ENSG000000001036	297	251	189	200	234	249	305	301	313	254	151	331	292	106	379	201	88	140
ENSG000000001084	630	737	306	336	984	459	417	328	885	298	569	218	1062	786	1110	873	664	1752
ENSG000000001167	36	30	36	29	33	28	63	80	25	69	74	41	62	34	108	97	35	61
ENSG000000001460	3	1	5	1	4	2	0	1	1	1	1	3	1	1	1	0	1	3
ENSG000000001461	49	37	34	28	62	32	75	69	40	90	69	60	210	92	176	247	81	117
ENSG000000001487	117	93	88	80	131	110	125	98	75	108	130	131	138	95	187	137	158	172

20,689 genes

# 3群間比較論文

3群間比較用に特化した手法選択のガイドライン。①反復ありデータの場合は(内部的にedgeRの関数を用いた)TCC、②反復なしの場合は(内部的にDESeq2を用いた)TCCがおススメ。

BMC Bioinformatics. 2015 Nov 4;16:361. doi: 10.1186/s12859-015-0794-7.

## Evaluation of methods for differential expression analysis on multi-group RNA-seq count data.

Tang M<sup>1</sup>, Sun J<sup>2</sup>, Shimizu K<sup>3</sup>, Kadota K<sup>4</sup>.

### Author information

### Abstract

**BACKGROUND:** RNA-seq is a powerful tool for measuring transcriptomes, especially for identifying differentially expressed genes or transcripts (DEGs) between sample groups. A number of methods have been developed for this task, and several evaluation studies have also been reported. However, those evaluations so far have been restricted to two-group comparisons. Accumulations of comparative studies for multi-group data are also desired.

**METHODS:** We compare 12 pipelines available in nine R packages for detecting differential expressions (DE) from multi-group RNA-seq count data, focusing on three-group data with or without replicates. We evaluate those pipelines on the basis of both simulation data and real count data.

**RESULTS:** As a result, the pipelines in the TCC package performed comparably to or better than other pipelines under various simulation scenarios. TCC implements a multi-step normalization strategy (called DEGES) that internally uses functions provided by other representative packages (edgeR, DESeq2, and so on). We found considerably different numbers of identified DEGs (18.5 ~ 45.7% of all genes) among the pipelines for the same real dataset but similar distributions of the classified expression patterns. We also found that DE results can roughly be estimated by the hierarchical dendrogram of sample clustering for the raw count data.

**CONCLUSION:** We confirmed the DEGES-based pipelines implemented in TCC performed well in a three-group comparison as well as a two-group comparison. We recommend using the DEGES-based pipeline that internally uses edgeR (here called the EEE-E pipeline) for count data with replicates (especially for small sample sizes). For data without replicates, the DEGES-based pipeline with DESeq2 (called SSS-S) can be recommended.

①

②



# データ正規化周辺

- RPM (Mortazavi *et al.*, *Nat. Methods*, **5**: 621-628, 2008)
  - RPKM(Reads per kilobase of exon per million mapped reads)の長さ補正を行わないバージョン
  - Reads Per Million mapped readsの略。
- TMM正規化 (Robinson and Oshlack, *Genome Biol.*, **11**: R25, 2010)
  - Trimmed Mean of M valuesの略。edgeRパッケージに実装されている。
  - 発現変動遺伝子(DEG)のデータ正規化時の悪影響を排除すべく、M-A plot上で周縁部にあるデータを使わずに(トリムして)正規化係数を決定する方法。
- TbT正規化 (Kadota *et al.*, *Algorithms Mol. Biol.*, **7**: 5, 2012)
  - TMM法の改良版で、TMM-baySeq-TMMという3ステップで正規化を行う方法。
  - 1st stepで得られたTMM正規化係数を用いて、2nd step (baySeq)でDEG同定を行い、3rd step (TMM)ではDEGを排除した残りのデータでTMM正規化。DEGの影響を排除しつつもできるだけ多くのnon-DEGデータを用いて頑健に正規化係数を決めるという思想(DEG elimination strategy提唱論文)。
- iDEGES正規化 (Sun *et al.*, *BMC Bioinformatics*, **14**: 219, 2013)
  - TCCパッケージの原著論文
  - DEG elimination strategy (DEGES) を一般化し、より高速且つ頑健にしたもの。TbTは「複製あり」のデータのみに対応していなかったが、「複製なし」データにも対応。
  - iDEGES/edgeR正規化法:「複製あり」データ正規化用。TMM-(edgeR-TMM)<sub>n</sub>パイプライン
  - iDEGES/DESeq正規化法:「複製なし」データ正規化用。DESeq-(DESeq-DESeq)<sub>n</sub>パイプライン

# TbT正規化法

- TCCパッケージに実装している基本コンセプトの原著論文
- 本来の目的である発現変動遺伝子(DEG)自体がデータ正規化時に悪影響を与えるのでDEG候補を除去して正規化を行うほうがよいこと(DEG Elimination Strategy)を提唱した論文。既存の正規化法は、比較するグループ間でDEG数に偏りが無い(unbiased DE)場合にはうまく正規化できるが、偏りがある場合(biased DE)にはうまく正規化できないことを示した。
- TbT法の実体は、①edgeRパッケージ中のTMM正規化法実行、②baySeqパッケージ中のDEG検出法実行、および③DEG候補を除去した残りのnon-DEG候補のみを用いたTMM正規化法実行、の3ステップを基本とするTMM-baySeq-TMMパイプライン。出力は正規化後の結果(正確には正規化係数)なので、TbT正規化後に任意のDEG検出法を適用することで一連の発現変動解析が終了することになる。例えばTbT正規化法実行後にedgeR中のDEG検出法を適用する一連の手順はTMM-baySeq-TMM-edgeRに相当し、原著論文中ではedgeR/TbTと略記している。論文中ではTbTにした理由を論理的に書いたが、本音は”ToT”に近いものということでTMMとbaySeqを採用。
- 提案したマルチステップの正規化パイプラインは、第2および第3ステップを繰り返して実行することでより頑健な正規化を実現可能であることも示している。これが図3で説明しているiterative TbT approachに相当するものであり、TMM-(baySeq-TMM) $n$ とも表現できる。例えばiterative TbT正規化法実行後にedgeR中のDEG検出法を適用する一連の手順はTMM-(baySeq-TMM) $n$ -edgeRに相当する。 $n = 0$ の場合はTMM-edgeRとなり、これはedgeRパッケージ中のオリジナルの手順と同じである。

# TCC

- TbT論文の考え方を一般化し、Rパッケージとしてまとめたという論文
- TbTはDEG Elimination Strategy (DEGES; でげす)に基づく一つの正規化パイプラインにすぎないこと、第2ステップのbaySeqによるDEG同定ステップが律速であり高速化が課題であったこと、そして各ステップにおいて他の方法が原理的に適用可能であることなどを述べている。
- 第2ステップのDEG同定法をedgeR中のものに置き換えると、TMM-edgeR-TMMという正規化パイプラインになる。これは、全てedgeRパッケージ中の関数のみで成立するため、DEGES/edgeRと略記している。また、DEGES正規化後にedgeR中のDEG同定法を適用する一連の解析手順は「DEGES/edgeR-edgeR」または「TMM-edgeR-TMM-edgeR」と表記できる。これは実質的にedgeRパッケージ中のオリジナルの解析手順を2回繰り返して行っていることと同義である(ただし、第3ステップのTMMは検出されたDEG候補以外のデータで実行される)。それが、実質的に「TCCは例えばiterative edgeRという理解でよい」と主張する根拠である。
- TbT論文中で言及したiterative TbTに相当するものは、この論文中ではiterative DEGES (略してiDEGES)と称している。例えば、「iDEGES/edgeR-edgeR」はTMM-(edgeR-TMM) $n$ -edgeRに相当する。 $n=1$ は「DEGES/edgeR-edgeR」に相当する。 $n$ が2以上の場合がiDEGESに相当するが、 $n$ の数を増やしてもその分計算コストがかかる一方で、実質的に $n=3$ 程度で頭打ちになることを論文中で示している。それゆえ、iterative DEGESのデフォルトは $n=3$ としている。compcodeR (Soneson, C., Bioinformatics, 2014)中でもデフォルトはそうなっている。

# 3群間比較

sample\_blekhman\_18.txtを入力として、ヒト(HS)6サンプル、チンパンジー(PT)6サンプル、アカゲザル(RM)6サンプルの3群間比較を行います。どこかの群間で発現変動している遺伝子を検出するやり方です。各群のサンプルは全て別個体です。例えばヒトの場合は6人分のデータ(6 biological replicates)であり、1人のサンプルを6個に分割したデータ(6 technical replicates)ではありません。

	ヒト ( <i>Homo sapiens</i> ; HS)						チンパンジー ( <i>Pan troglodytes</i> ; PT)						アカゲザル ( <i>Rhesus macaque</i> ; RM)					
	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3
ENSG000000000003	329	300	168	121	421	359	574	429	386	409	685	428	511	464	480	424	1348	705
ENSG000000000005	0	0	0	0	1	0	1	4	1	0	1	1	0	1	2	2	0	0
ENSG000000000419	81	61	56	39	78	62	100	66	65	59	58	93	67	72	57	49	82	90
ENSG000000000457	91	62	76	114	73	95	131	229	87	274	239	149	89	69	118	117	114	163
ENSG000000000460	6	17	12	15	7	17	8	8	5	12	7	10	4	4	10	7	3	4
ENSG000000000938	44	65	210	73	43	65	84	104	76	198	31	58	73	28	54	80	34	72
ENSG000000000971	4765	7225	3405	3600	6383	5546	5382	8331	4335	2568	5019	2653	13566	9964	18247	14236	5196	11834
ENSG000000001036	297	251	189	200	234	249	305	301	313	254	151	331	292	106	379	201	88	140
ENSG000000001084	630	737	306	336	984	459	417	328	885	298	569	218	1062	786	1110	873	664	1752
ENSG000000001167	36	30	36	29	33	28	63	80	25	69	74	41	62	34	108	97	35	61
ENSG000000001460	3	1	5	1	4	2	0	1	1	1	1	3	1	1	1	0	1	3
ENSG000000001461	49	37	34	28	62	32	75	69	40	90	69	60	210	92	176	247	81	117
ENSG000000001487	117	93	88	80	131	110	125	98	75	108	130	131	138	95	187	137	158	172

20,689 genes

②例題7。③入力はsample\_blekhman\_18.txt、出力はhoge7.txtのみ。M-A plotはない。

# 3群間比較

- 解析 | 発現変動 | 2群間 | 対応あり | 複製なし | [DESeq\(Anders\\_2010\)](#) (last modified 2014/03/14)
- 解析 | 発現変動 | 2群間 | 対応あり | 複製なし | [edgeR\(Robinson\\_2010\)](#) (last modified 2014/01/07)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/05)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [DESeq2\(Love\\_2014\)](#) (last modified 2015/02/04)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/05) 推奨
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [EBSeq\(Leng\\_2015\)](#) (last modified 2016/02/16) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [SAMseq\(Li\\_2013\)](#) (last modified 2015/02/10)

## 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | TCC(Sun\_2013)

TCCを用いたやり方を示します。内部的にiDEGES/edgeR(Sun\_2013)正規化を実行したのち、edgeRパッケージ中のGLMLRT法で発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文でのiDEGES/edgeR-edgeRという解析パイプラインに相当します。TCC原著論文(Sun et al., *BMC Bioinformatics*, 2013)では3群間複製ありデータ用の推奨パイプラインを示していませんでしたが、多群間比較用の推奨ガイドライン提唱論文(Tang et al., *BMC Bioinformatics*, 2015)で推奨しているパイプライン"EEE-E"が"iDEGES/edgeR-edgeR"と同じものです。この2つの論文を引用し、安心してご利用ください。尚、ここでやっていることはANOVAのような「どこかの群間で発現に差がある」遺伝子を検出します。

### 7. サンプルデータ42のリアルデータ(sample\_blekhman\_18.txt)の場合:

#### 1. サンプルデータ15

シミュレーションデータ  
DEG (gene\_1~gene\_3000)がG3群で

```
in_f <- "data_15.txt"
out_f <- "hoge7.txt"
param_G1 <- 3
param_G2 <- 3
param_G3 <- 3
param_FDR <- 0.05
```

#必要なパッケージ

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al., *Genome Res.*, 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
in_f <- "sample_blekhman_18.txt"
out_f <- "hoge7.txt"
```

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

```
param_G1 <- 6
param_G2 <- 6
param_G3 <- 6
param_FDR <- 0.05
```

#G1群のサンプル数を指定  
#G2群のサンプル数を指定  
#G3群のサンプル数を指定

#DEG検出時のfalse discovery rate (FDR)閾値を指定

```
#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
```

```
c(Sun et al., BMC Bioinformatics, 14: 219, 2013
```

```
# Tang et al., BMC Bioinformatics, 16: 361, 2015
```

```
, quote="")#in_fで指定した
```

# 3群間比較

①ここで各群のサンプル数(列数)を指定。②FDR 閾値を指定するところだが、出力ファイルの読み取り方が分かっていたら気にしなくてもよい。

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al., *Genome Res.*, 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```

in_f <- "sample_blekhman_18.txt"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"                  #出力ファイル名を指定してout_fに格納
param_G1 <- 6                         #G1群のサンプル数を指定
param_G2 <- 6                         #G2群のサンプル数を指定
param_G3 <- 6                         #G3群のサンプル数を指定
param_FDR <- 0.05                     #DEG検出時のfalse discovery rate (FDR)閾値を指定

#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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.cl)      #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edgeR",
                       iteration=3, FDR=0.1, floorPDEG=0.05)#正規化
normalized <- getNormalizedData(tcc)  #正規化後のデータを取り出し
    
```

```

R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="blekh")
[1] "sample_blekhman_18.txt"
> |
    
```

Sun et al., *BMC Bioinformatics*, 14: 219, 2013

Tang et al., *BMC Bioinformatics*, 16: 361, 2015

# 3群間比較

コピーで実行した結果の最後の部分を表示。約2分。①赤枠部分は、4種類のFDR閾値を満たす遺伝子数を表示している。例えば②は5%の偽物混入を許容すると7,247個がDEGと判定されるということ。

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome Res., 2010の 20,689 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較で

```
#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3)) #G1群を1、G2群を2、G3群を3、G4群を4、G5群を5、G6群を6、G7群を7、G8群を8、G9群を9、G10群を10、G11群を11、G12群を12、G13群を13、G14群を14、G15群を15、G16群を16、G17群を17、G18群を18、G19群を19、G20群を20、G21群を21、G22群を22、G23群を23、G24群を24、G25群を25、G26群を26、G27群を27、G28群を28、G29群を29、G30群を30、G31群を31、G32群を32、G33群を33、G34群を34、G35群を35、G36群を36、G37群を37、G38群を38、G39群を39、G40群を40、G41群を41、G42群を42)
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", iteration=3, FDR=0.1)
normalized <- getNormalizedData(tcc) #正規化

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edger", FDR=param_FDR)
result <- getResult(tcc, sort=FALSE) #p値などの結果

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)
tmp <- tmp[order(tmp$rank),] #発現変動順に$
write.table(tmp, out_f, sep="\t", append=F, quote=F, as.is=T)
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を$
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を$
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を$
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を$
```

```
R Console
> tcc <- estimateDE(tcc, test.method="edger", FDR=param_FDR)
TCC::INFO: Identifying DE genes using edger ...
TCC::INFO: Done.
> result <- getResult(tcc, sort=FALSE) #p値などの結果$
>
> #ファイルに保存(テキストファイル)
> tmp <- cbind(rownames(tcc$count), normalized, result)$
> tmp <- tmp[order(tmp$rank),] #発現変動順に$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, as.is=T)
> sum(tcc$stat$q.value < 0.05) #FDR < 0.05を$
[1] 7247
> sum(tcc$stat$q.value < 0.10) #FDR < 0.10を$
[1] 8487
> sum(tcc$stat$q.value < 0.20) #FDR < 0.20を$
[1] 10063
> sum(tcc$stat$q.value < 0.30) #FDR < 0.30を$
[1] 11469
>
```



# DEG数の見積もり

FDR閾値が比較的緩めのところを眺め、①  
20,689 genes中8,000個程度がどこかの群間で発現変動している本物のDEGと判断する

## 7. サンプルデータ42のリアルデータ(sample blekhan 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome Res., 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
#前処理(TCCクラスオブジェクトの作成)
data.c1 <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3)) #G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.c1) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", iteration=3, FDR=0.1)
normalized <- getNormalizedData(tcc) #正規化

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edgeR", test.p.value=0.05)
result <- getResult(tcc, sort=FALSE) #p値を降順でソート

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized)
tmp <- tmp[order(tmp$rank),] #発現変動の大きい順でソート
write.table(tmp, out_f, sep="\t", append=F, as.is=T)
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を$
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を$
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を$
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を$
```

```
R Console
> write.table(tmp, out_f, sep="\t", append=F, quote=F, $
> sum(tcc$stat$q.value < 0.05) #FDR < 0.05を$
[1] 7247
> sum(tcc$stat$q.value < 0.10) #FDR < 0.10を$
[1] 8487
> sum(tcc$stat$q.value < 0.20) #FDR < 0.20を$
[1] 10063
> sum(tcc$stat$q.value < 0.30) #FDR < 0.30を$
[1] 11469
> 7247*(1 - 0.05)
[1] 6884.65
> 8487*(1 - 0.10)
[1] 7638.3
> 10063*(1 - 0.20)
[1] 8050.4
> 11469*(1 - 0.30)
[1] 8028.3
> |
```





# 出力ファイル解説

①出力ファイル(hoge7.txt)の中身を解説。②正規化後のデータ、③統計解析結果(p値、q値、順位情報など)。④発現変動順にソートされた結果

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome Res. 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)閾値を指定
```



rownames(tc)	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDE
	ENSG00000208587	1	1	4	0	0	0	###	###	###	###	###	###	5	2	1	4	9	4	ENSNANA	NA	NA	1.2E-148	2.4E-144	1
ENSG00000209449	0	0	1	0	4	1	1	1	0	2	2	0	657	857	378	718	###	551	ENSNANA	NA	NA	2.4E-141	2.5E-137	2	1
ENSG00000157399	446	390	297	302	660	462	2	6	3	5	4	2	1	0	0	0	0	2	ENSNANA	NA	NA	3.0E-131	2.0E-127	3	1
ENSG00000209007	0	0	0	0	0	1	1	0	0	0	4	0	627	880	405	483	###	440	ENSNANA	NA	NA	6.5E-125	3.4E-121	4	1
ENSG00000134201	16	7	18	3	15	12	0	1	0	0	0	0	###	###	###	###	###	###	ENSNANA	NA	NA	6.5E-115	2.7E-111	5	1
ENSG00000145244	1	0	1	0	2	3	0	3	1	0	1	0	216	363	355	263	266	174	ENSNANA	NA	NA	2.6E-113	8.8E-110	6	1
ENSG00000208570	0	0	0	0	0	0	###	###	###	###	###	###	###	###	###	###	###	790	ENSNANA	NA	NA	2.4E-106	7.1E-103	7	1
ENSG00000220191	2	5	1	3	6	11	2	5	0	6	10	11	###	918	892	###	935	658	ENSNANA	NA	NA	2.0E-105	5.3E-102	8	1
ENSG00000208978	98	173	118	192	66	142	10	19	6	16	3	1	###	###	###	###	###	###	ENSNANA	NA	NA	2.0E-100	4.5E-97	9	1
ENSG00000208070	0	0	0	0	0	0	0	0	0	0	0	0	190	186	98	225	205	113	ENSNANA	NA	NA	8.6E-99	1.8E-95	10	1



正規化後のデータを取り出す部分の説明。①データ正規化(正確には正規化係数を得た)後のtccオブジェクトを入力として、②getNormalizedData関数を用いて正規化後のデータを取得した結果をnormalizedオブジェクトに格納し、③出力の一部として組み込んでいる。

# コードの解説

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genom samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(R)

```
#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edgeR", #正規化を実行した結果をtccに格納
                      iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtccに格納
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edgeR", FDR=param_FDR)#DEG検出を実行した結果をtccに格納
result <- getResult(tcc, sort=FALSE) #p値などの結果をした結果をresultに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側にDEG検出結果を格納
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果を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に保存
```

rownames(tcc)	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDE
ENSG00000208587	1	1	4	0	0	0	###	###	###	###	###	###	5	2	1	4	9	4	ENSNANA	NA	NA	1.2E-148	2.4E-144	1	1
ENSG00000209449	0	0	1	0	4	1	1	1	0	2	2	0	657	857	378	718	###	551	ENSNANA	NA	NA	2.4E-141	2.5E-137	2	1
ENSG00000157399	446	390	297	302	660	462	2	6	3	5	4	2	1	0	0	0	0	2	ENSNANA	NA	NA	3.0E-131	2.0E-127	3	1
ENSG00000209007	0	0	0	0	0	1	1	0	0	0	4	0	627	880	405	483	###	440	ENSNANA	NA	NA	6.5E-125	3.4E-121	4	1
ENSG00000134201	16	7	18	3	15	12	0	1	0	0	0	0	###	###	###	###	###	###	ENSNANA	NA	NA	6.5E-115	2.7E-111	5	1



(このウェブページではお約束的にそうしているので)  
 ①1番左側に行名をしているが、実は②のresultオブジェクトの③1番左側が同じ情報なのでなくてもよい

# コードの解説

## 7. サンプルデータ42のリアルデータ(sample blekman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekman et al. Genome Res., 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3、G4群を4、G5群を5、G6群を6、G7群を7、G8群を8、G9群を9、G10群を10、G11群を11、G12群を12、G13群を13、G14群を14、G15群を15、G16群を16、G17群を17、G18群を18
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger", #正規化を実行した結果をtccに格納
iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtccに格納
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edger", FDR=param_FDR)#DEG検出を実行した結果をtccに格納
result <- getResult(tcc, sort=FALSE) #p値などの結果をした結果をresultに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側にDEG検出結果を格納
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果を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に保存
```



rownames(tc	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDE
ENSG00000208587	1	1	4	0	0	0	###	###	###	###	###	###	5	2	1	4	9	4	ENSNA	NA	NA	1.2E-148	2.4E-144	1	1
ENSG00000209449	0	0	1	0	4	1	1	1	0	2	2	0	657	857	378	718	###	551	ENSNA	NA	NA	2.4E-141	2.5E-137	2	1
ENSG00000157399	446	390	297	302	660	462	2	6	3	5	4	2	1	0	0	0	0	2	ENSNA	NA	NA	3.0E-131	2.0E-127	3	1
ENSG00000209007	0	0	0	0	0	1	1	0	0	0	4	0	627	880	405	483	###	440	ENSNA	NA	NA	6.5E-125	3.4E-121	4	1
ENSG00000134201	16	7	18	3	15	12	0	1	0	0	0	0	###	###	###	###	###	###	ENSNA	NA	NA	6.5E-115	2.7E-111	5	1







①の段階では、まだ発現変動順にはソートされておらず、cbind関数を用いて列方向で結合した(column bind)結果をtmpオブジェクトに格納しているだけ。実際、②resultオブジェクトは入力ファイルの遺伝子名の並びになっている。

# コードの解説

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)

```
#前処理(TCCクラスオブジェクトの作成)
data.c1 <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.c1) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger", #正規化を実行した結果をtcc1
iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtcc1
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edger", FDR=param_F)
result <- getResult(tcc, sort=FALSE) #p値などの結果をresultに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)
tmp <- tmp[order(tmp$rank),] #発現変動順にソート
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=FALSE)
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を満たす遺伝子の数
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を満たす遺伝子の数
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を満たす遺伝子の数
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を満たす遺伝子の数
```

```
R Console
> head(result)
      gene_id a.value m.value      p.value
1 ENSG00000000003      NA      NA 0.006355395
2 ENSG00000000005      NA      NA 0.102190127
3 ENSG000000000419     NA      NA 0.907278882
4 ENSG000000000457     NA      NA 0.001334159
5 ENSG000000000460     NA      NA 0.002882158
6 ENSG000000000938     NA      NA 0.075320458
      q.value rank estimatedDEG
1 0.021425251 6137             1
2 0.208132657 10158            0
3 1.000000000 17497            0
4 0.005606829 4923             1
5 0.010911831 5463             1
6 0.162323433 9600             0
> |
```



(赤字のコメント部分をよく見ればわかるがw)①発現変動順にソートしている部分がココ。tmp[order(x), ]は、xの並びで行をソートするお約束的な書き方。発現変動順にソートしたくない場合は、行頭に#をつけてコメントアウトすればよい。

# コードの解説

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)

```

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger",#正規化を実行した結果をtccに格納
iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtccに格納
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edger", FDR=param_FDR)#DEG検出を実行した結果をtccに格納
result <- getResult(tcc, sort=FALSE) #p値などの結果をした結果をresultに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側にDEG検出結果を格納
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果をtmpに格納
write.table(tmp, out_f, sep=" ", append=F, quote=F, row.names=F)#tmpの中身を指定したファイルに保存
sum(tcc$stat$q.value < 0.05) #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を満たす遺伝子数を表示

```







# コードの解説

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome Res., 2010の 20,689 genes×18 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3、G4群を4、G5群を5、G6群を6、G7群を7、G8群を8、G9群を9、G10群を10、G11群を11、G12群を12、G13群を13、G14群を14、G15群を15、G16群を16、G17群を17、G18群を18
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger", #正規化を実行した結果をtccに格納
iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtccに格納
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納

#本番(DEG検出)
tcc <- estimateDE(tcc, test.method="edger", FDR=param_FDR)#DEG検出を実行した結果をtccに格納
result <- getResult(tcc, sort=FALSE) #p値などの結果をした結果をresultに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, result)#正規化後のデータの右側にDEG検出結果を格納
tmp <- tmp[order(tmp$rank),] #発現変動順にソートした結果を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に保存

sum(tcc$stat$q.value < 0.05) #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10) #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20) #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30) #FDR < 0.30を満たす遺伝子数を表示
```



	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDE
ENSG00000208587	1	1	4	0	0	0	###	###
ENSG00000209449	0	0	1	0	4	1	1	0
ENSG00000157399	446	390	297	302	660	462	2	6
ENSG00000209007	0	0	0	0	0	1	1	0
ENSG00000134201	16	7	18	3	15	12	0	1



# コードの解説

①1番右側のestimatedDEGという名前の列がq-value (q値) < 0.05のところから0に切り替わっているのは、②param\_FDRのところから0.05を指定していたから。②の指定部分を気にしなくてよいと最初解説したのは、q-value列の取り扱い方法を理解していればそれで充分だからです。

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。[Blekhman et al., Genome Res., 2010](#)の 20,689 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較で

```

in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)閾値を指定

#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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G3群を3
tcc <- new("TCC", data, data.cl) #TCCクラスオブジェクトtccを作成

#本番(正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger", #正規化を実行した結果をtcc1
iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した結果をtcc1
normalized <- getNormalizedData(tcc) #正規化後のデータを取り出してnormalizedに格納
    
```



ENSG00000178662	6	1	1	0	1	0	0	1	3	2	12	1	6	6	11	6	1	12	EN	NA	NA	0.0175	0.049905	7245	1
ENSG00000165105	1	6	18	6	0	20	84	15	26	6	4	10	23	16	57	23	62	23	EN	NA	NA	0.0175	0.049935	7246	1
ENSG00000006576	43	33	37	52	33	33	15	41	24	20	19	9	64	54	48	33	18	29	EN	NA	NA	0.0175	0.049963	7247	1
ENSG00000126460	17	11	10	19	17	10	11	9	28	27	17	12	26	30	25	23	26	33	EN	NA	NA	0.0176	0.050148	7248	0
ENSG00000117691	165	161	217	111	201	186	303	180	212	404	213	259	191	196	156	135	219	93	EN	NA	NA	0.0176	0.050173	7249	0
ENSG00000105339	11	9	101	46	20	15	77	138	27	185	34	28	22	16	15	16	92	15	EN	NA	NA	0.0176	0.050335	7250	0

# パターン分類...

(どの群間で違いがあるかは問わずに)どこかの群間で発現に差があるものを検出する枠組み(ANOVA的な解析)なのでこのような結果になる...のはしょうがないとして。G1(HS)群で高発現のものがx個、G2(PT)群で高発現のものがy個、みたいなものが欲しい!

## 7. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

5.と基本的に同じで、入力ファイルが違うだけです。Blekhman et al. Genome Res. 20 samplesのカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物

```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してi
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)閾値を指定
```

rownames(tc	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDE
ENSG00000208587	1	1	4	0	0	0	###	###	###	###	###	###	5	2	1	4	9	4	ENSN	NA	NA	1.2E-148	2.4E-144	1	1
ENSG00000209449	0	0	1	0	4	1	1	1	0	2	2	0	657	857	378	718	###	551	ENSN	NA	NA	2.4E-141	2.5E-137	2	1
ENSG00000157399	446	390	297	302	660	462	2	6	3	5	4	2	1	0	0	0	0	2	ENSN	NA	NA	3.0E-131	2.0E-127	3	1
ENSG00000209007	0	0	0	0	0	1	1	0	0	0	4	0	627	880	405	483	###	440	ENSN	NA	NA	6.5E-125	3.4E-121	4	1
ENSG00000134201	16	7	18	3	15	12	0	1	0	0	0	0	###	###	###	###	###	###	ENSN	NA	NA	6.5E-115	2.7E-111	5	1
ENSG00000145244	1	0	1	0	2	3	0	3	1	0	1	0	216	363	355	263	266	174	ENSN	NA	NA	2.6E-113	8.8E-110	6	1
ENSG00000208570	0	0	0	0	0	0	###	###	###	###	###	###	###	###	###	###	###	790	ENSN	NA	NA	2.4E-106	7.1E-103	7	1
ENSG00000220191	2	5	1	3	6	11	2	5	0	6	10	11	###	918	892	###	935	658	ENSN	NA	NA	2.0E-105	5.3E-102	8	1
ENSG00000208978	98	173	118	192	66	142	10	19	6	16	3	1	###	###	###	###	###	###	ENSN	NA	NA	2.0E-100	4.5E-97	9	1
ENSG00000208070	0	0	0	0	0	0	0	0	0	0	0	0	190	186	98	225	205	113	ENSN	NA	NA	8.6E-99	1.8E-95	10	1

# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)

# 遺伝子クラスタリング

(個人的には結論として非推奨だが) 遺伝子間クラスタリングを行っておき、どの遺伝子がどのパターンに属するかという情報を  
①MBCluster.Seqなどを用いて得ておき、特定のFDR閾値を満たす遺伝子サブセットの分類分けを行えばよい。おそらくこれも王道なので一応紹介。

[Bioinformatics](#). 2014 Jan 15;30(2):197-205. doi: 10.1093/bioinformatics/btt632. Epub 2013

## Model-based clustering for RNA-seq data.

Si Y<sup>1</sup>, Liu P, Li P, Brutnell TP.

### Author information

### Abstract

**MOTIVATION:** RNA-seq technology has been widely adopted as an attractive alternative to microarray-based methods to study global gene expression. However, robust statistical tools to analyze these complex datasets are still lacking. By grouping genes with similar expression profiles across treatments, cluster analysis provides insight into gene functions and networks, and hence is an important technique for RNA-seq data analysis.

**RESULTS:** In this manuscript, we derive clustering algorithms based on appropriate probability models for RNA-seq data. An expectation-maximization algorithm and another two stochastic versions of expectation-maximization algorithms are described. In addition, a strategy for initialization based on likelihood is proposed to improve the clustering algorithms. Moreover, we present a model-based hybrid-hierarchical clustering method to generate a tree structure that allows visualization of relationships among clusters as well as flexibility of choosing the number of clusters. Results from both simulation studies and analysis of a maize RNA-seq dataset show that our proposed methods provide better clustering results than alternative methods such as the K-means algorithm and hierarchical clustering methods that are not based on probability models.



**AVAILABILITY AND IMPLEMENTATION:** An R package, [MBCluster.Seq](#), has been developed to implement our proposed algorithms. This R package provides fast computation and is publicly available at <http://www.r-project.org>

# 遺伝子クラスタリング

①MBCluster.Seq単体での利用はこちら。②例題4。③K-meansクラスタリングの一種なので、③クラスター数を指定する。50など比較的大きめの値にしても、non-redundantにしてくれるので、最終的に得られるクラスター数は(データの性質にもよるが通常は)減る。

- ・ 解析 | クラスタリング | について (last modified 2016/02/12) **NEW**
- ・ 解析 | クラスタリング | サンプル間 | [hclust](#) (last modified 2015/02/26)
- ・ 解析 | クラスタリング | サンプル間 | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/15)
- ・ 解析 | クラスタリング | 遺伝子間(基礎) | [MBCluster.Seq\(Si\\_2014\)](#) (last modified 2016/02/12)
- ・ 解析 | クラスタリング | 遺伝子間(応用) | [MBCluster.Seq\(Si\\_2014\)+k-means++\(Sun\\_2013\)](#) (last modified 2016/02/12)
- ・ 解析 | シミュレーション | カウントデータ | について (last modified 2015/11/10)

## 解析 | クラスタリング | 遺伝子間(基礎) | MBCluster.Seq(Si\_2014)

[MBCluster.Seq](#)パッケージを用いたやり方を示します。k-means++(Arthur and Vassilvitskii, 2007)と似た方法でクラスター中心を決める方法を内部的に利用しているらしいです。param\_clust\_numで指定するクラスター数は最初は気持ち多めにしておいても、クラスタリング実行中に重複のないパターン(non-redundant expression patterns)にある程度はしてくれます。例えば、これを利用

### 4. サンプルデータ42のリアルデータ(sample\_blekhman\_18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターのどこに割り振られたかの情報を、クラスターごとにノートして保存しています。とりあえずクラスター数を10にしています。

```

in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png" #出力ファイル名を指定してout_f1に格納(pngファイル)
out_f2 <- "hoge4.txt" #出力ファイル名を指定してout_f2に格納(txtファイル)
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10 #クラスター数を指定
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定

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

#入力ファイルの読み込みとラベル情報の作成
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルを読み込み
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3)) #G1群を1、G2群を2、G3群を3とラベルを付ける
dim(data) #オブジェクトdataの行数と列数を表示
    
```

②  
 最近の論文で最終的に20 clusterに利用だと思いましたが、Normalizer=NULLで正規化係数、logFC、NIを入力ファイルや指定「ファイル」-「ディレクトリ」

### 1. サンプルデータ

Biological replicates (最初の180個)は既知です。このデータには3つのクラスターしか存在し

```

in_f <- "data/sample_blekhman_18.txt"
out_f <- "hoge4.png"
param_fig <- c(800, 500)
param_clust_num <- 10
param_G1 <- 6
param_G2 <- 6
    
```

※前処理(正規化)は別途行います

# 遺伝子クラスタリング

①通常のクラスタリングの際は、入力データのどの列がどの群かという情報を与えない。  
 ②MBCluster.Seqは、同一群内のバラツキを考慮してくれるので、列のラベル情報を与えている。Model-based clusteringというのは、(発現変動解析時にnon-DEGの分布を見積もると同様に)同一群内のバラツキを超えた意味のある発現パターンを見積もって返してくれると理解すればよい。

Bioinformatics. 2014 Jan 15;30(2):197-205. doi: 10.1093/bioinformatics/btt632. Epub 2014

## Model-based clustering for RNA-seq data. ②

Si Y<sup>1</sup>, Liu P, Li P, Brutnell TP.

### Author information

### Abstract

**MOTIVATION:** RNA-seq technology has been widely adopted as an attractive method to study global gene expression. However, robust statistical tools to analyze these

complex datasets are still lacking. Model-based clustering provides insight for RNA-seq data analysis.

**RESULTS:** In this manuscript, we propose a model-based clustering method for RNA-seq data. An expectation-maximization algorithm is proposed to improve the likelihood of hierarchical clustering method among clusters as well as flexibility in the analysis of a maize dataset. Our clustering results are compared with alternative methods that are not based on

**AVAILABILITY AND IMPLEMENTATION:** We have implemented our proposed algorithm and made it available at <http://www.r-project.org>

### 4. サンプルデータ42のリアルデータ(sample blekman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターのどこに割り振られたかの情報を、クラスターごとにコメントして保存しています。とりあえずクラスター数を10にしています。

```

in_f <- "sample_blekman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png" #出力ファイル名を指定してout_f1に格納(pngファイル)
out_f2 <- "hoge4.txt" #出力ファイル名を指定してout_f2に格納(txtファイル)
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10 #クラスター数を指定
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
  
```

```

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

```

#入力ファイルの読み込みとラベル情報の作成
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルを読み込み
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3)) #G1群を1、G2群を2、G3群を3とラベルを付ける
dim(data) #オブジェクトdataの行数と列数を表示
  
```



# コピペ

## 4. サンプルデータ42のリアルデータ(sample\_blekhman\_18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターのどこに割り振られたかの情報を、クラスターごとにソートして保存しています。とりあえずクラスター数を10にしています。

```

in_f <- "sample_blekhman_18.txt"      #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png"                 #出力ファイル名を指定してout_f1に格納(pngファイル)
out_f2 <- "hoge4.txt"                 #出力ファイル名を指定してout_f2に格納(txtファイル)
param_fig <- c(800, 500)              #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10                 #クラスター数を指定
param_G1 <- 6                         #G1群のサンプル数を指定
param_G2 <- 6                         #G2群のサンプル数を指定
param_G3 <- 6                         #G3群のサンプル数を指定

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

#入力ファイルの読み込みとラベル情報の作成
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したこ
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))#G1群を1、G2群を2、G
dim(data)                             #オブジェクトdataの行数と列数を表示

#前処理(基礎情報取得)
hoge <- RNASeq.Data(data, Normalizer=NULL,#クラスタリングに必要な基礎情報を
Treatment=data.cl, GeneID=rownames(data))#クラスタリングに必要な

#本番(クラスタリング)

```

```

R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="blekh")
[1] "sample_blekhman_18.txt"
> ls()
character(0)
> |

```

①800×500ピクセルの②hoge4.png。  
内部的に乱数を発生させているので、見栄えはヒトによって異なる。

# 結果の説明

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターのどこに割り振られたかの情報を、クラスターごとにソートして保存しています。とりあえずクラスター数を10にしています。

```

in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4.txt"
param_fig <- c(800, 500)
param_clust_num <- 10
param_G1 <- 6
param_G2 <- 6
param_G3 <- 6

```

```

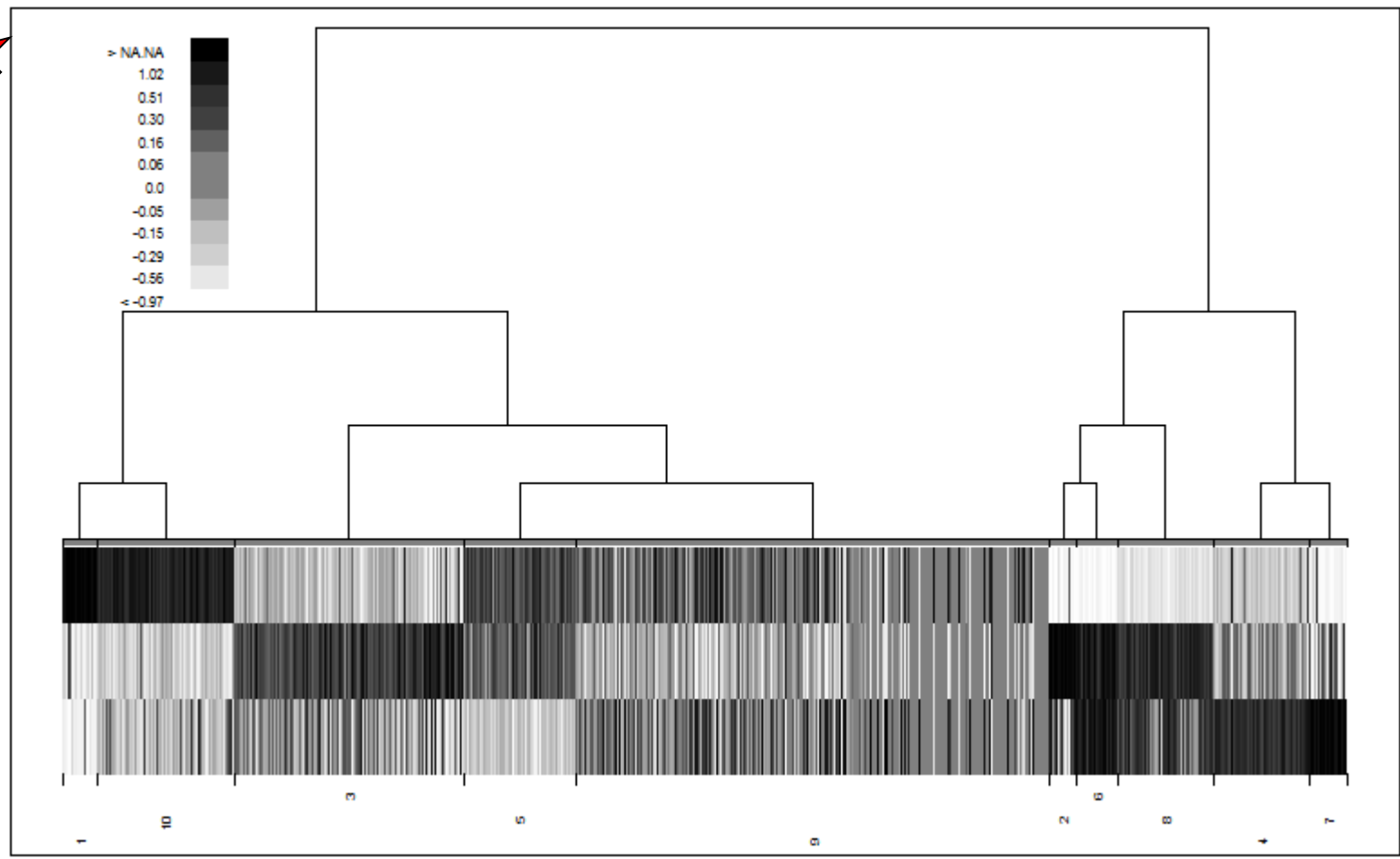
#必要なパッケージをロード
library(MBCluster.Seq)

#入力ファイルの読み込みとラベル
data <- read.table(in_f, header=T, as.is=T)
data.cl <- c(rep(1, param_clust_num), rep(2, param_clust_num), rep(3, param_clust_num), rep(4, param_clust_num), rep(5, param_clust_num))

#前処理(基礎情報取得)
hoge <- RNASeq.Data(data, Treatment=data.cl)

#本番(クラスタリング)

```



# 結果の説明

①下から順にG1(HS)、G2(PT)、G3(RM)。順序は、②のテキストファイル(後述)と見比べることでわかる。  
 ③cluster 1と10はG3群で高発現、④cluster 6と8はG3群で低発現パターンのものだ、みたいに解釈する。全体を眺めることで各クラスターを構成するメンバー数(遺伝子数)の概要をつかめる。例えば、⑤cluster 9を構成する遺伝子数が最多、とか。

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターごとにソートして保存しています。とりあえずクラスター数を10にして

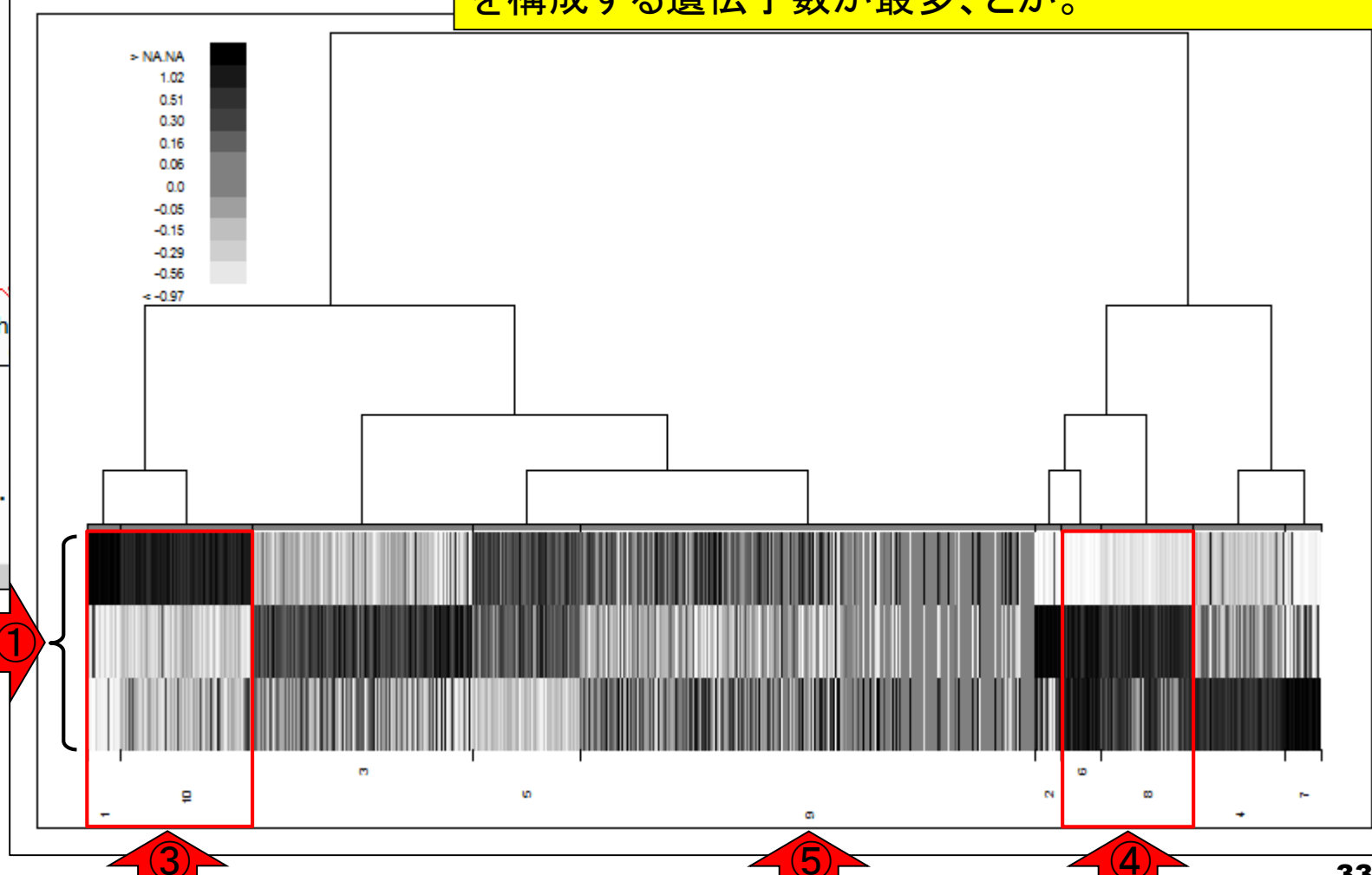
```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定
out_f1 <- "hoge4.png" #出力ファイル名を指定
out_f2 <- "hoge4.txt"
param_fig <- c(800, 500)
param_clust_num <- 10
param_G1 <- 6
param_G2 <- 6
param_G3 <- 6
```

```
#必要なパッケージをロード
library(MBCluster.Seq)

#入力ファイルの読み込みとラベル
data <- read.table(in_f, header=T, as.is=T)
data.cl <- c(rep(1, param_clust_num), rep(2, param_clust_num), rep(3, param_clust_num))

#前処理(基礎情報取得)
hoge <- RNASeq.Data(data, Treatment=data.cl)

#本番(クラスタリング)
hoge <- MBCluster(hoge, param_fig, param_clust_num, param_G1, param_G2, param_G3)
```



①テキストファイル(hoge4.txt)の中身。②1番右側の列がクラスター番号情報。③例題4の場合、出力ファイルはクラスター番号順にソートされている。確かにデンドログラム(樹形図)でみた通り、cluster 1はG3(RM)群で高発現パターンになっている。

# 結果の説明

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターごとにソートして保存しています。とりあえずクラスター数を10にして

```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png" #出力ファイル名を指定してout_f1に格納(pngファイル)
out_f2 <- "hoge4.txt" #出力ファイル名を指定してout_f2に格納(txtファイル)
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10 #クラスター数を指定
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
```

#必要なパッケージをロード

rownames(c)	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	cls\$cluster
ENSG00000002726	1	41	56	41	1	4	1	4	0	0	0	0	17492	25226	10548	18797	23216	11485	1
ENSG00000006747	0	1	1	2	0	1	0	2	1	0	0	0	23	8	12	6	1	10	1
ENSG00000007174	1	0	0	0	0	1	1	1	0	0	0	1	55	6	63	52	82	69	1
ENSG00000016490	0	0	0	0	0	0	0	0	0	0	0	0	4	0	4	0	0	0	1
ENSG00000017483	4	1	11	3	3	3	17	10	6	129	4	4	128	212	206	193	118	121	1
ENSG00000039600	1	0	0	0	0	0	0	1	0	0	1	5	19	16	50	18	3	20	1
ENSG00000042813	0	0	0	0	0	0	1	1	0	0	0	0	6	7	10	13	7	4	1
ENSG00000048545	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	1
ENSG00000049768	2	3	3	6	2	6	0	1	1	0	2	20	100	131	94	84	13	78	1

# コードの解説

①テキストファイル(hoge4.txt)作成部分のコード。② dataオブジェクトは入力ファイルを読み込んだ直後のものなので、正規化前のデータ。③cls\$clusterという数値ベクトルが、入力データの遺伝子の並び順に、どの遺伝子がどのクラスターに属するかを示した情報に相当する。出力がクラスター番号順になっている理由は、④cls\$clusterの並びでソートしているから。門田亡き後もこのようにコードの中身を自力で解読できるようになっておけば大丈夫

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターごとにソートして保存しています。とりあえずクラスター

#前処理(基礎情報取得)

```
hoge <- RNASeq.Data(data, Normalizer=NULL, #クラスタリング
                    Treatment=data.cl, GeneID=rownames(data))
```

#本番(クラスタリング)

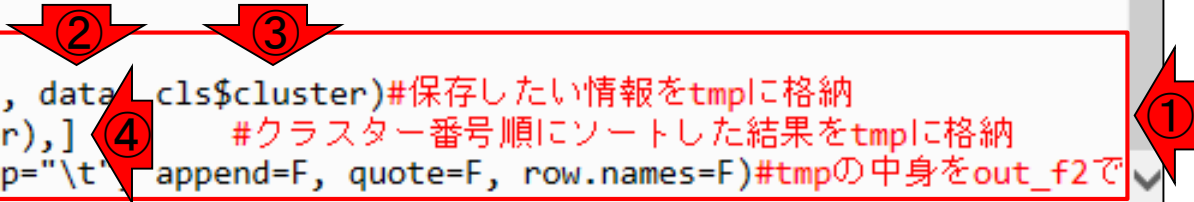
```
c0 <- KmeansPlus.RNASeq(data=hoge, nK=param_clust_num, #K-means clusteringのクラスター中心の初期値を取
                        model="nbinom", print.steps=F) #K-means clusteringのクラスター中心の初期値を取
cls <- Cluster.RNASeq(data=hoge, model="nbinom", #クラスタリング
                     centers=c0$centers, method="EM") #クラスタリング
tr <- Hybrid.Tree(data=hoge, model="nbinom", cluster=cls$cluster) #hybrid-hierarchical clustering
table(cls$cluster) #param_clust_numで指定したパターンに属する遺伝子数
```

#ファイルに保存(pngファイル)

```
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータ
plotHybrid.Tree(merge=tr, cluster=cls$cluster, logFC=hoge$logFC, tree.title=NULL) #描画
dev.off() #おまじない
```

#ファイルに保存(txtファイル)

```
tmp <- cbind(rownames(data), data[, cls$cluster]) #保存したい情報をtmp1に格納
tmp <- tmp[order(cls$cluster), ] #クラスター番号順にソートした結果をtmp1に格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F) #tmpの中身をout_f2で保存
```



# クラスタごとの遺伝子数

hoge4.pngを眺めることで、cluster 9を構成する遺伝子数が最多とかが一応わかるが、各クラスタを構成するメンバー数(遺伝子数)を正確に調べるやり方を伝授。

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスターのどこに割り振られたかの情報を、クラスターごとにソートして保存しています。とりあえずクラスター数を10にしています。

```
in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge4.png"
out_f2 <- "hoge4.txt"
param_fig <- c(800, 500)
param_clust_num <- 10
param_G1 <- 6
param_G2 <- 6
param_G3 <- 6
```

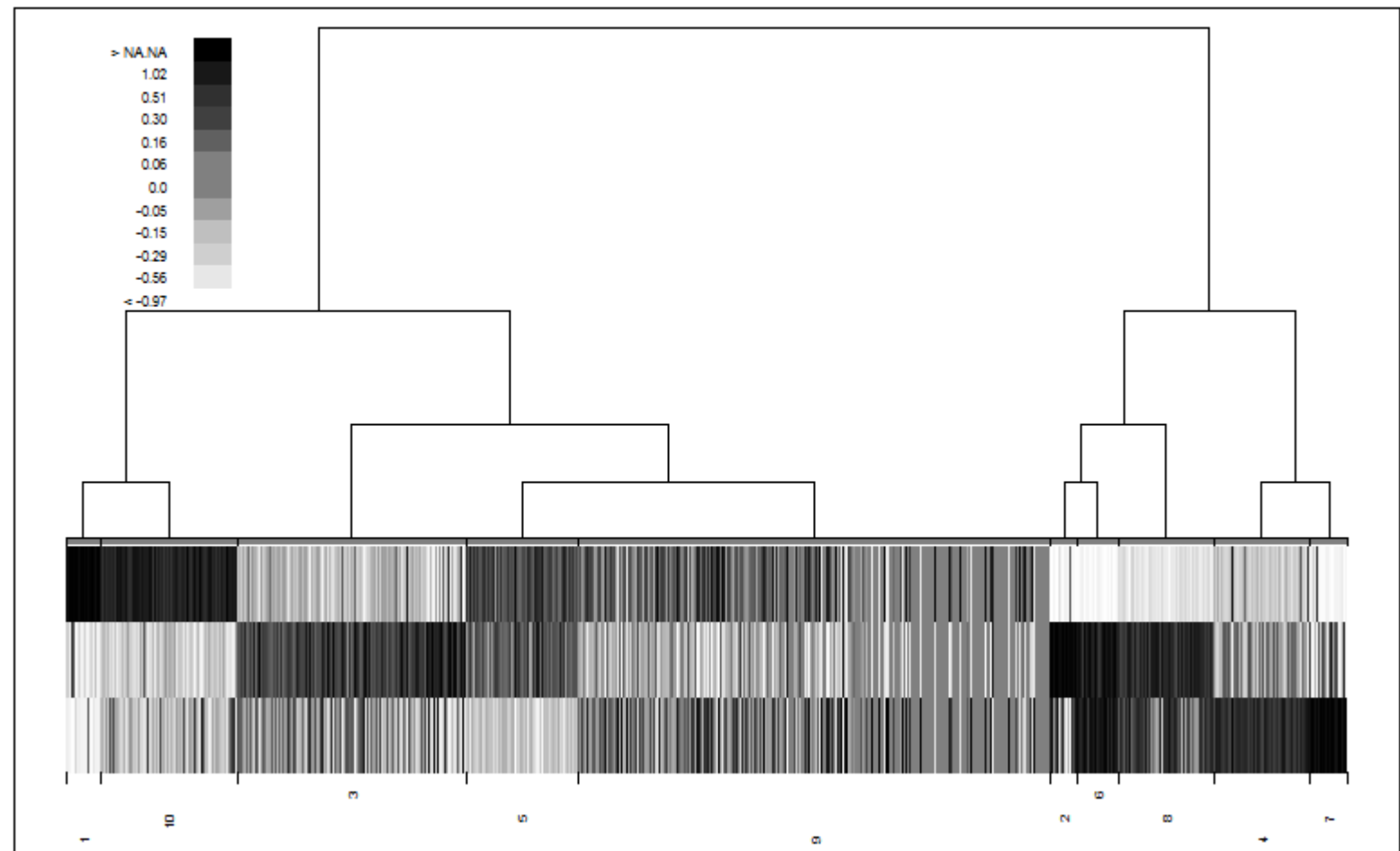
```
#必要なパッケージをロード
library(MBCluster.Seq)
```

```
#入力ファイルの読み込みとラベル
data <- read.table(in_f, header=TRUE)
data.cl <- c(rep(1, param_clust_num), rep(2, param_clust_num))
```

```
#前処理(基礎情報取得)
hoge <- RNASeq.Data(data, Treatment=data.cl)
```

```
#本番(クラスタリング)
```

```
#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_f1に格納
```



# クラスタごとの遺伝子数

## 4. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

3.と基本的に同じですが、遺伝子ごとにparam\_clust\_numで指定したクラスタのどこに割り振るクラスタごとにソートして保存しています。とりあえずクラスタ数を10にしています。

```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.png" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4.txt" #出力ファイル名を指定してout_f2に格納
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定
param_clust_num <- 10 #クラスタ数を指定
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
```

```
#必要なパッケージをロード
library(MBCluster.Seq) #パッケージをロード
```

```
#入力ファイルの読み込みとラベル情報の作成
data <- read.table(in_f, header=TRUE, row.names=1)
data.cl <- c(rep(1, param_G1), rep(2, param_G2), rep(3, param_G3))
dim(data) #オブジェクトの次元
```

```
#前処理(基礎情報取得)
hoge <- RNASeq.Data(data, Normalizer=NULL, #クラスタリング
                    Treatment=data.cl, GeneID=rownames(data))
```

```
#本番(クラスタリング)
```

入力データの遺伝子の並び順にどの遺伝子がどのクラスタに属するかを示した cls\$cluster という数値ベクトルを入力として、  
 ① table関数を実行した結果が欲しいものです。  
 ②を眺めると、確かにcluster 9のメンバー数が最多(7,617個)であることからこの結果が妥当であることが分かります。この結果も乱数を発生させているので、ヒトによって結果は異なる  
 ③は理解を助ける補足情報。

```
R Console
> dim(data)
[1] 20689 18
> length(cls$cluster)
[1] 20689
> head(cls$cluster)
[1] 5 3 3 3 4 3
> table(cls$cluster)
 1     2     3     4     5     6     7     8     9    10
565  437 3697 1555 1807  654  621 1535 7617 2201
> x <- c("kk", "apu", "kk", "iu", "iu", "kk")
> x
[1] "kk" "apu" "kk" "iu" "iu" "kk"
> table(x)
x
apu  iu  kk
 1    2   3
```



# Contents2

## ■ トランスクリプトーム解析

- インTRODakシヨN: 簡単な原理、基本イメージ
- 様々な解析目的
- 解析データ: 乳酸菌(*L. casei* 12A)
- QuasRでマッピング(基礎): コード各部の説明と結果の解釈
- QuasRでマッピング(応用): オプションを指定して実行
- カウント情報取得1, 2
- サンプル間クラスタリング(TCC)
- 発現変動解析(TCC)、M-A plot
- モデル、分布、統計的手法
- 3群間比較(TCCによるANOVA的な解析)
- 遺伝子間クラスタリング(MBCluster.Seq)
- 3群間比較(TCCによるANOVA的な解析 + MBCluster.Seqでのパターン分類)



# TCC + MBCluster.Seq

①「応用」の項目の、②例題8が、TCCでどこかの群間で発現変動する遺伝子とMBCluster.Seqを組み合わせたものです。コピペ。約20分。

- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [DESeq2\(Love\\_2014\)](#) (last modified 2015/02/04)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/05) 推奨
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [EBSeq\(Leng\\_2013\)](#) (last modified 2016/02/16) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [SAMseq\(Li\\_2013\)](#) (last modified 2015/02/10)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [DESeq\(Anders\\_2010\)](#) (last modified 2014/03/13)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [baySeq\(Hardcastle\\_2010\)](#) (last modified 2016/02/16)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎 | [edgeR\(Robinson\\_2010\)](#) (last modified 2015/02/03)
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 応用 | [TCC\(Sun\\_2013\)](#) (last modified 2016/02/17) 推奨 NE
- 解析 | 発現変動 | 3群間 | 対応なし | 複製なし | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/05) 推奨
- 解析 | 発現変動 | 5群間 | 対応なし | 複製あり | [TCC\(Sun\\_2013\)](#) (last modified 2015/11/05) 推奨

## 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 応用 | [TCC\(Sun\\_2013\)](#) NEW

「応用」でやりたいことは、例題1-7「指定した群間で発現に差がある遺伝子の検出」、そして例題8以降が「基礎」の結果と遺伝子間クラスタリングを組み合わせたパターン分類です。例題1-7「指定した群間で発現に差がある遺伝子の検出」デザイン行列design中のparam\_coefで指定した列を除くことでreduced\_modelを作成します。TCCを用いたやり方を示します。

### ② 8. サンプルデータ42のリアルデータ(sample\_blekhman\_18.txt)の場合:

7と基本的に同じで、入力ファイルが違うだけです。[Blekhman et al., Genome Res., 2010](#)の 20,689 genes×18 samples のカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```

in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge8.txt" #出力ファイル名を指定してout_f1に格納(txtファイル)
out_f2 <- "hoge8.png" #出力ファイル名を指定してout_f2に格納(pngファイル)
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)閾値を指定
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10 #クラスター数を指定

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

### 1. サンプルデータ

シミュレーション  
DEG (gene\_1~  
gene\_3000)がG3  
く、どこかの群間

# TCC + MBCluster.Seq

①800×500ピクセルの②hoge8.png。内部的に乱数を発生させているので、見栄えはヒトによって異なる。この結果の場合は、③cluster 7がnon-DEGパターンであり、最多の遺伝子数から構成される

## 8. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

7と基本的に同じで、入力ファイルが違うだけです。Blekhman et al., Genome Res., 2010の 20,689 のカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
in_f <- "sample_blekhman_18.txt"
out_f1 <- "hoge8.txt"
out_f2 <- "hoge8.png"
param_G1 <- 6
param_G2 <- 6
param_G3 <- 6
param_FDR <- 0.05
param_fig <- c(800, 500)
param_clust_num <- 10
```

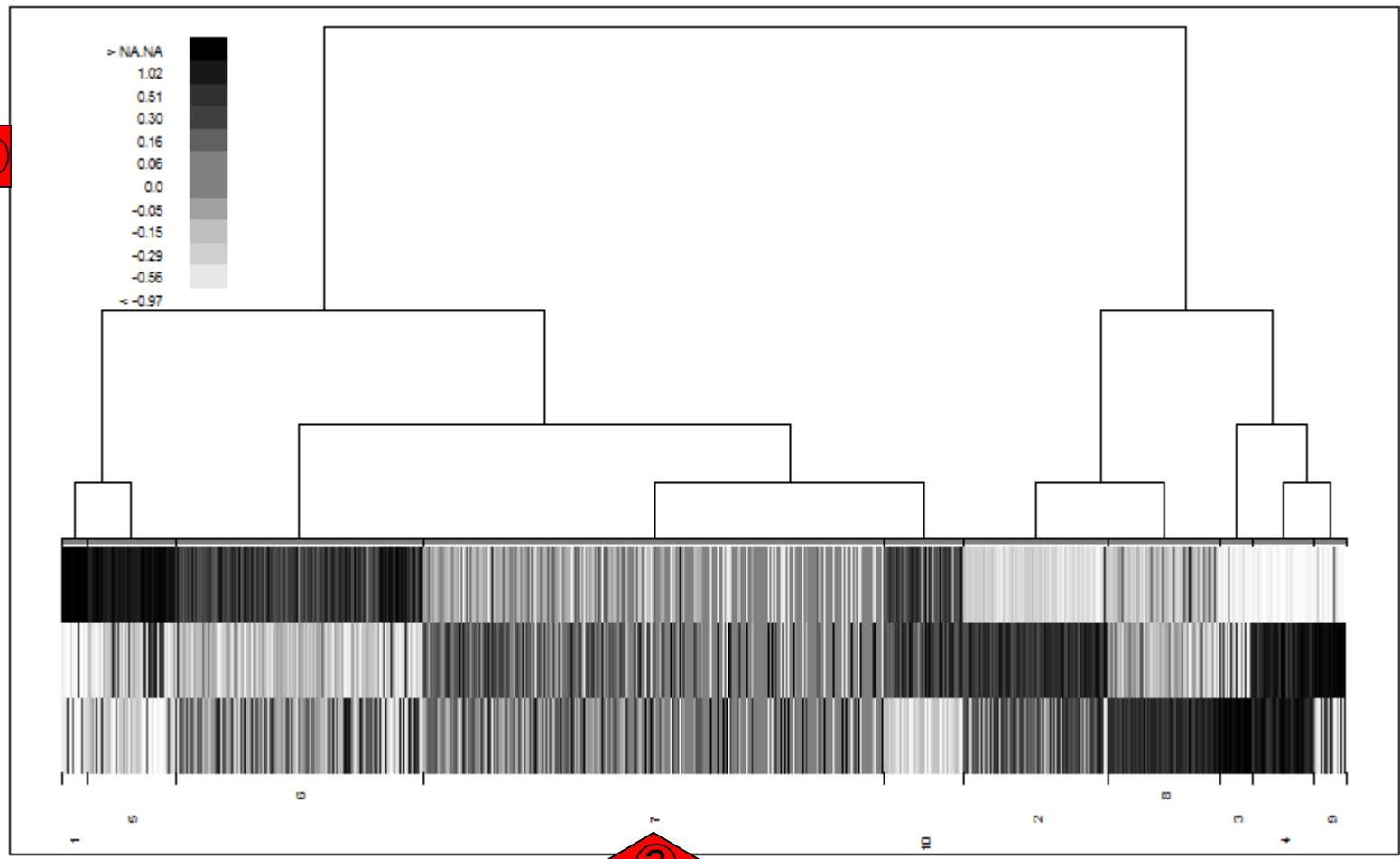
#入力ファイル名を指定してin\_fに格納  
#出力ファイル名を指定してout\_f1に格納(テキストファイル)  
#出力ファイル名を指定してout\_f2に格納(ピクセル)

```
#必要なパッケージをロード
library(TCC)
library(MBCluster.Seq)

#入力ファイルの読み込み
data <- read.table(in_f, h

#前処理(TCCクラスオブジェクト)
data.cl <- c(rep(1, param_
tcc <- new("TCC", data, da

#本番(TCC正規化)
```



# TCC + MBCluster.Seq

①テキストファイル(hoge8.txt)の中身。TCCの結果部分是不変。②一番右側の列の数値はおそらくヒトによって異なる

## 8. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

7と基本的に同じで、入力ファイルが違うだけです。Blekhman et al., Genome Res., 2010の 20,689 genes×18 samples のカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
in_f <- "sample_blekhman_18.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge8.txt" #出力ファイル名を指定してout_f1に格納(txtファイル)
out_f2 <- "hoge8.png" #出力ファイル名を指定してout_f2に格納(pngファイル)
param_G1 <- 6 #G1群のサンプル数を指定
param_G2 <- 6 #G2群のサンプル数を指定
param_G3 <- 6 #G3群のサンプル数を指定
param_FDR <- 0.05 #DEG検出時のfalse discovery rate (FDR)閾値を指定
param_fig <- c(800, 500) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_clust_num <- 10 #クラスター数を指定
```

#必要なパッケージをロード

rownames(t)	HSF1	HSF2	HSF3	HSM1	HSM2	HSM3	PTF1	PTF2	PTF3	PTM1	PTM2	PTM3	RMF1	RMF2	RMF3	RMM1	RMM2	RMM3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG	cls\$cluster
ENSG00000208587	1	1	4	0	0	0	###	###	###	###	###	###	5	2	1	4	9	4	ENNA	NA	NA	1.E-148	2.E-144	1	1	9
ENSG00000209449	0	0	1	0	4	1	1	1	0	2	2	0	657	857	378	718	###	551	ENNA	NA	NA	2.E-141	3.E-137	2	1	1
ENSG00000157399	446	390	297	302	660	462	2	6	3	5	4	2	1	0	0	0	0	2	ENNA	NA	NA	3.E-131	2.E-127	3	1	3
ENSG00000209007	0	0	0	0	0	1	1	0	0	0	4	0	627	880	405	483	###	440	ENNA	NA	NA	7.E-125	3.E-121	4	1	1
ENSG00000134201	16	7	18	3	15	12	0	1	0	0	0	0	###	###	###	###	###	###	ENNA	NA	NA	7.E-115	3.E-111	5	1	1
ENSG00000145244	1	0	1	0	2	3	0	3	1	0	1	0	216	363	355	263	266	174	ENNA	NA	NA	3.E-113	9.E-110	6	1	1
ENSG00000208570	0	0	0	0	0	0	###	###	###	###	###	###	###	###	###	###	###	790	ENNA	NA	NA	2.E-106	7.E-103	7	1	10
ENSG00000220191	2	5	1	3	6	11	2	5	0	6	10	11	###	918	892	###	935	658	ENNA	NA	NA	2.E-105	5.E-102	8	1	1
ENSG00000208978	98	173	118	192	66	142	10	19	6	16	3	1	###	###	###	###	###	###	ENNA	NA	NA	2.E-100	5.E-97	9	1	1
ENSG00000208070	0	0	0	0	0	0	0	0	0	0	0	0	190	186	98	225	205	113	ENNA	NA	NA	9.E-99	2.E-95	10	1	1
ENSG00000218007	0	0	0	0	0	0	0	0	0	0	0	0	628	###	394	611	463	123	ENNA	NA	NA	6.E-97	1.E-93	11	1	1
ENSG00000220688	27	12	29	27	17	21	201	362	244	424	566	416	0	1	0	0	0	1	ENNA	NA	NA	1.E-95	2.E-92	12	1	9

# クラスターごとの遺伝子数

コードの最後の部分を表示。①全遺伝子(20,689個)を対象としたクラスターごとの遺伝子数。pngファイルで眺めたnon-DEGパターンに相当する②cluster 7の遺伝子数が最多(7,413個)となっており妥当。

## 8. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

7と基本的に同じで、入力ファイルが違うだけです。Blekhman et al., Genome Res., 2010の 20,689 のカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, re
tmp <- tmp[order(tmp$rank),] #発現変動
write.table(tmp, out_f1, sep="\t", append=F, quo
sum(tcc$stat$q.value < 0.05) #q-value
sum(tcc$stat$q.value < 0.10) #q-value
sum(tcc$stat$q.value < 0.20) #q-value
sum(tcc$stat$q.value < 0.30) #q-value
```

```
#ファイルに保存(pngファイル)
png(out_f2, pointsize=13, width=param_fig[1], he
plotHybrid.Tree(merge=tr, cluster=cls$cluster, l
dev.off() #おまじない
```

```
#後処理(各クラスターに属する遺伝子数を表示)
table(cls$cluster) #各クラスターごとの遺伝子数
table(cls$cluster[tcc$stat$q.value < 0.05]) #各ク
table(cls$cluster[tcc$stat$q.value < 0.10]) #各ク
table(cls$cluster[tcc$stat$q.value < 0.20]) #各ク
table(cls$cluster[tcc$stat$q.value < 0.30]) #各ク
```

```
R Console
> #後処理 (各クラスターに属する遺伝子数を表示)
> table(cls$cluster) #各クラスターごとの遺伝子数
  1     2     3     4     5     6     7     8     9    10
415 2320  515  986 1417 3999 7413 1808  532 1284
> table(cls$cluster[tcc$stat$q.value < 0.05]) #各ク
  1     2     3     4     5     6     7     8     9    10
377 1309  432  812 1181 1081  67  783  442  763
> table(cls$cluster[tcc$stat$q.value < 0.10]) #各ク
  1     2     3     4     5     6     7     8     9    10
410 1522  485  869 1248 1399  152  970  493  939
> table(cls$cluster[tcc$stat$q.value < 0.20]) #各ク
  1     2     3     4     5     6     7     8     9    10
415 1802  513  930 1377 1816  348 1228  532 1102
> table(cls$cluster[tcc$stat$q.value < 0.30]) #各ク
  1     2     3     4     5     6     7     8     9    10
415 1981  514  975 1408 2278  675 1496  532 1195
>
```



# クラスターごとの遺伝子数

①5% FDR閾値を満たす遺伝子(7,247個)に限定して、クラスターごとの遺伝子数を再分類。non-DEGパターンに相当する②cluster 7の遺伝子数が最少(67個)となっており妥当

## 8. サンプルデータ42のリアルデータ(sample blekhman 18.txt)の場合:

7と基本的に同じで、入力ファイルが違います。Blekhman et al., Genome Res., 2010の 20,000 genes ~ 18 samples のカウントデータです。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種間比較です。

```
#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(tcc$count), normalized, re
tmp <- tmp[order(tmp$rank),] #発現変動
write.table(tmp, out_f1, sep="\t", append=F, quo
sum(tcc$stat$q.value < 0.05) #q-value
sum(tcc$stat$q.value < 0.10) #q-value
sum(tcc$stat$q.value < 0.20) #q-value
sum(tcc$stat$q.value < 0.30) #q-value

#ファイルに保存(pngファイル)
png(out_f2, pointsize=13, width=param_fig[1], he
plotHybrid.Tree(merge=tr, cluster=cls$cluster, l
dev.off() #おまじない

#後処理(各クラスターに属する遺伝子数を表示)
table(cls$cluster) #各クラ$
table(cls$cluster[tcc$stat$q.value < 0.05]) #各クラ$
table(cls$cluster[tcc$stat$q.value < 0.10]) #各クラ$
table(cls$cluster[tcc$stat$q.value < 0.20]) #各クラ$
table(cls$cluster[tcc$stat$q.value < 0.30]) #各クラ$
```

```
R Console
> #後処理 (各クラスターに属する遺伝子数を表示)
> table(cls$cluster) #各クラ$
 1     2     3     4     5     6     7     8     9    10
415 2320  515  986 1417 3999 7413 1808  532 1284

> table(cls$cluster[tcc$stat$q.value < 0.05]) #各クラ$
 1     2     3     4     5     6     7     8     9    10
377 1309  432  812 1181 1081  67  783  442  763

> table(cls$cluster[tcc$stat$q.value < 0.10]) #各クラ$
 1     2     3     4     5     6     7     8     9    10
410 1522  485  869 1248 1399  152  970  493  939

> table(cls$cluster[tcc$stat$q.value < 0.20]) #各クラ$
 1     2     3     4     5     6     7     8     9    10
415 1802  513  930 1377 1816  348 1228  532 1102

> table(cls$cluster[tcc$stat$q.value < 0.30]) #各クラ$
 1     2     3     4     5     6     7     8     9    10
415 1981  514  975 1408 2278  675 1496  532 1195

>
```



# 共同研究者、謝辞

所属・敬称略。他にもバグレポートやプログラム提供をいただいた諸氏、R本体および有用なパッケージ開発者諸氏に御礼申し上げますm(\_ \_)m

## アグリバイオ本体、TCCパッケージ、および手法比較

清水謙多郎、寺田透、三浦文、孫建強、西山智明、湯敏

## DDBJ Pipeline、Platanus、および日本乳酸菌学会誌

谷澤靖洋、神沼英里、中村保一、遠野雅徳、有田正規、伊藤武彦、鈴木チセ、坂本光央

## HPCI人材養成プログラム

杉原稔、寺田朋子

## グラント

- 基盤研究(C)(H27-29年度):「ロングリード時代に対応したトランスクリプトームデータ解析ガイドラインの構築」(代表)
- 基盤研究(C)(H24-26年度):「シーケンスに基づく比較トランスクリプトーム解析のためのガイドライン構築」(代表)
- 新学術領域研究(研究領域提案型)(H22-26年度):「非モデル生物におけるゲノム解析法の確立」(分担;研究代表者:西山智明)
- NBDCとの共同研究(H26-27年度):NGS講習会関連

