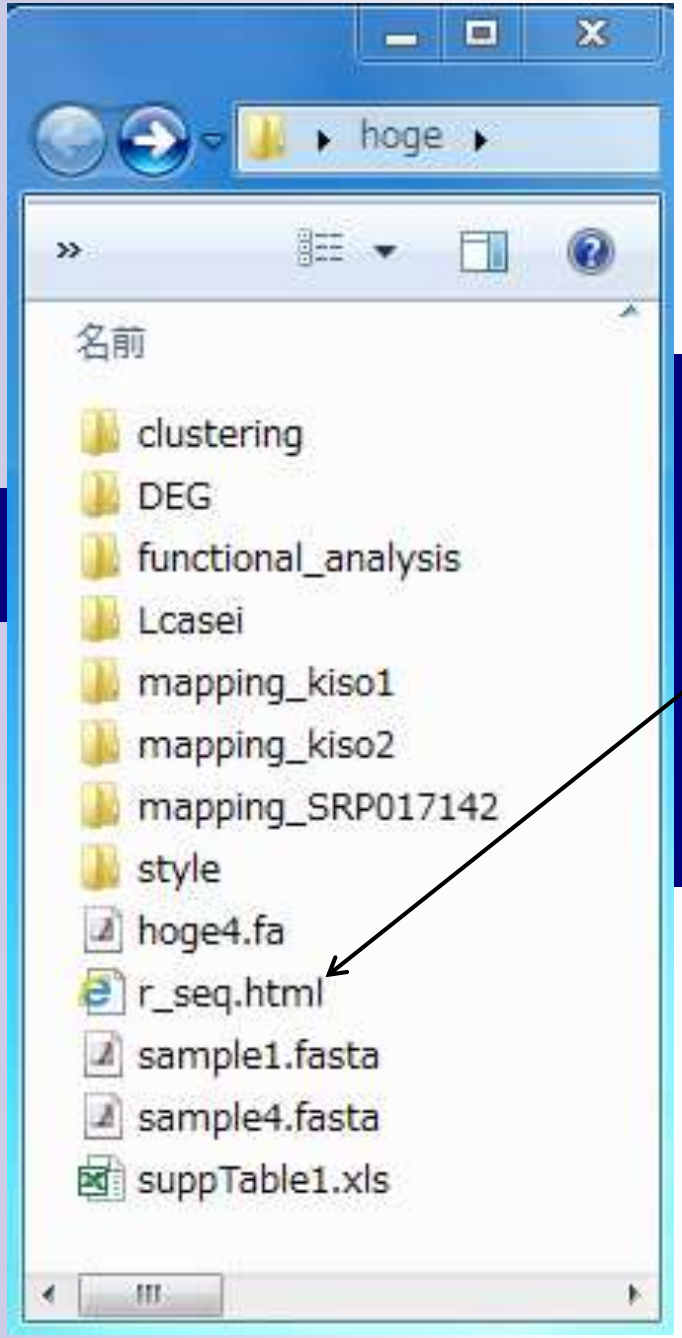


実習用PCのデスクトップ上に、hogeフォルダがあります。この中に解析に必要な入力ファイルがあります。ネットワーク不具合時（そうでなくても）はローカル環境でhtmlファイルを起動して各自対応してください。

Rでゲノム・トランスクリプトーム解析

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

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ですが、平成27年度もやります！

- 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

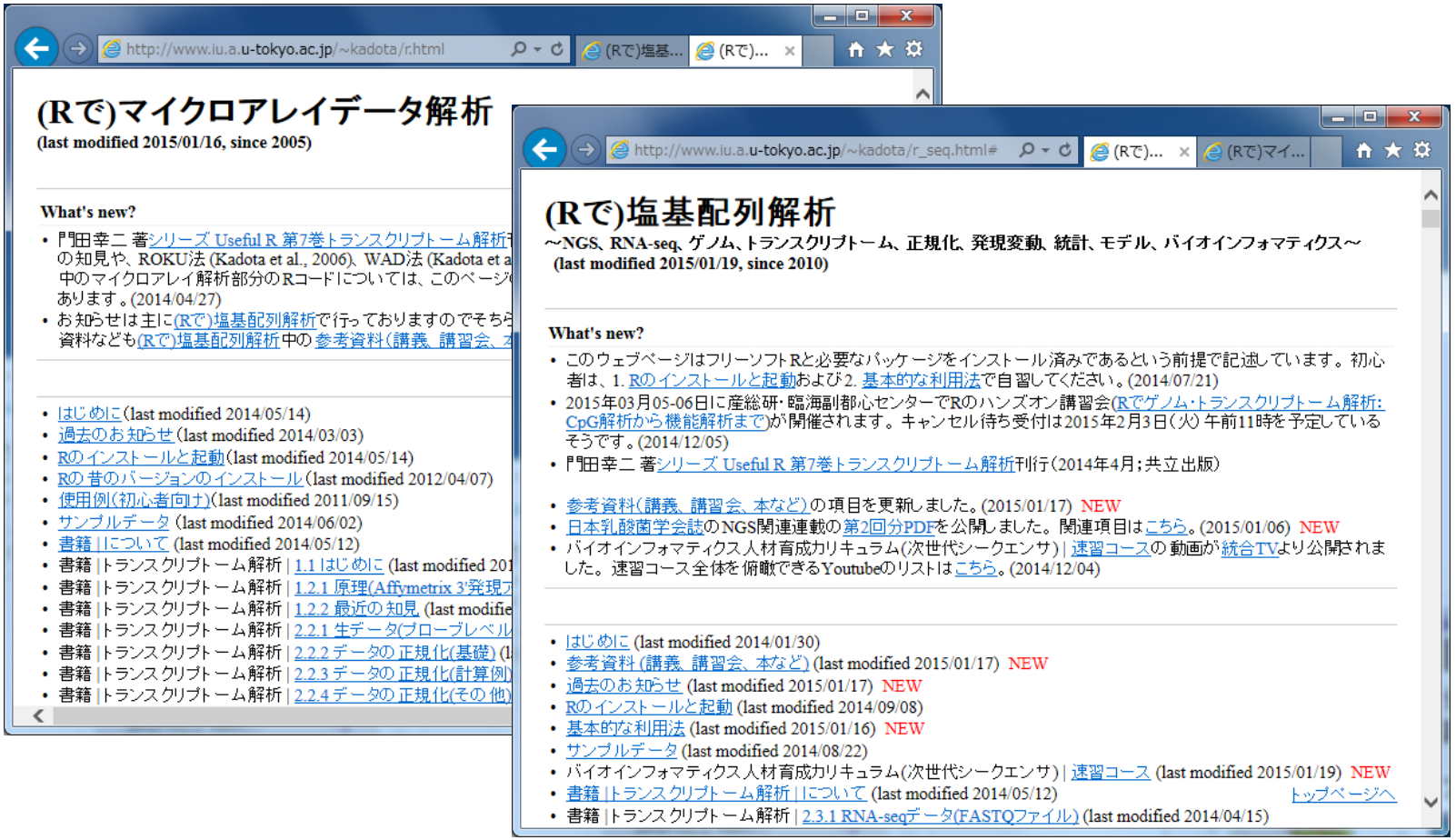
年度	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
修士課程	12	65	73	83	68	72	107	100	121	124	108
博士課程	3	7	11	13	6	8	12	21	16	19	24
社会人	5	3	8	4	1	0	11	19	32	26	55
合計	**20**	**75**	**92**	**100**	**75**	**80**	**130**	**140**	**169**	**169**	**187**
開講科目数	9	15	15	15	15	12	15	15	14	15	13
常勤教員数	6	6	7	7	7	3	4	4	3	2	2

1科目以上の合格者数

自己紹介（基本スタンス）

自習可能な学習教材作成が使命。
Rのバージョンアップに伴う不具合
対応に忙殺され、カナリお疲れ気味。

- 分からないヒトの立場にたち、わかりやすいハンズオン講義を追求
- 詳細かつ丁寧な講義・講演資料をPDFで無条件公開
- 主にフリーソフトウェアRを用いたデータ解析の手順をウェブ上で公開



Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

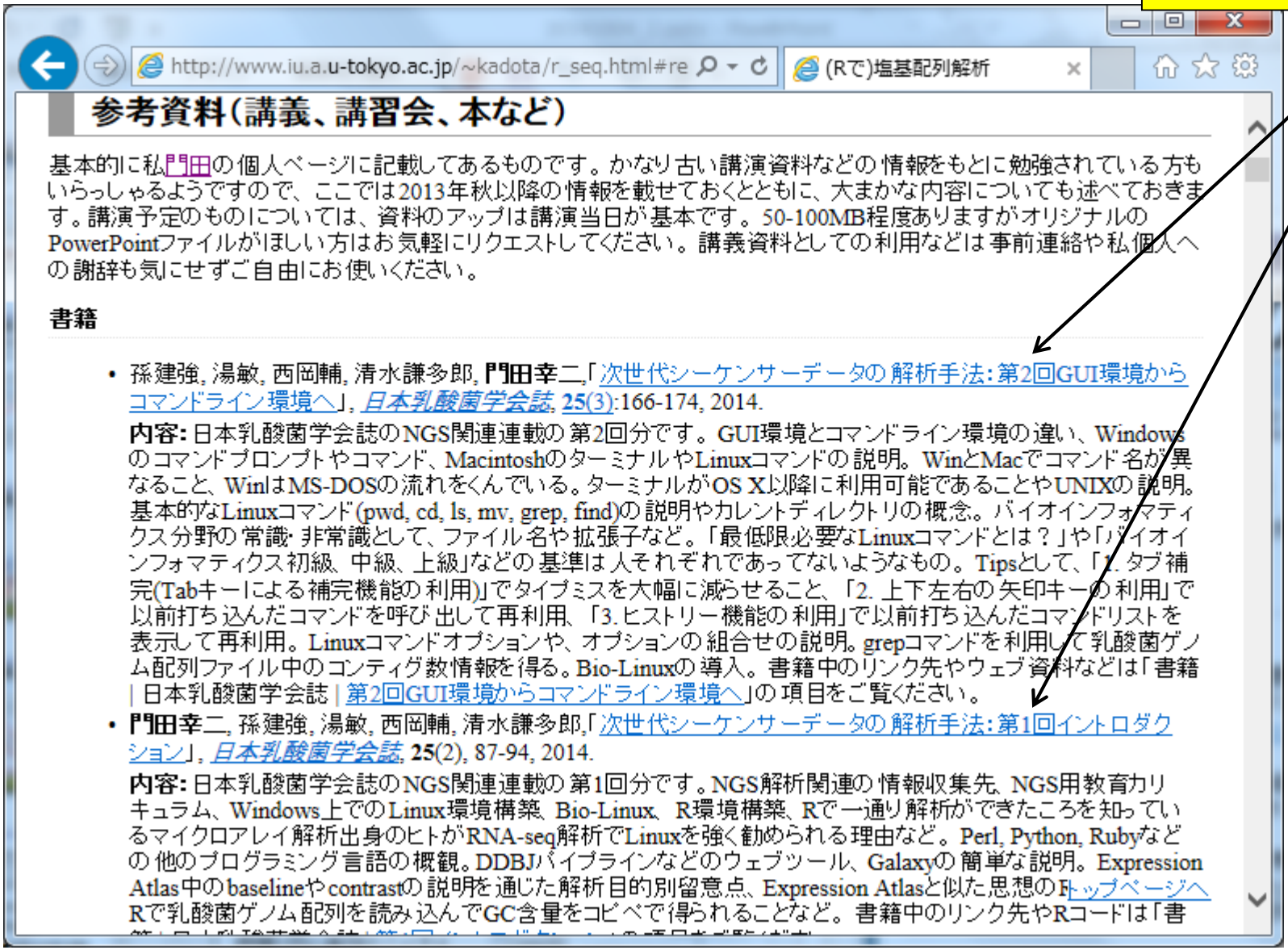
イントロダクション

ここでは、Rを中心として、本務の大学院講義(90分×18コマ=**27時間分**)スライドを含め、2013年秋以降のPDFファイルを簡単な解説つきで公開。

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

イントロダクション

これはLinux系の教材。日本乳酸菌学会誌の連載。第3回分は2015年3-4月ごろ公開予定。



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

(Rで)塩基配列解析

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。

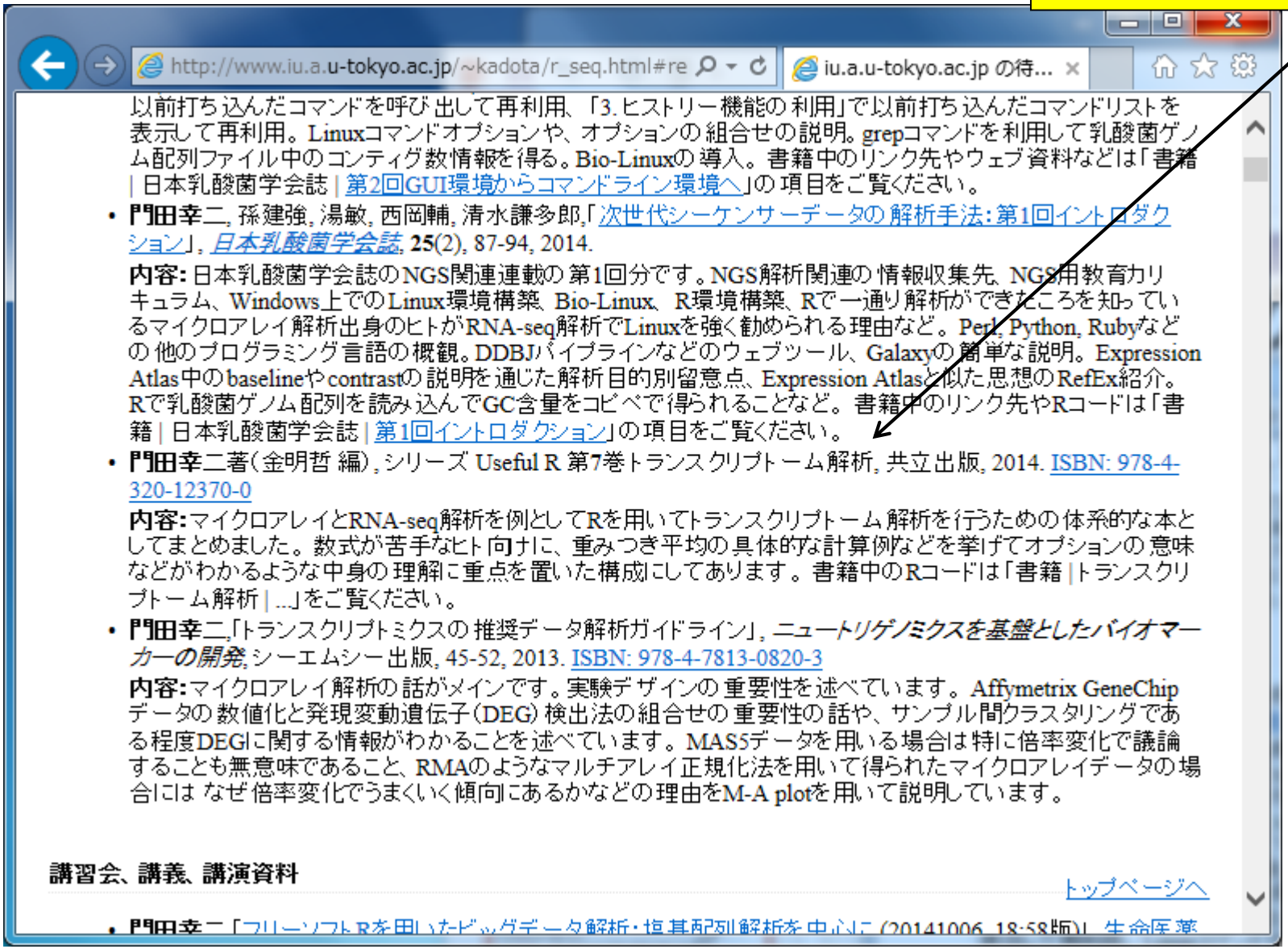
書籍

- 孫建強, 湯敏, 西岡輔, 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2回GUI環境からコマンドライン環境へ」, *日本乳酸菌学会誌*, **25**(3):166-174, 2014.
  内容: 日本乳酸菌学会誌のNGS関連連載の第2回分です。GUI環境とコマンドライン環境の違い、Windowsのコマンドプロンプトやコマンド、Macintoshのターミナルや Linuxコマンドの説明。WinとMacでコマンド名が異なること、WinはMS-DOSの流れをくんでいる。ターミナルがOS X以降に利用可能であることやUNIXの説明。基本的なLinuxコマンド(pwd, cd, ls, mv, grep, find)の説明やカレントディレクトリの概念。バイオインフォマティクス分野の常識・非常識として、ファイル名や拡張子など。「最低限必要なLinuxコマンドとは?」や「バイオインフォマティクス初級、中級、上級」などの基準は人それぞれであってないようなもの。Tipsとして、「1. タブ補完(Tabキーによる補完機能の利用)」でタイプミスを大幅に減らせること、「2. 上下左右の矢印キーの利用」で以前打ち込んだコマンドを呼び出して再利用、「3. ヒストリー機能の利用」で以前打ち込んだコマンドリストを表示して再利用。Linuxコマンドオプションや、オプションの組合せの説明。grepコマンドを利用して乳酸菌ゲノム配列ファイル中のコンティグ数情報を得る。Bio-Linuxの導入。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2回GUI環境からコマンドライン環境へ」の項目をご覧ください。
- **門田幸二**, 孫建強, 湯敏, 西岡輔, 清水謙多郎,「次世代シーケンサーデータの解析手法:第1回イントロダクション」, *日本乳酸菌学会誌*, **25**(2), 87-94, 2014.
  内容: 日本乳酸菌学会誌のNGS関連連載の第1回分です。NGS解析関連の情報収集先、NGS用教育カリキュラム、Windows上でのLinux環境構築、Bio-Linux、R環境構築、Rで一通り解析ができたころを知っているマイクロアレイ解析出身のヒトがRNA-seq解析でLinuxを強く勧められる理由など。Perl, Python, Rubyなどの他のプログラミング言語の概観。DDBJパイプラインなどのウェブツール、Galaxyの簡単な説明。Expression Atlas中のbaselineやcontrastの説明を通じた解析目的別留意点、Expression Atlasと似た思想のRで乳酸菌ゲノム配列を読み込んでGC含量をコピペで得られることなど。書籍中のリンク先やRコードは「書

トップページへ

イントロダクション

2014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。



以前打ち込んだコマンドを呼び出して再利用、「3.ヒストリー機能の利用」で以前打ち込んだコマンドリストを表示して再利用。Linuxコマンドオプションや、オプションの組合せの説明。grepコマンドを利用して乳酸菌ゲノム配列ファイル中のコンティグ数情報を得る。Bio-Linuxの導入。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2回GUI環境からコマンドライン環境へ」の項目をご覧ください。

- **門田幸二**, 孫建強, 湯敏, 西岡輔, 清水謙多郎,「次世代シーケンサーデータの解析手法:第1回イントロダクション」, *日本乳酸菌学会誌*, **25**(2), 87-94, 2014.
  内容: 日本乳酸菌学会誌のNGS関連連載の第1回分です。NGS解析関連の情報収集先、NGS用教育カリキュラム、Windows上でのLinux環境構築、Bio-Linux、R環境構築、Rで一通り解析ができたころを知っているマイクロアレイ解析出身のヒトがRNA-seq解析でLinuxを強く勧められる理由など。Perl, Python, Rubyなどの他のプログラミング言語の概観。DDBJパイプラインなどのウェブツール、Galaxyの簡単な説明。Expression Atlas中のbaselineやcontrastの説明を通じた解析目的別留意点、Expression Atlasと似た思想のRefEx紹介。Rで乳酸菌ゲノム配列を読み込んでGC含量をコピペ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あります。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 トップページへ

- **門田幸二**「フリーソフトRを用いたビッグデータ解析・塩基配列解析を中心に(20141006_18:58版)」, 生命医薬

イントロダクション

さらに1ページ分ほど下に移動すると、「講習会、講義、講演資料」のPDFが見られる。時系列順にリストアップ。

- **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。

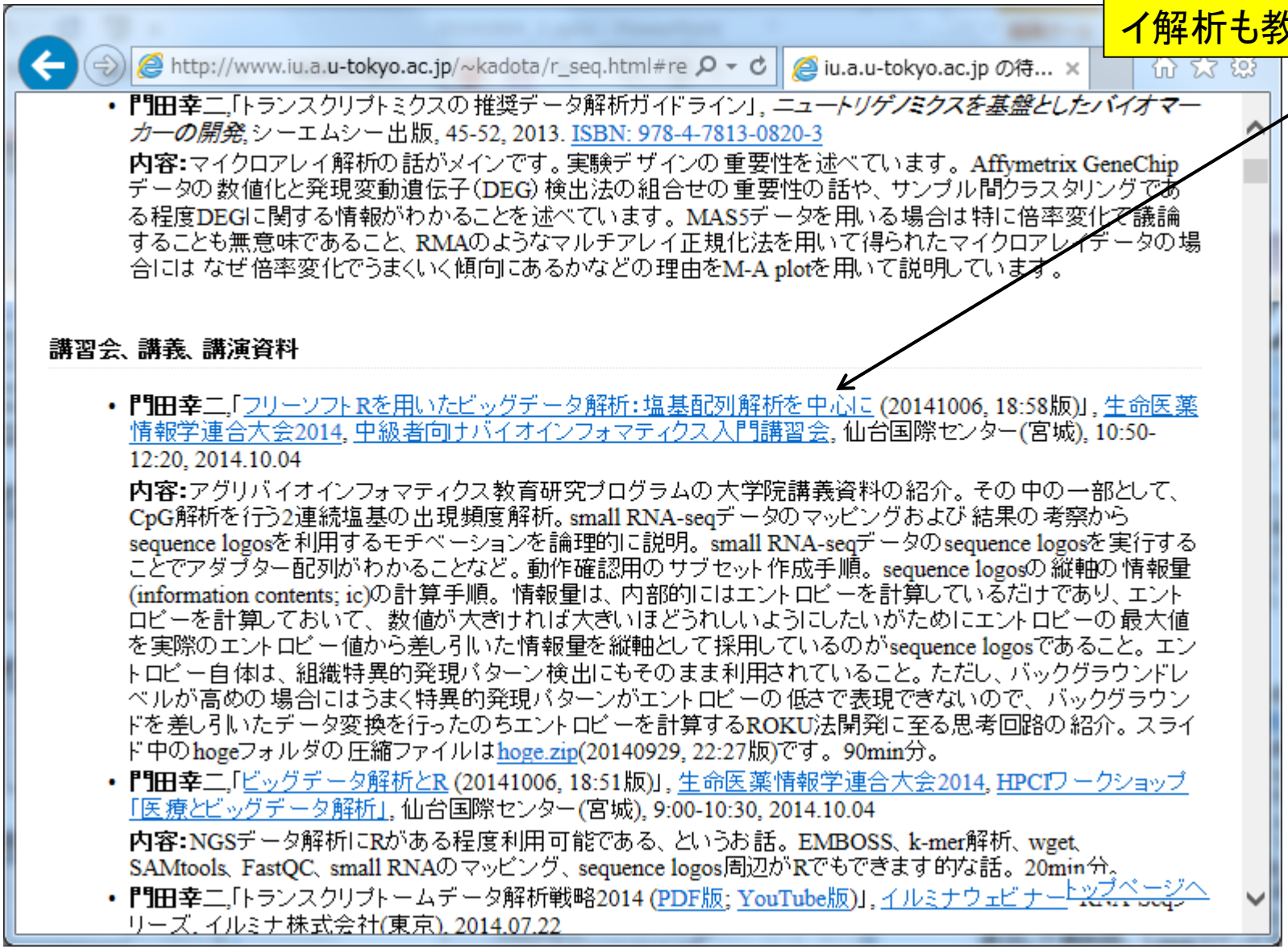
講習会、講義、講演資料

- **門田幸二**,「フリーソフトRを用いたビッグデータ解析:塩基配列解析を中心に (20141006, 18:58版)」, 生命医薬情報学連合大会2014, 中級者向けバイオインフォマティクス入門講習会, 仙台国際センター(宮城), 10:50-12:20, 2014.10.04
  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大学院講義資料の紹介。その中の一部として、CpG解析を行う2連続塩基の出現頻度解析。small RNA-seqデータのマッピングおよび結果の考察からsequence logosを利用するモチベーションを論理的に説明。small RNA-seqデータのsequence logosを実行することでアダプター配列がわかることなど。動作確認用のサブセット作成手順。sequence logosの縦軸の情報量(information contents; ic)の計算手順。情報量は、内部的にはエントロピーを計算しているだけであり、エントロピーを計算しておいて、数値が大きければ大きいほどうれしいようにしたいがためにエントロピーの最大値を実際のエントロピー値から差し引いた情報量を縦軸として採用しているのがsequence logosであること。エントロピー自体は、組織特異的発現パターン検出にもそのまま利用されていること。ただし、バックグラウンドレベルが高めの場合にはうまく特異的発現パターンがエントロピーの低さで表現できないので、バックグラウンドを差し引いたデータ変換を行ったのちエントロピーを計算するROKU法開発に至る思考回路の紹介。スライド中のhogeフォルダの圧縮ファイルはhoge.zip(20140929, 22:27版)です。90min分。
- **門田幸二**,「ビッグデータ解析とR (20141006, 18:51版)」, 生命医薬情報学連合大会2014, HPCIワークショップ「医療とビッグデータ解析」, 仙台国際センター(宮城), 9:00-10:30, 2014.10.04
  内容: NGSデータ解析にRがある程度利用可能である、というお話。EMBOSS、k-mer解析、wget、SAMtools、FastQC、small RNAのマッピング、sequence logos周辺がRでもできます的な話。20min分。
- **門田幸二**,「トランスクリプトームデータ解析戦略2014 (PDF版; YouTube版)」, イルミナウェビナーシリーズ, イルミナ株式会社(東京), 2014.07.22

トップページへ

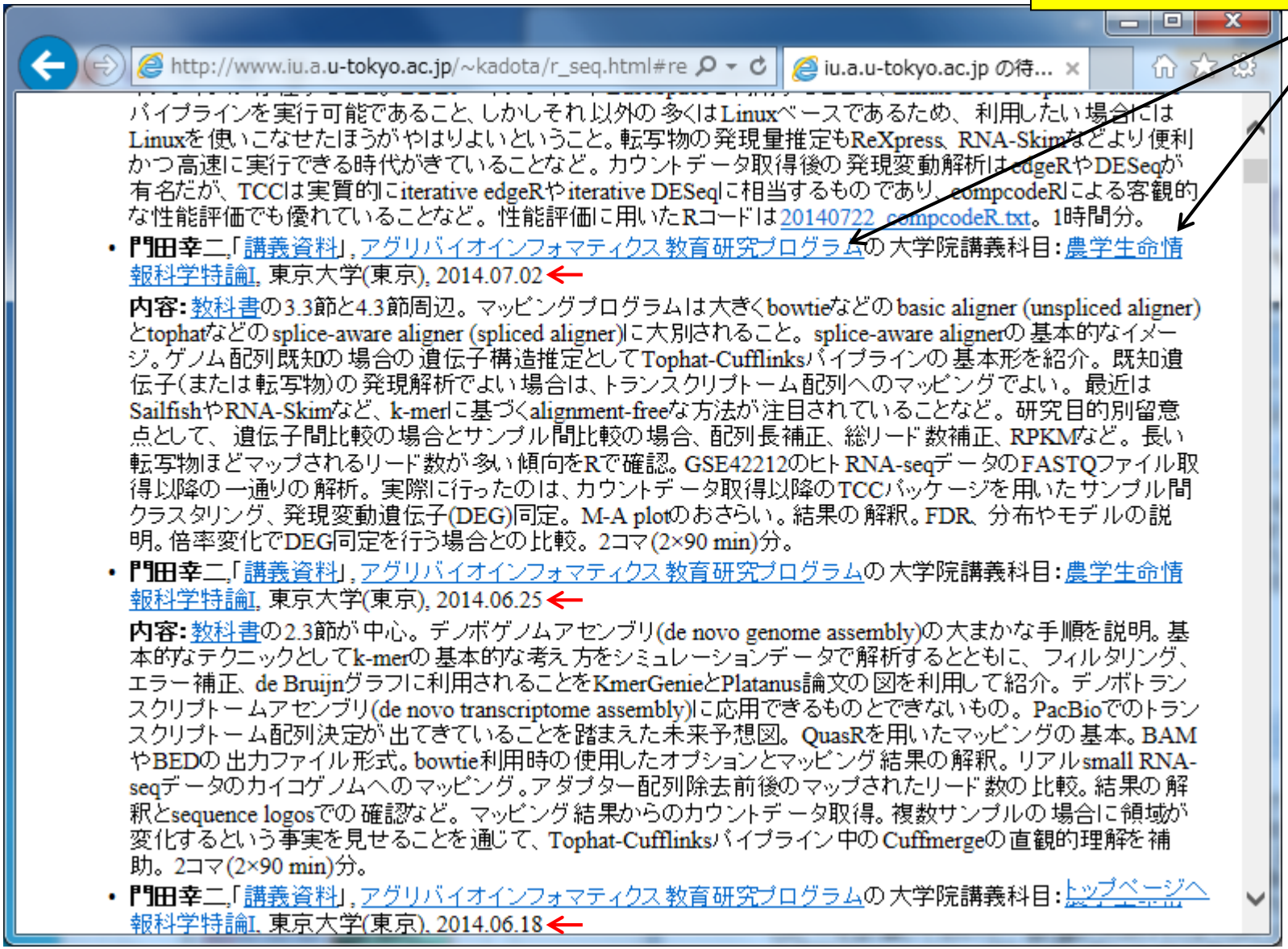
イントロダクション

これは、2013年度HPCI講習会でのリクエスト（Rコードの中身の詳細な説明が欲しい、マイクロアレイ解析も教えて）に応えたもの。



イントロダクション

Rを中心としたアグリバイオ大学院講義資料(門田分のみ)の平成26年度分は、2014.04.09から2014.07.02です。



パイプラインを実行可能であること、しかしそれ以外の多くはLinuxベースであるため、利用したい場合にはLinuxを使いこなせたほうがやはりよいということ。転写物の発現量推定もReXpress、RNA-Skimなどより便利かつ高速に実行できる時代がきていることなど。カウントデータ取得後の発現変動解析はedgeRやDESeqが有名だが、TCCは実質的にiterative edgeRやiterative DESeqに相当するものであり、compcodeRによる客観的な性能評価でも優れていることなど。性能評価に用いたRコードは20140722_compcodeR.txt。1時間分。

- **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, 東京大学(東京), 2014.07.02 ←

  内容: 教科書の3.3節と4.3節周辺。マッピングプログラムは大きくbowtieなどのbasic aligner (unspliced aligner)とtophatなどのsplice-aware aligner (spliced aligner)に大別されること。splice-aware alignerの基本的なイメージ。ゲノム配列既知の場合の遺伝子構造推定としてTophat-Cufflinksパイプラインの基本形を紹介。既知遺伝子(または転写物)の発現解析でよい場合は、トランスクリプトーム配列へのマッピングでよい。最近はSailfishやRNA-Skimなど、k-merに基づくalignment-freeな方法が注目されていることなど。研究目的別留意点として、遺伝子間比較の場合とサンプル間比較の場合、配列長補正、総リード数補正、RPKMなど。長い転写物ほどマップされるリード数が多い傾向をRで確認。GSE42212のヒトRNA-seqデータのFASTQファイル取得以降の一通りの解析。実際に行ったのは、カウントデータ取得以降のTCCパッケージを用いたサンプル間クラスタリング、発現変動遺伝子(DEG)同定。M-A plotのおさらい。結果の解釈。FDR、分布やモデルの説明。倍率変化でDEG同定を行う場合との比較。2コマ(2×90 min)分。

- **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, 東京大学(東京), 2014.06.25 ←

  内容: 教科書の2.3節が中心。デノボゲノムアセンブリ(de novo genome assembly)の大まかな手順を説明。基本的なテクニックとしてk-merの基本的な考え方をシミュレーションデータで解析するとともに、フィルタリング、エラー補正、de Bruijnグラフに利用されることをKmerGenieとPlatanus論文の図を利用して紹介。デノボトランスクリプトームアセンブリ(de novo transcriptome assembly)に応用できるものとできないもの。PacBioでのトランスクリプトーム配列決定が出てきていることを踏まえた未来予想図。QuasRを用いたマッピングの基本。BAMやBEDの出力ファイル形式。bowtie利用時の使用したオプションとマッピング結果の解釈。リアルsmall RNA-seqデータのカイコゲノムへのマッピング。アダプター配列除去前後のマップされたリード数の比較。結果の解釈とsequence logosでの確認など。マッピング結果からのカウントデータ取得。複数サンプルの場合に領域が変化するという事実を見せることを通じて、Tophat-Cufflinksパイプライン中のCuffmergeの直観的理解を補助。2コマ(2×90 min)分。

- **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:トップページへ報科学特論I, 東京大学(東京), 2014.06.18 ←

イントロダクション

これら3科目の講義資料を順番にみていくとよい

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、k-mer、アセンブリ、マッピング、RPKM、発現変動など。
実施日:2014.06.18、2014.06.25、2014.07.02

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:2014.05.14、2014.05.21、2014.05.28、2014.06.04

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:2014.04.09、2014.04.23、2014.04.30

イントロダクション

(RobLoxBioC)の紹介および結果が変わらないことの確認までをやってもらった。2コマ(2×90 min)分。

- 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:ゲノム情報解析基礎, 東京大学(東京), 2014.04.30
  内容: Rで塩基配列解析を行うための基本的なところ。例題としてシロイヌナズナゲノムのCpG出現頻度を解析し考察。Rパッケージのインストール、エラーメッセージへの対処法、利用可能な関数の概観。sequence logosを主な講義内容とし、エントロピー計算や、なぜエントロピーをそのまま利用せずに情報量に変換するかの意義。subseq関数のオプションをうまく利用して効率的に目的のプロモーター配列領域を切り出して計算するやり方など。課題4はプログラムの一部を任意に変更する基礎的な能力を問うもの。他の例題の中に回答が存在するので、それを効率的に見つける能力を見ている。講義自体はスライド39までで、スライド40以降はうまくいかないこともあるという事例やRのバージョンの違いに気をつけろ的な話。「農学生命情報科学特論I」で改めて話す予定。1コマ(90 min)分。
- 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:ゲノム情報解析基礎, 東京大学(東京), 2014.04.23
  内容: Rで塩基配列解析を行うための基本的なところ。初心者が犯しがちなミス、プログラムの中身の説明、アノテーションファイルやmulti-FASTAファイルからの情報抽出、意図的にエラーを出させてエラーへの対処能力向上、GC含量計算やそのプログラム内部の説明、ヒトゲノムのCpG出現頻度を解析するための連続塩基出現頻度解析、BSgenomeパッケージとか。課題は、自分が解析したい入力ファイルの全体像を把握し、適切な列およびキーワードで効率よく情報収集するための練習問題レベルのものにしている。Rがいかに簡単であるかをわかってもらうことに重点を置いている。ただし、ヘッダー行でひっかけを作っており、目で見て明らかに回答がわかっている状況下でそれを正しく判断し適切なテンプレートプログラムを利用できるかを問うている。また、課題2では、ゲノム配列にもバージョンがあるということを認識してもらう。2コマ(2×90 min)分。
- 門田幸二,「ウェブページと講義資料」, 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:ゲノム情報解析基礎, 東京大学(東京), 2014.04.09
  内容: 初心者向けバイオインフォマティクス全般およびゲノム情報解析系のイントロダクションの話。Rのイントロダクションやこのウェブページの簡単な使い方を含む。1コマ(90 min)分。
- 門田幸二,「比較トランスクリプトーム解析とその周辺:モデル、正規化、発現変動検出など」, よく分かる次世代シークエンサー解析ワークショップ, 九州大学(福岡), 2014.03.19
  内容: 初心者向けRNA-seqの話。主にカウントデータ取得以降の話。シリーズ最終日の統計

たとえば「ゲノム情報解析基礎」はこれらを参照。

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、k-mer、アセンブリ、マッピング、RPKM、発現変動など。
実施日:2014.06.18、2014.06.25、2014.07.02

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:2014.05.14、2014.05.21、2014.05.28、2014.06.04

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:2014.04.09、2014.04.23、2014.04.30

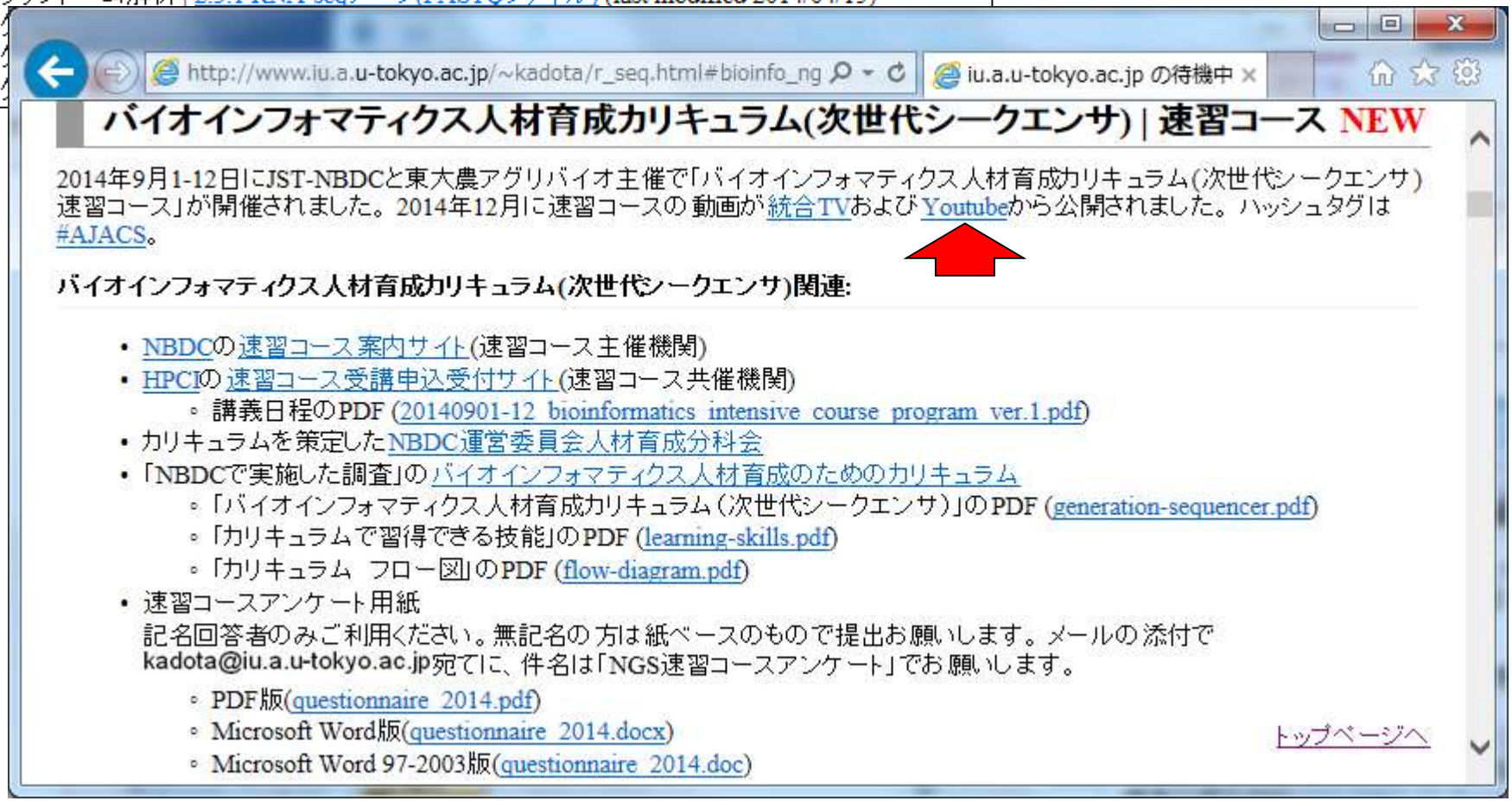
イントロダクション

2014年9月1-12日に開催された「NGS速習コース」にもR関連項目あり。

イントロダクション

全部で10日間のうち、R関連項目は2日分。講義映像もYoutubeから公開されている。

- 過去のお知らせ (last modified 2015/02/08) NEW
- Rのインストールと起動 (last modified 2014/09/08)
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (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
- 書籍 | トランス
- 書籍 | トランス

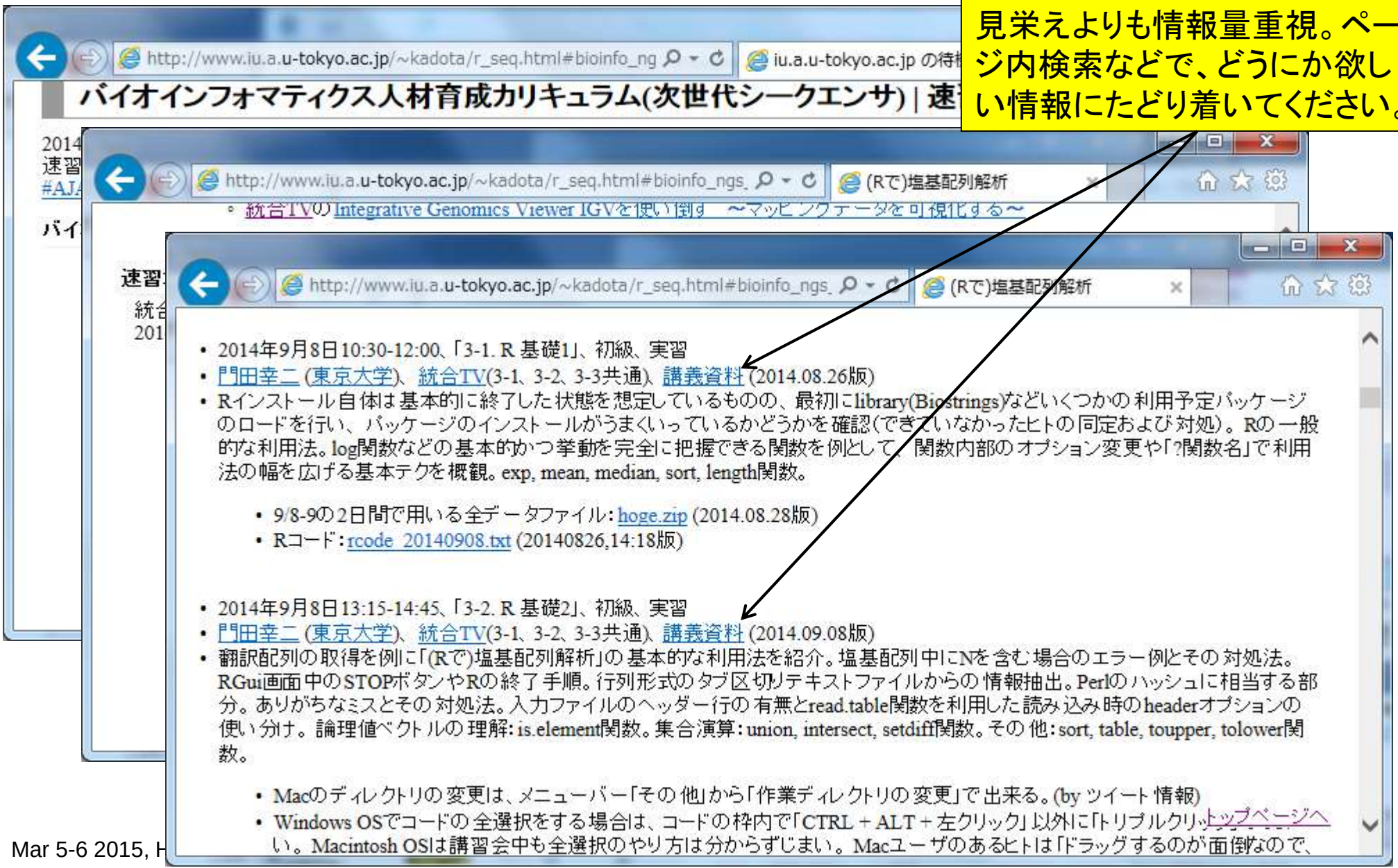


イントロダクション

全部で10日間のうち、R関連項目は2日分。統計部分を含まないので、基礎的な部分をみっちり勉強したい場合はこちらの視聴をおすすめ。

イントロダクション

講義資料PDFもページ下部に移動することで取得可能ではある。見づらいことは承知してますが、見栄えよりも情報量重視。ページ内検索などで、どうにか欲しい情報にたどり着いてください。



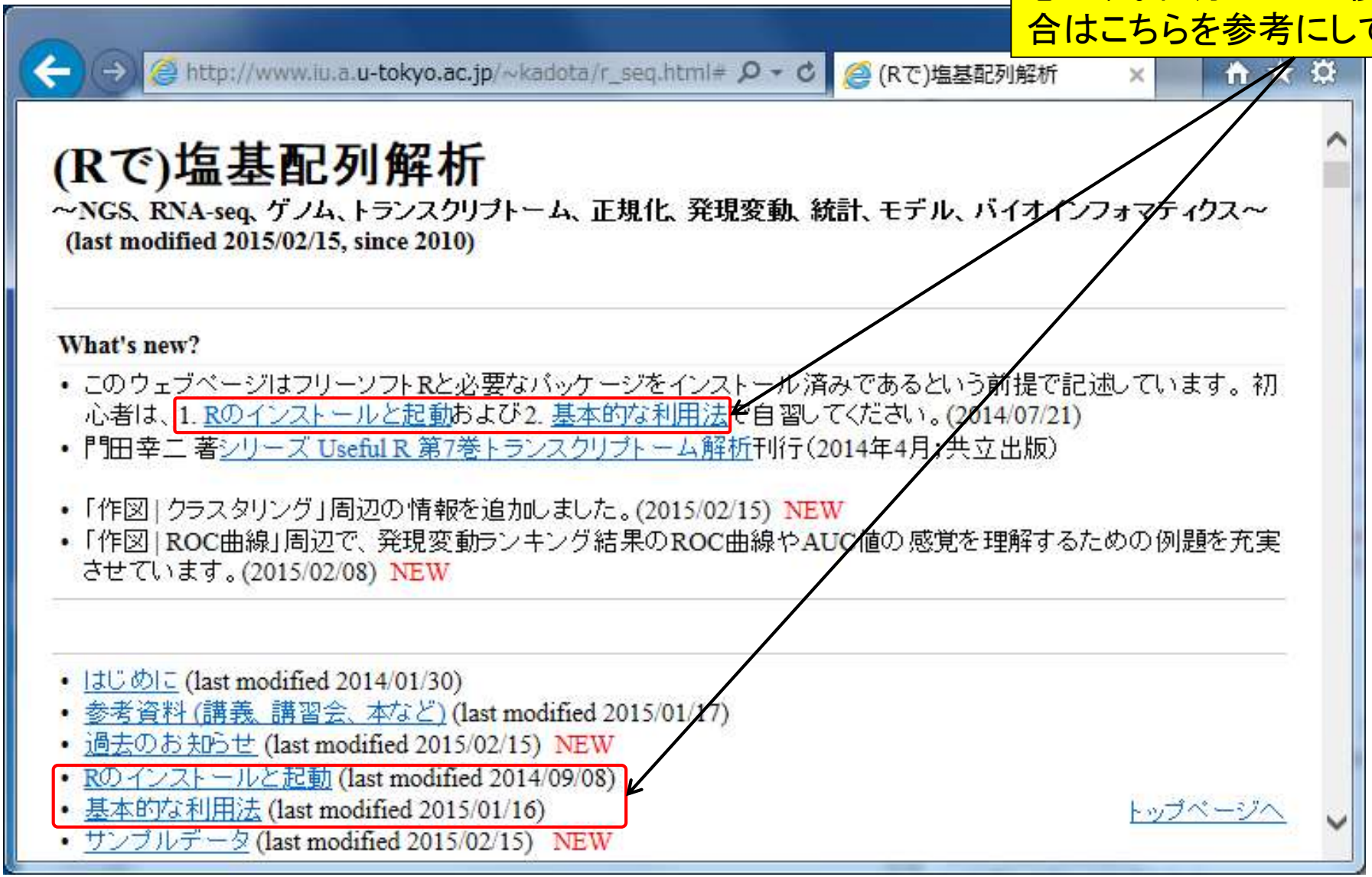
教育は大衆迎合型であるべし

求められていることに対応しきれなくなっています。ご新規様の講演や執筆依頼はお控えくださいますようお願い申し上げます。

- 平成25年度(2014年3月)HPCI講習会受講生の要望
  - マイクロアレイやコードの中身についての詳細情報を詳しく知りたい
    → 2014年10月の「中級者向けバイオインフォマティクス入門講習会(HPCI主催)」
  - 2日間くらい(平成25年度は1日のみ)使ってもっとガッツリやってほしい
    → 平成26年度は2日間に倍増
- 2014年9月の「NGS速習コース」受講生の要望
  - 統計解析(発現変動解析)についてもやってほしい。作図も。
    → HPCI講習会で例年統計解析の基本的な考え方を含めて行っている。平成26年度HPCI講習会(2015年3月)で需要を満たせると思っていたが。。。

イントロダクション

実習用PC環境は、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こちらを参考にしてください。



Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

塩基配列解析基礎（その1）

塩基配列を入力として、その翻訳されたアミノ酸配列を取得することができます。（2014年9月のNGS速習コース 3-2とほぼ同内容）

- イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2013/06/14)
- イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- イントロ | 一般 | 指定したID(染色体やdescri…n)の配列を取得 (last modified 2014/03/10)
- イントロ | 一般 | 翻訳配列(translate)を取得 …modified 2014/08/13)
- イントロ | 一般 | 相補鎖(complement)を取得 …st modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(rever…
- イントロ | 一般 | 逆鎖(reverse)を…
- イントロ | 一般 | 2連続塩基の出…
- イントロ | 一般 | 3連続塩基の出…
- イントロ | 一般 | 任意の長さの連…
- イントロ | 一般 | Tips | 任意の拡…

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

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

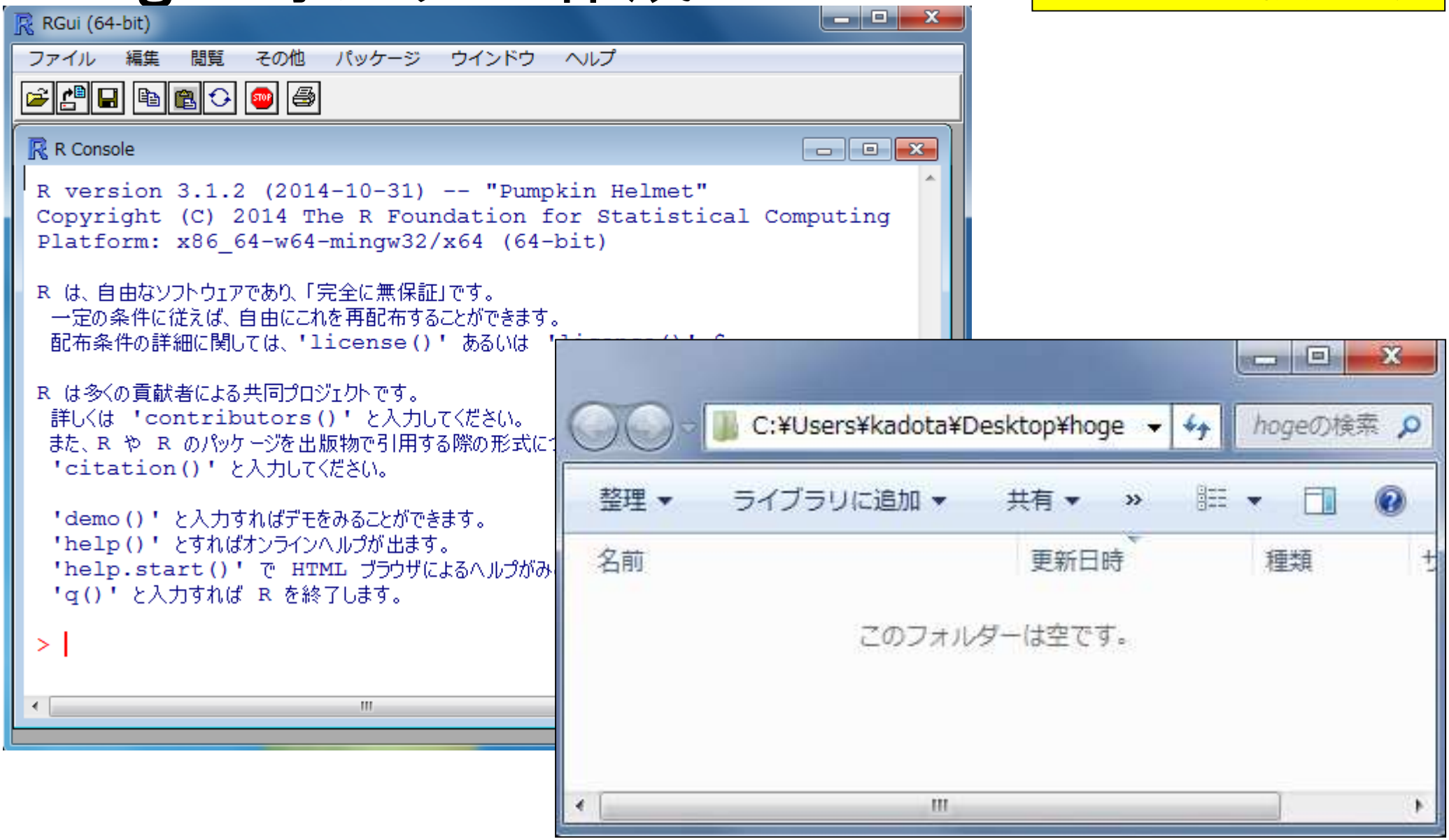
1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

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

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

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

#本番
fasta <- translate(fasta)                  #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                      #確認してるだけです

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

hogeフォルダの作成

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



イントロ | 一般 | 翻訳配列(translate)を取得 NEW

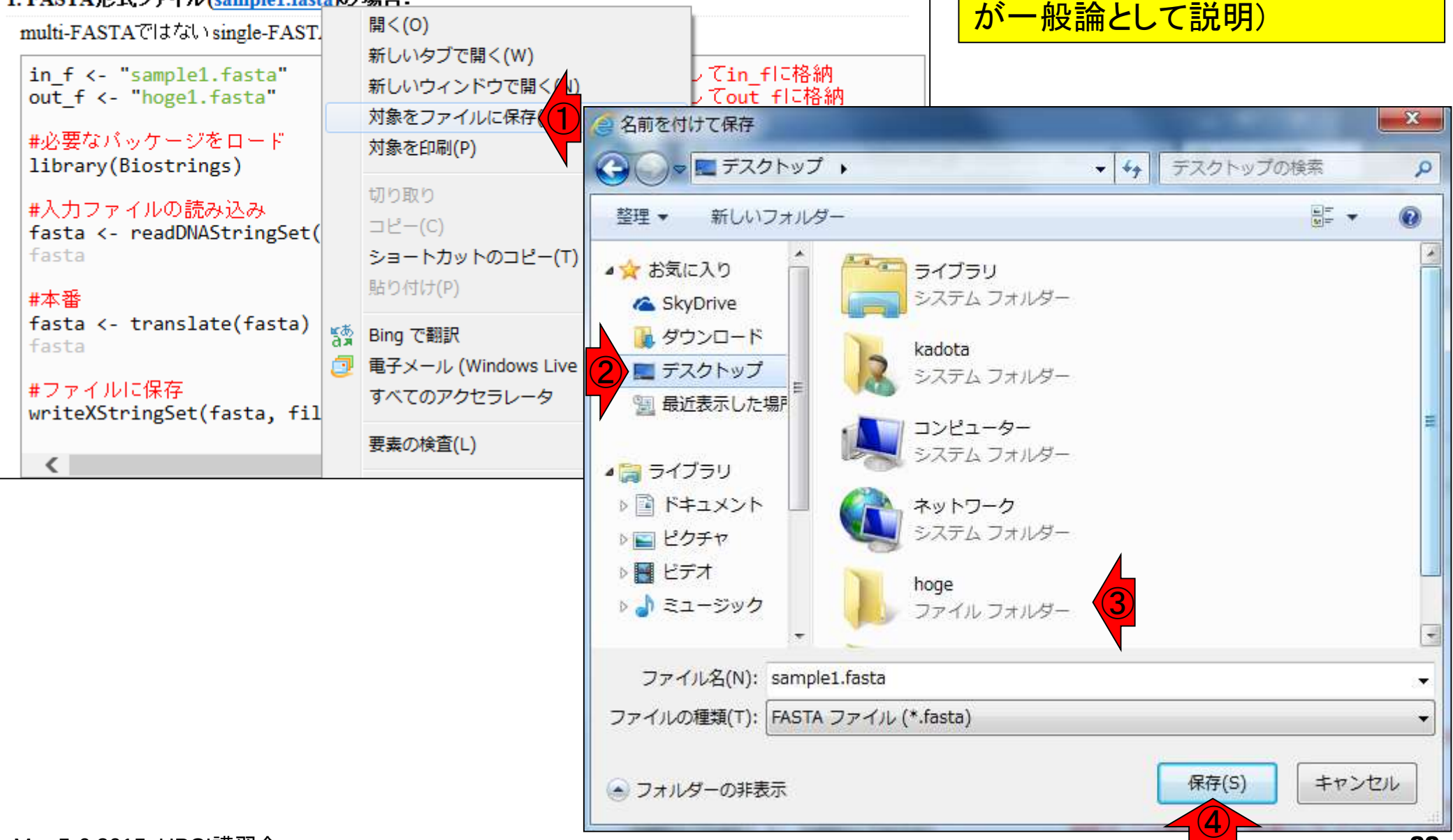
塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

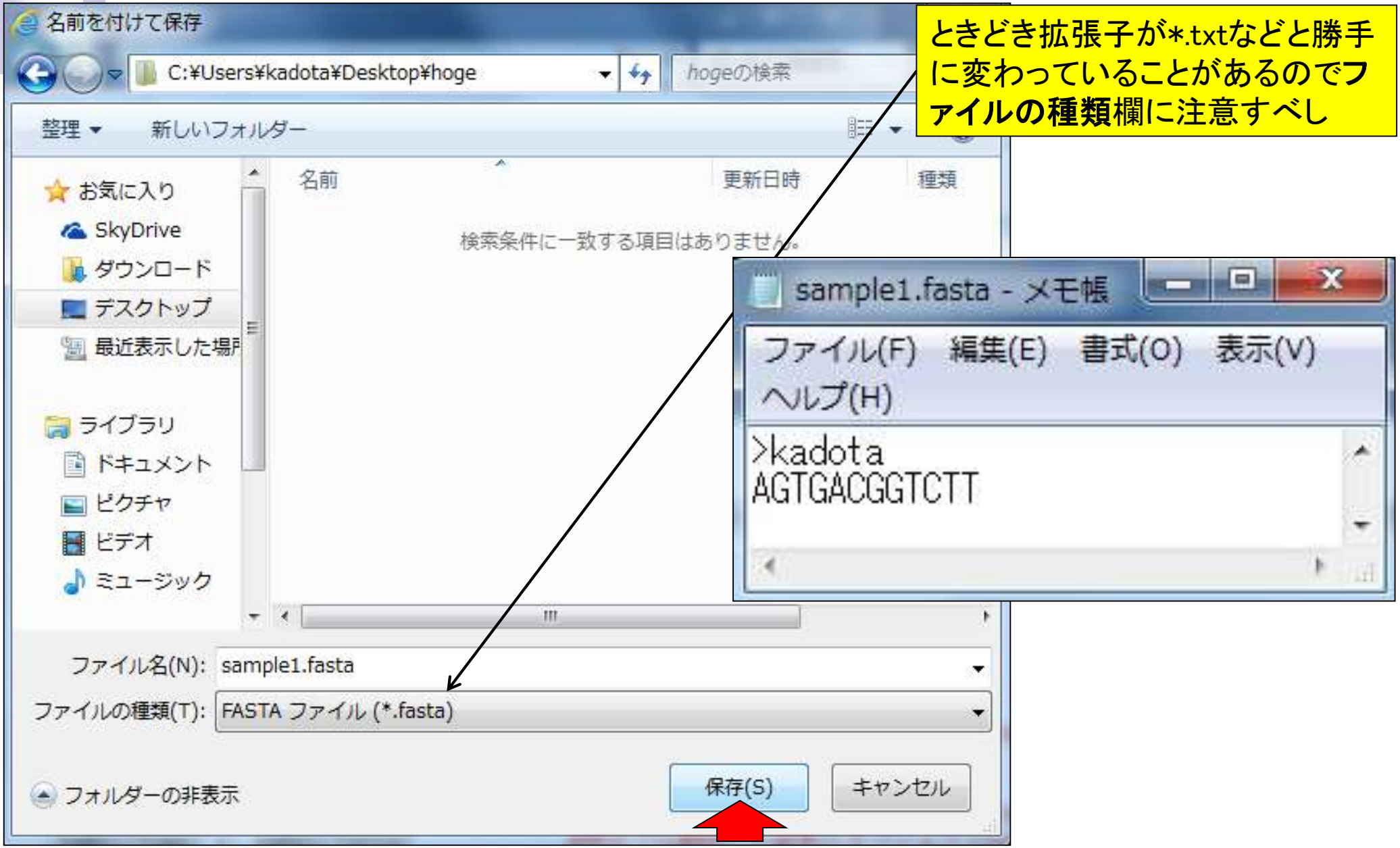
1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FAST

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

#必要なパッケージをロード
library(Biostrings)

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

#本番
fasta <- translate(fasta)
fasta

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

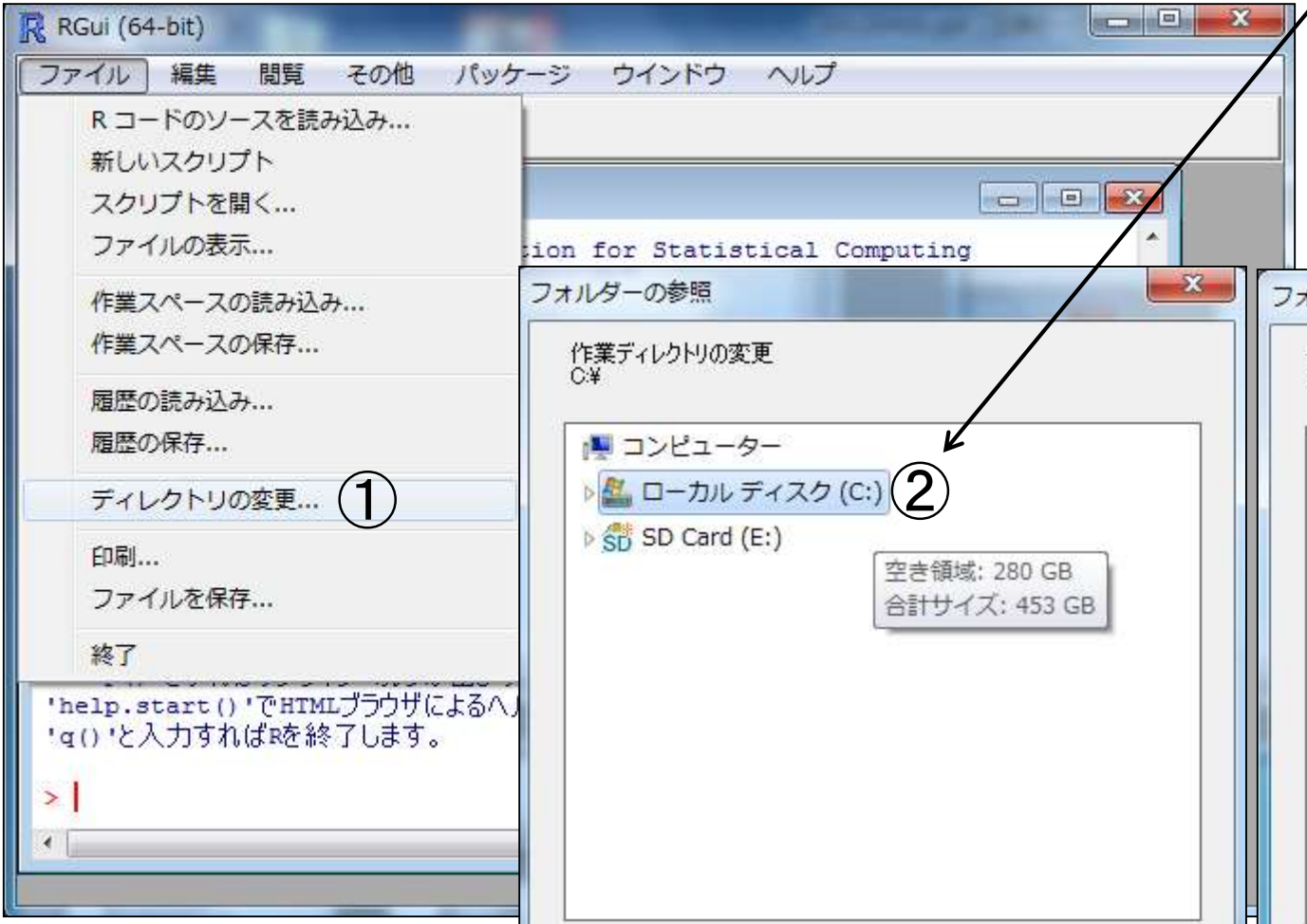
解析したいファイルsample1.fastaをhogeフォルダ中に保存(本当は既にhogeフォルダ中に存在するが一般論として説明)

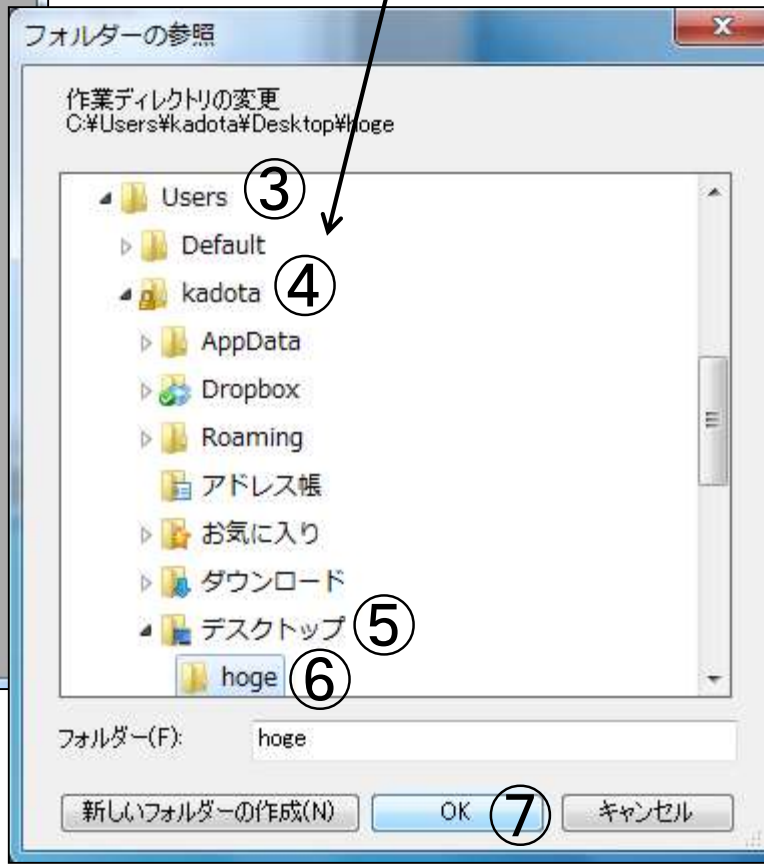



名前を付けて保存
C:¥Users¥kadota¥Desktop¥hoge
hogeの検索
整理
新しいフォルダー
お気に入り
SkyDrive
ダウンロード
デスクトップ
最近表示した場所
ライブラリ
ドキュメント
ピクチャ
ビデオ
ミュージック
名前
更新日時
種類
検索条件に一致する項目はありません。
ファイル名(N): sample1.fasta
ファイルの種類(T): FASTA ファイル (*.fasta)
フォルダーの非表示
保存(S)
キャンセル
sample1.fasta - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)
ヘルプ(H)
>kadota
AGTGACGGTCTT
ときどき拡張子が*.txtなどと勝手に変わっていることがあるのでファイルの種類欄に注意すべし

作業ディレクトリの変更

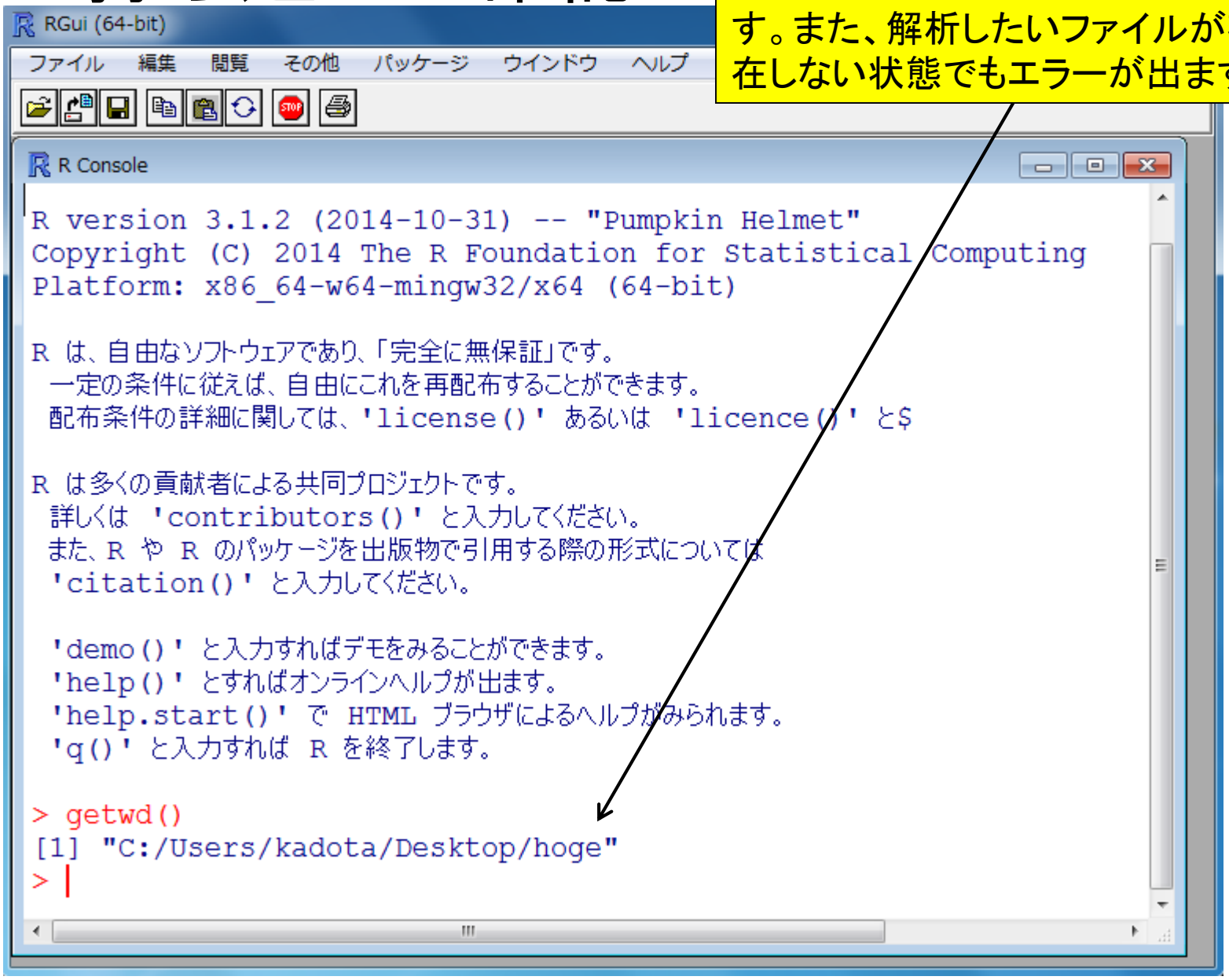
②「Windows(C:)」となっている場合もあるが、気にしない。④はヒトによって違う。





getwd()と打ち込んで確認

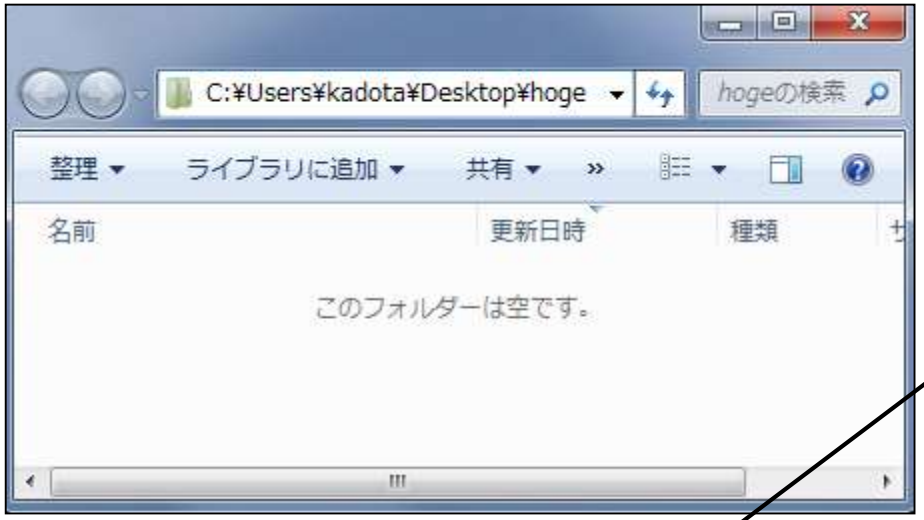
当たり前ですが、解析したいディレクトリ（またはフォルダ）を正しく指定できていなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます。



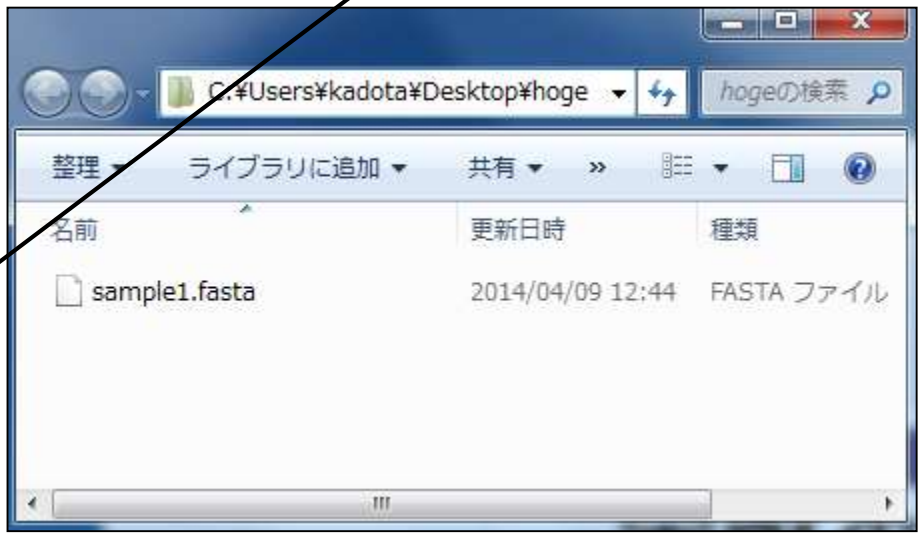
character(0)は何もないという意味

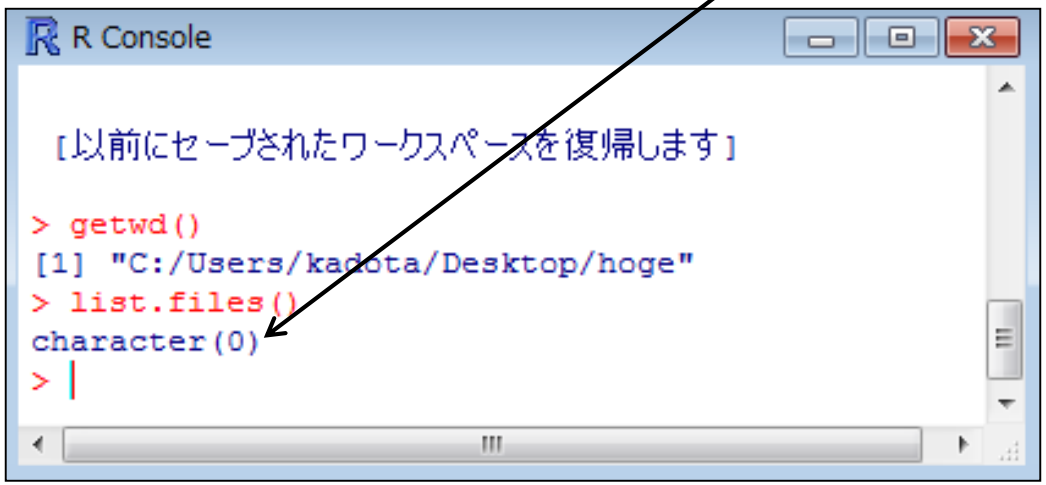
実際のhogeフォルダとR操作画面の関係

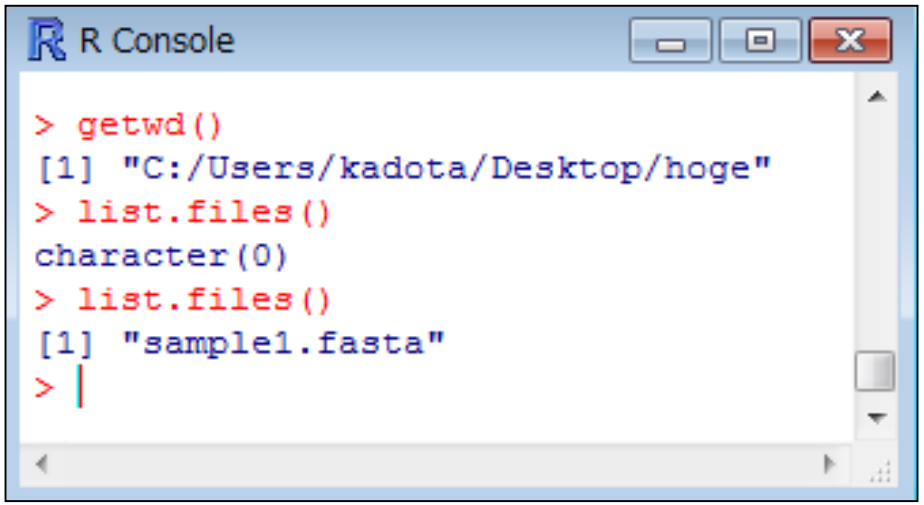
ファイル保存前



ファイル保存後







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。
Macintoshはよくわかりません。

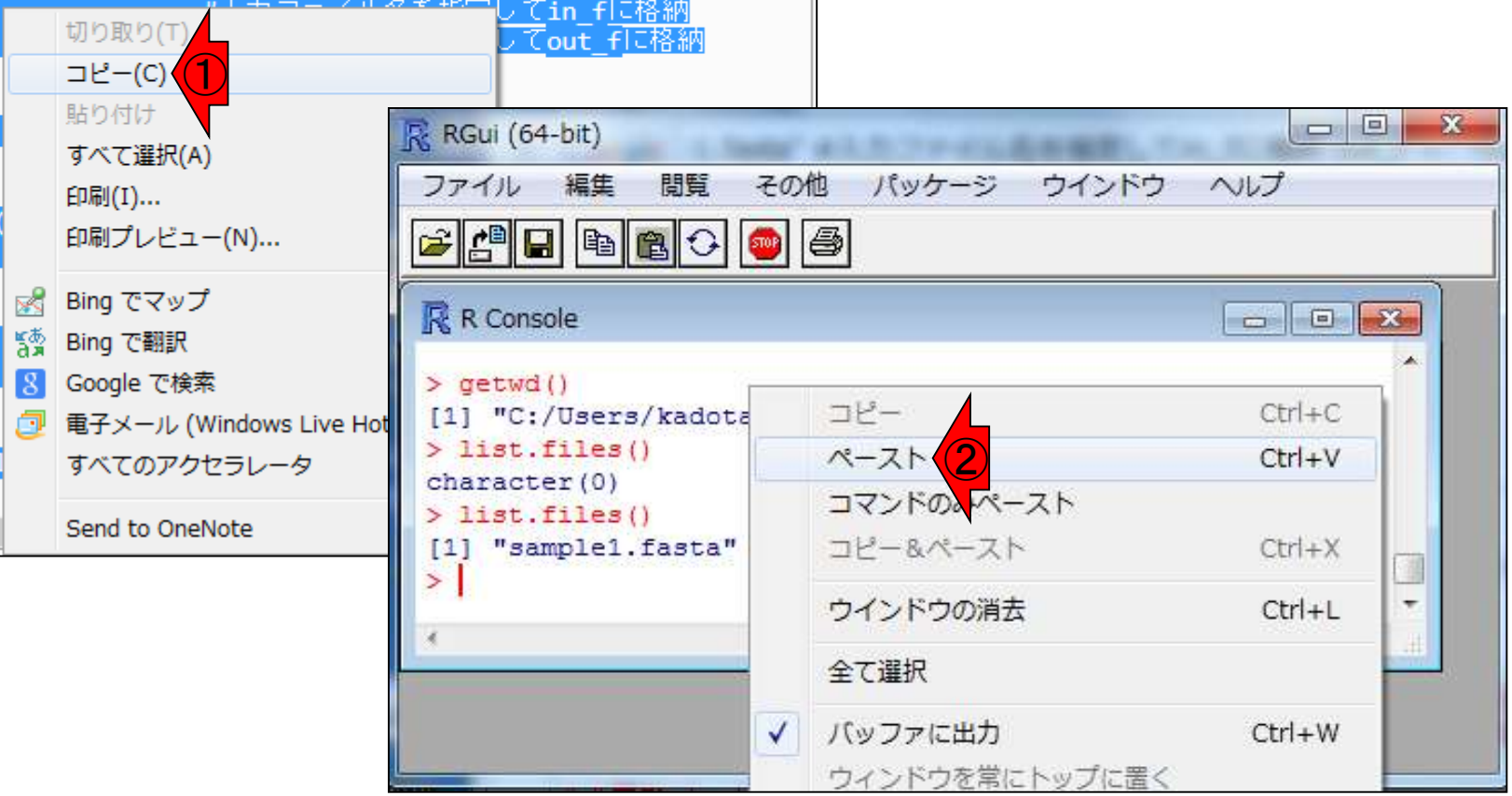
イントロ | 一般 | 翻訳配列(translate)を取得 NEW

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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

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

#必要なパッケージをロード
library(Biostrings)

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

#本番
fasta <- translate(fasta)
fasta

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



基本はコピペ

イントロ | 一般 | 翻訳配列(translate)を取得 NE

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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではない single-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"                    #入力ファイル名
out_f <- "hoge1.fasta"                     #出力ファイル名

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

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

#本番
fasta <- translate(fasta)                  #アミノ酸配列に
fasta                                      #確認してるだけ

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

エラーなく実行できた場合の全貌

R Console

```
> in_f <- "sample1.fasta"
> out_f <- "hoge1.fasta"                   #出力ファイル名を指定してout_fに格納
>
> #必要なパッケージをロード
> library(Biostrings)                      #パッケージの読み込み
 要求されたパッケージ BiocGenerics をロード中です
 要求されたパッケージ parallel をロード中です

 次のパッケージを付け加えます: 'BiocGenerics'

The following objects are masked from 'package:parallel':

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

The following object is masked from 'package:stats':

    xtabs

The following objects are masked from 'package:base':

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

 要求されたパッケージ IRanges をロード中です
 要求されたパッケージ XVector をロード中です
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読$
> fasta                                    #確認してるだけです
  A DNAStringSet instance of length 1
    width seq                                              names
[1]    12 AGTGACGGTCTT                                     kadota
>
> #本番
> fasta <- translate(fasta)                #アミノ酸配列に翻訳した結果をfasta$
> fasta                                    #確認してるだけです
  A AAStringSet instance of length 1
    width seq                                              names
[1]     4 SDGL                                             kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を$
>
```

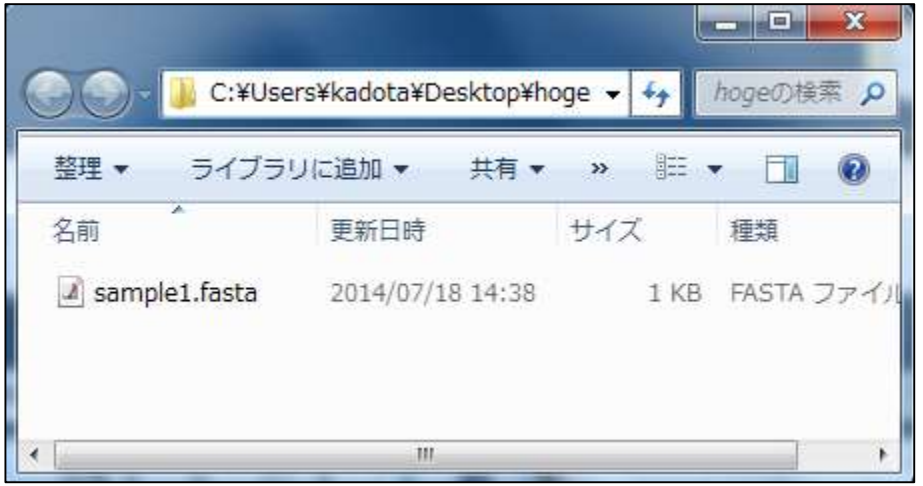
実行結果

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

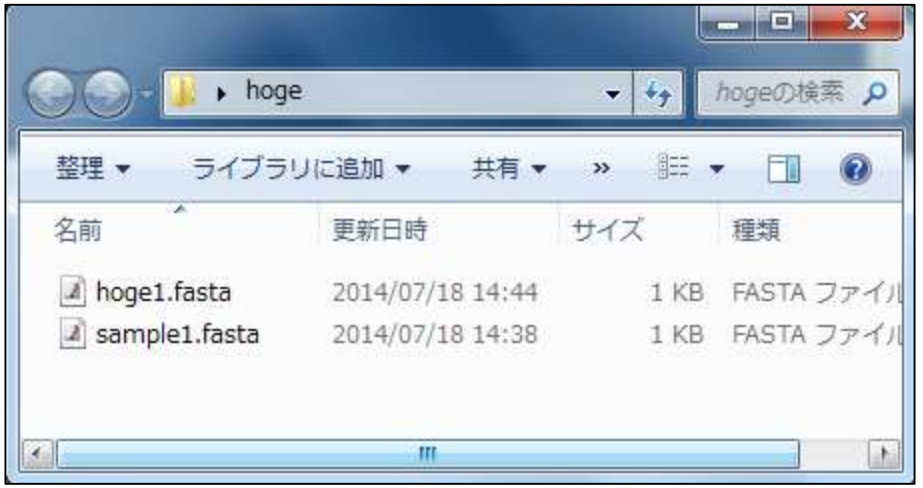
R Console

```
> fasta <- readDNAStringSet(in_f, format="fasta")#$
> fasta                                          #確認して$
  A DNAStringSet instance of length 1
    width seq                    names
[1]    12 AGTGACGGTCTT           kadota
>
> #本番
> fasta <- translate(fasta)                      #アミノ酸$
> fasta                                          #確認して$
  A AAStringSet instance of length 1
    width seq                    names
[1]     4 SDGL                   kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta$
>
```

実行前のhogeフォルダ



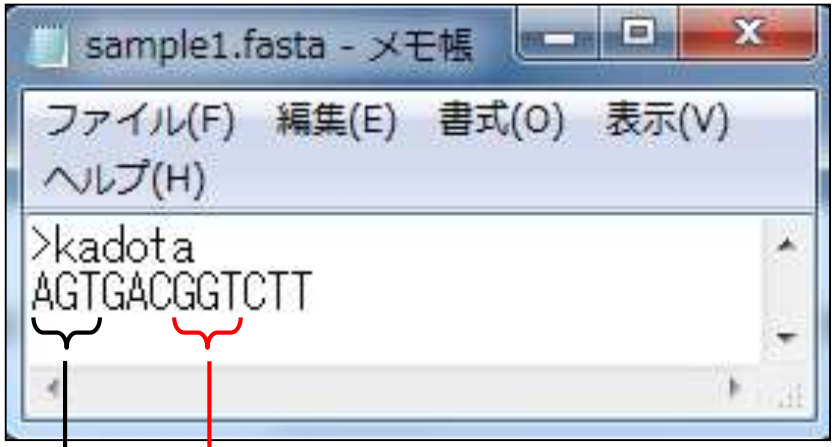
実行後のhogeフォルダ



実行結果

入力ファイル中の塩基配列は、3の倍数の12塩基長、ACGTのみからなるので何のエラーも出ない

入力：塩基配列ファイル（sample1.fasta）



出力：アミノ酸配列ファイル（hoge1.fasta）

コドン表

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

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

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

塩基配列解析基礎(その2)

multi-FASTAファイルの場合も基本形は同じ。基本形のみではエラーが出る場合を示し、対処法を学ぶ。入力ファイル(sample4.fasta)をダウンロードして実行してみよう。(2014年9月のNGS速習コース 3-2とほぼ同じ内容)

- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- イントロ | 一般 | 指定したID(染色体やdescri…n)の配列を取得 (last modified 2014/03/10)
- イントロ | 一般 | 翻訳配列(translate)を取得 …modified 2014/08/13)
- イントロ | 一般 | 相補鎖(complement)を取得 …st modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "…mple1.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <-

#必要なパッケ
library(Bios

#入力ファイル
fasta <- rea
fasta

#本番
fasta <- tra
fasta

#ファイルに保
writeXString
```

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

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta"                   #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"                    #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)                 #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                     #確認してるだけです

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

塩基配列解析基礎(その2)

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

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

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"                     #出力ファイル名を指定してout_fに格納

#必要なパッケージをロード
library(Biostrings)

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

#本番
fasta <- translate(fasta)
fasta

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

R Console

```
> #本番
> fasta <- translate(fasta)                #アミノ酸配列に翻訳した$
 以下にエラー .Call2("DNAStringSet_translate", x, skip_code, dna$
  in 'x[[4]]': not a base at pos 17
> fasta                                    #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                        names
[1]    21 CGACAGCTCCTCGGCATCCGA                      gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                      gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG             gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG             gene_5
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#f$
>
```

実行結果

各項目の例題は1つとは限りません。例えば2.は出力ファイルを盲目的に信じてはいけない、という典型例。目的は翻訳配列の取得で、得られた結果は塩基配列。

入力：sample4.fasta

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

出力：hoge2.fasta

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

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

エラーの原因は、gene_4の17番目のポジションがNであることに由来。

R Console

```
> #本番
> fasta <- translate(fasta)                    #アミノ酸配列に翻訳した$
 以下にエラー.Call2("DNAStringSet_translate", x, skip_code, dna$
  in 'x[[4]]': not a base at pos 17
> fasta                                        #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                  names
[1]    21 CGACAGCTCCTCGGCATCCGA                gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT          gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG       gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG       gene_5
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#f$
>
```

塩基配列解析基礎(その2)

2.のエラー対策の一つは、Nを含む配列を除くこと。3.はそれを実現したコードです。

- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- イントロ | 一般 | 指定したID(染色体やdescri…n)の配列を取得 (last modified 2014/03/10)
- イントロ | 一般 | 翻訳配列(translate)を取得 …modified 2014/08/13)
- イントロ | 一般 | 相補鎖(complement)を取得 …st modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-…

```
in_f <-
out_f <

#必要な…
library

#入力フ
fasta <
fasta

#本番
fasta <
fasta

#ファイ…
writeXS
```

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

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

```
in_f <- "sample4.fasta"                 #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fasta"                  #出力ファイル名を指定してout_fに格納

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

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

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

#本番
fasta <- translate(fasta)               #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                   #確認してるだけです

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

塩基配列解析基礎(その2)

全選択はCTRLとALTキーを押しながらコードの枠内で左クリック。

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

エラーへの対策として、ACGTのみからなる配列を抽出したサブセットを抽出しています。翻訳はそれらのサブセットのみに対して行っているので「文字が塩基ではない」という類のエラーは出ません。ただ、終止コドン(stop codon)を表すようですね。

```
in_f <- "sample4.fasta"                 #
out_f <- "hoge3.fasta"                  #

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

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(fasta)
obj <- (width(fasta) == hoge)           #
fasta <- fasta[obj]                     #
fasta                                   #

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

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

R Console

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

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

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta"                 #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"                  #出力ファイル名を指定してout_fに格納

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

#入力ファイルの読み込
fasta <- readDNASt
fasta

#本番
fasta <- translate
fasta

#ファイルに保存
writeXStringSet(fa
```

2と3の違いは、3のコード中の黒枠部分が追加されただけ。

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

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

```
in_f <- "sample4.fasta"                 #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fasta"                  #出力ファイル名を指定してout_fに格納

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

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

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

#本番
fasta <- translate(fasta)               #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                   #確認してるだけです

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

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

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

```
in_f <- "sample4.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fasta"                     #出力ファイル名を指定してout_fに格納

#必要なパッケージをロード
library(Biostrings)

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
fasta

#本番
fasta <- translate(fasta)
fasta

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

このウェブページでは、条件判定を行って得られた論理値ベクトルをobjというオブジェクト名で統一的に取り扱っています。

R Console

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定した$
> fasta                                        #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                          names
[1]    21 CGACAGCTCCTCGGCATCCGA                        gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                  gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                        gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG               gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG               gene_5
>
> #前処理(ACGTのみからなる配列を抽出)
> hoge <- rowSums(alphabetFrequency(fasta)[,1:4])#A,C,G,T,..の数$
> obj <- (width(fasta) == hoge)                #条件を満たすかどうかを$
> fasta <- fasta[obj]                          #objがTRUEとなる要素のみ$
> fasta                                        #確認してるだけです
  A DNAStringSet instance of length 4
    width seq                                          names
[1]    21 CGACAGCTCCTCGGCATCCGA                        gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                  gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                        gene_3
[4]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG               gene_5
>
```

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

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

```
in_f <- "sample4.fasta"
out_f <- "hoge3.fasta"

#必要なパッケージをロード
library(Biostrings)

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
fasta

#本番
fasta <- translate(fasta)
fasta

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

ACGTのみからなるサブセットに対して、translate関数を適用しているのでエラーが出ない。

R Console

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

実行結果

エラーの原因であった17番目のポジションにNを含むgene_4が除かれて、うまく翻訳配列を出力できていることが分かる。尚、アスタリスク(*)は終止コドンを表すようです。

入力：sample4.fasta

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

出力：hoge3.fasta

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

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

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

塩基配列解析基礎(その2)

サブセット(ACGTのみからなる配列)の抽出の中身を説明します。

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

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

```
in_f <- "sample4.fasta"                 #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fasta"                  #出力ファイル名を指定してout_fに格納

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

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

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

#本番
fasta <- translate(fasta)               #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                   #確認してるだけです

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

塩基配列解析基礎(その2)

fastaオブジェクトの中身が変遷している点に注意。

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

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

```
in_f <- "sample4.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fasta"                     #出力ファイル名を指定してout_fに格納

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

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(fasta)[,1:4])#A,C,G,T,..の数を配列ごとにカウントした結果をhogeに
obj <- (width(fasta) == hoge)              #条件を満た
fasta <- fasta[obj]                        #objがTRUE
fasta                                      #確認してる

#本番
fasta <- translate(fasta)                  #アミノ酸配
fasta                                      #確認してる

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

R Console

```
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", wi$
>
> fasta
  A AAStringSet instance of length 4
    width seq                                      names     $
[1]     7 RQLLGIR                                  gene_1
[2]     9 VCLKRPKWV                                gene_2
[3]     7 CRRRA*S                                  gene_3
[4]    10 RADSTQQ*TR                               gene_5
>
```

塩基配列解析基礎(その2)

入力ファイル読み込み直後の
fastaオブジェクトの中身を表示

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

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

```
in_f <- "sample4.fasta"                    #入力ファイルを指定してin_fに格納
out_f <- "hoge3.fasta"                     #出力ファイルを指定してout_fに格納

#必要なパッケージをロード
library(Biostrings)

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(fasta)[,c("A","C","G","T")])
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
fasta

#本番
fasta <- translate(fasta)
fasta

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta")#out_fで指定したファイルに書き出し
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)

R Console

```
    Reduce, rep.int, rownames, sapply, setdiff, sort, table, tapply,
    union, unique, unlist

 要求されたパッケージ IRanges をロード中です
 要求されたパッケージ XVector をロード中です
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読$
> fasta                                    #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                               names
[1]    21 CGACAGCTCCTCGGCATCCGA                             gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                       gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                             gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG                    gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG                    gene_5
>
```

塩基配列解析基礎(その2)

alphabetFrequency関数は塩基ごとの出現頻度を返す。gene_4の配列中にNが1つ存在することは出力結果からもわかる。

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

R Console

```
> fasta                                #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                                names  $
[1]    21 CGACAGCTCCTCGGCATCCGA                              gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                        gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                              gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG                     gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG                     gene_5
> alphabetFrequency(fasta)
     A C G T M R W S Y K V H D B N - + .
[1,] 4 9 5 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 4 9 8 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 6 2 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 7 9 8 5 0 0 0 0 0 0 0 0 0 0 1 0 0 0
[5,] 8 9 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
```

塩基配列解析基礎(その2)

dim関数は行列の行数と列数を返す。alphabetFrequency関数出力結果は5行×18列からなることが分かる。キーボードの上下キーを上手に利用して最小限の労力でキータイプ(あるいはコピペ)すべし！

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

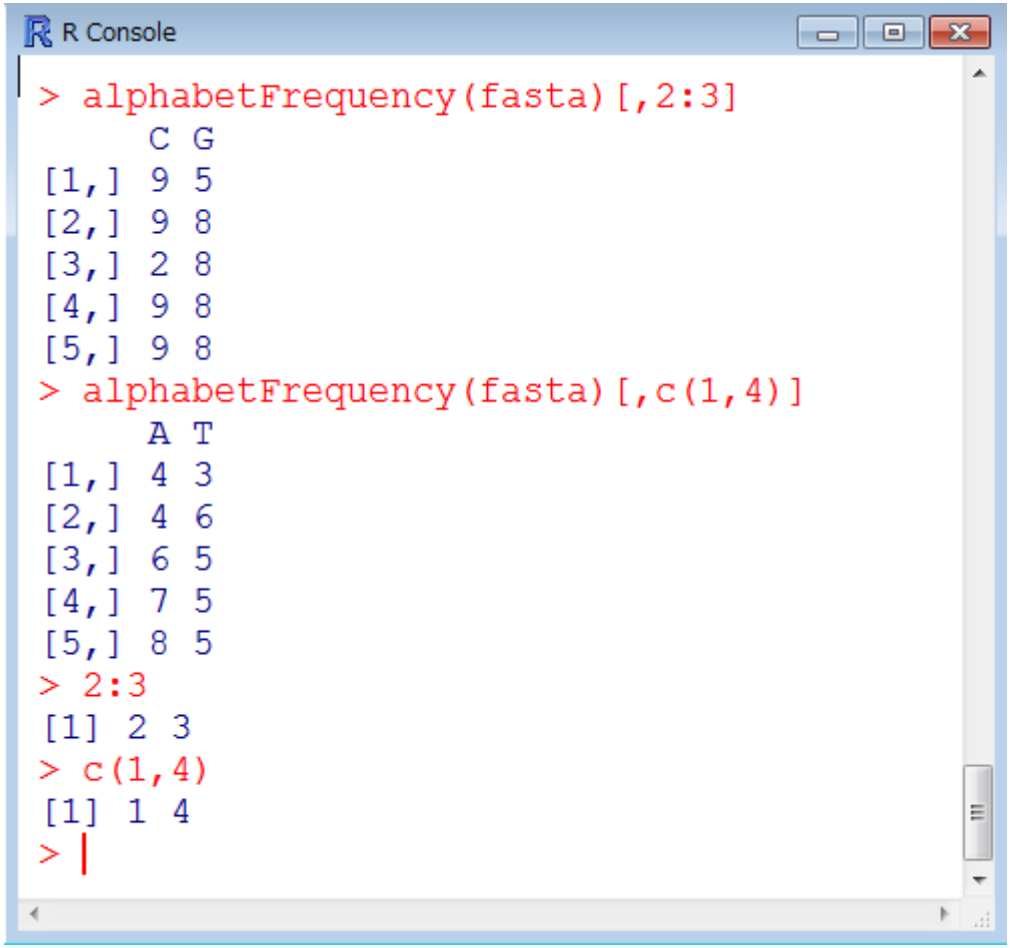
R Console

```
> alphabetFrequency(fasta)
     A C G T M R W S Y K V H D B N - + .
[1,] 4 9 5 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 4 9 8 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 6 2 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 7 9 8 5 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
[5,] 8 9 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dim(alphabetFrequency(fasta))
[1]  5 18
> alphabetFrequency(fasta)[,1:4]
     A C G T
[1,] 4 9 5 3
[2,] 4 9 8 6
[3,] 6 2 8 5
[4,] 7 9 8 5
[5,] 8 9 8 5
>
```

塩基配列解析基礎(その2)

任意のサブセットを取得可能。2:3やc(1,4)などをうまく利用すべし。

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



塩基配列解析基礎(その2)

もちろんこのように一旦ugeというオブジェクトに格納してから取り扱ってもよい。

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

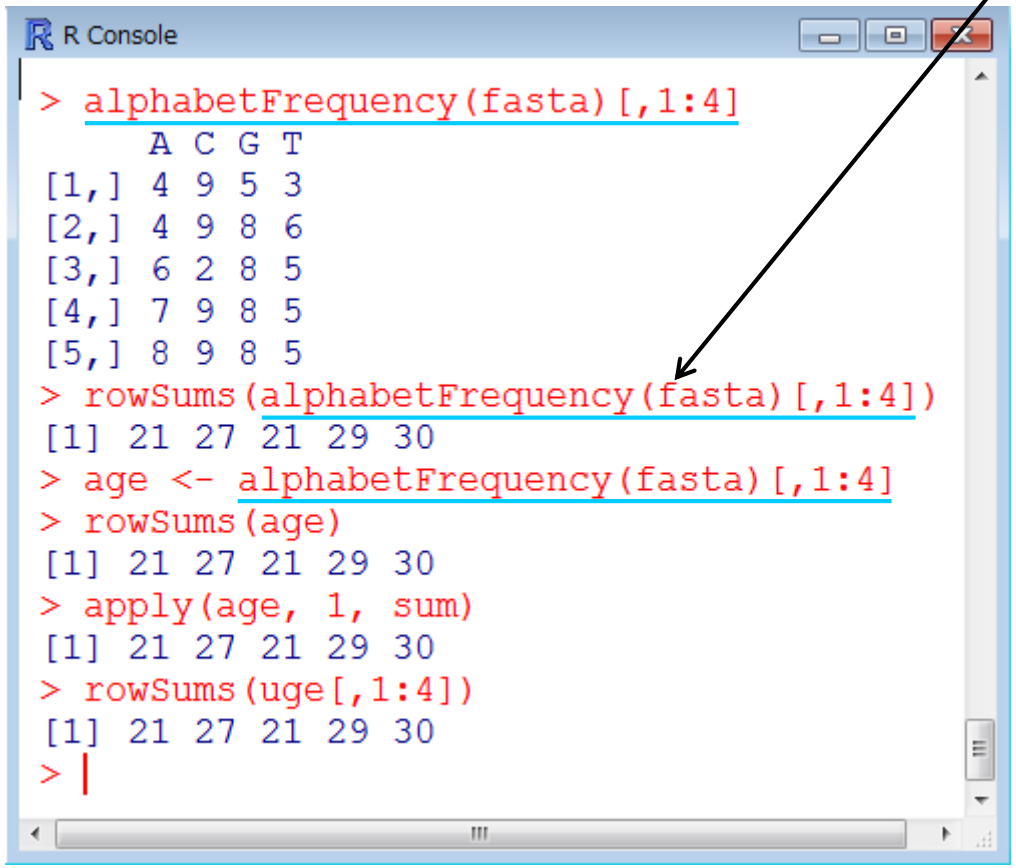
R Console

```
> uge <- alphabetFrequency(fasta)
> uge
     A C G T M R W S Y K V H D B N - + .
[1,] 4 9 5 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 4 9 8 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 6 2 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 7 9 8 5 0 0 0 0 0 0 0 0 0 0 1 0 0 0
[5,] 8 9 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> uge[,2:3]
     C G
[1,] 9 5
[2,] 9 8
[3,] 2 8
[4,] 9 8
[5,] 9 8
>
```

塩基配列解析基礎(その2)

rowSums関数は行ごとの和を返す。この場合は入力としてACGTのみの出現頻度情報を与えているので、ACGTのみの塩基数が返される。

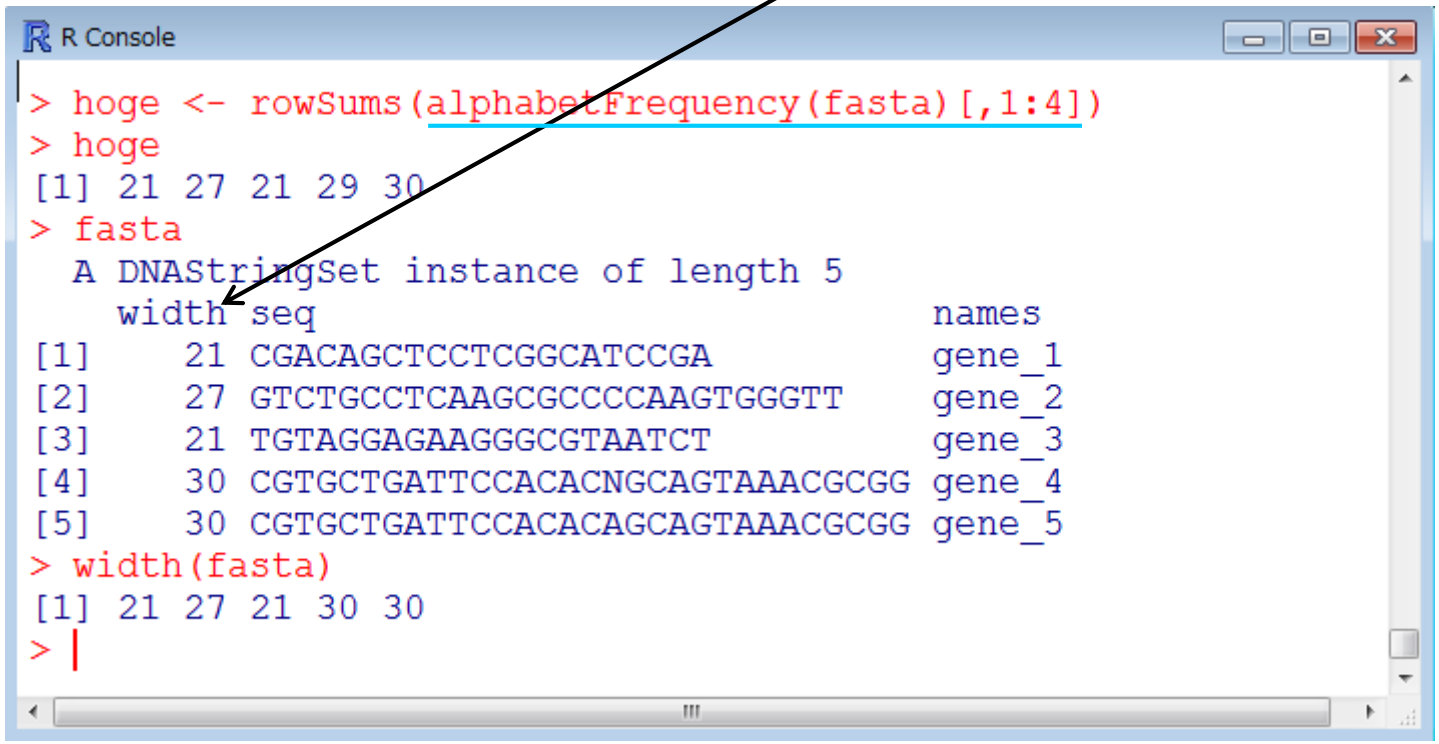
```
#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(fasta)[,1:4])#A,C,G,T,..の数を配列ごとにカウントした結果をhogeに格納
obj <- (width(fasta) == hoge)          #条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]                    #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                  #確認してるだけです
```



塩基配列解析基礎(その2)

fastaオブジェクトを表示させると、width, seq, namesという列名のようなものが見える。width列の数値は配列長情報に相当。

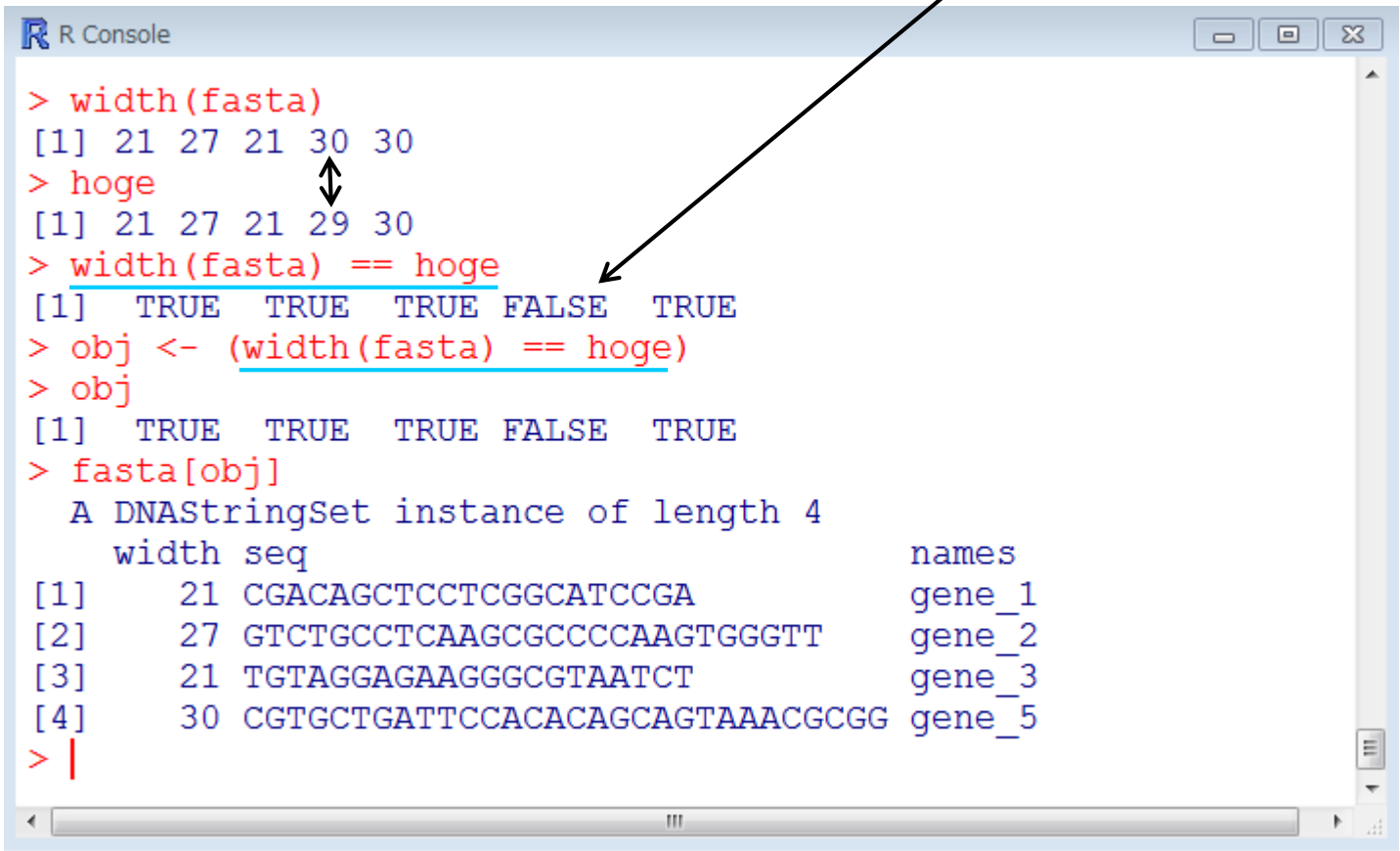
```
#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(fasta)[,1:4])#A,C,G,T,..の数を配列ごとにカウントした結果をhogeに格納
obj <- (width(fasta) == hoge)          #条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]                    #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                  #確認してるだけです
```



塩基配列解析基礎(その2)

配列長情報に相当するwidth(fasta)とACGTのみの塩基数に相当するhogeの数値を比較すれば、ACGT以外の文字を含む配列情報を浮かび上がらせることができる。

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



塩基配列解析基礎(その2)

同じような戦略で、例えば25塩基長以上の配列のみ抽出することもできる。

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

R Console

```
> fasta
  A DNAStringSet instance of length 5
    width seq                                names
[1]    21 CGACAGCTCCTCGGCATCCGA              gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT        gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT              gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG     gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG     gene_5
> width(fasta) >= 25
[1] FALSE  TRUE FALSE  TRUE  TRUE
> fasta[width(fasta) >= 25]
  A DNAStringSet instance of length 3
    width seq                                names
[1]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT        gene_2
[2]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG     gene_4
[3]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG     gene_5
>
```

このようなフィルタリングも可能。

塩基配列解析基礎(その2)

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

R Console

```
> fasta[1:2]
  A DNAStringSet instance of length 2
    width seq                                      names
[1]    21 CGACAGCTCCTCGGCATCCGA                    gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT              gene_2
> fasta[c(1,4)]
  A DNAStringSet instance of length 2
    width seq                                      names
[1]    21 CGACAGCTCCTCGGCATCCGA                    gene_1
[2]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG           gene_4
> fasta[c("gene_3","gene_5")]
  A DNAStringSet instance of length 2
    width seq                                      names
[1]    21 TGTAGGAGAAGGGCGTAATCT                    gene_3
[2]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG           gene_5
>
```

塩基配列解析基礎(その3)

「イントロ | 一般」や「前処理 | フィルタリング」は基本テクニックおよびその組合せになっています。

- イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2013/06/14)
- イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- イントロ | 一般 | 指定したID(染色体やdescription)の配列を取得 (last modified 2014/03/10)
- イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2015/02/17) NEW
- イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 逆鎖(reverse)を取得 (last modified 2013/06/14)

- 前処理 | フィルタリング | ACGTのみからなる配列を抽出 (last modified 2014/08/04)
- 前処理 | フィルタリング | ACGT以外のcharacter"-"をNに変換 (last modified 2013/06/18)
- 前処理 | フィルタリング | ACGT以外の文字数が閾値以下の配列を抽出 (last modified 2013/09/27)
- 前処理 | フィルタリング | 重複のない配列セットを作成 (last modified 2013/06/18)
- 前処理 | フィルタリング | 指定した長さ以上の配列を抽出 (last modified 2014/02/07)
- 前処理 | フィルタリング | 任意のリード(サブセット)を抽出 (last modified 2014/08/21)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | 任意のIDを含む配列を抽出 (last modified 2013/06/18)

塩基配列解析基礎(その3)

multi-FASTA形式ファイルを自在に取り扱えるのはBiostringsパッケージのおかげです。

- 前処理 | フィルタリング | ACGTのみからなる配列を抽出 (last modified 2014/08/04)
- 前処理 | フィルタリング | ACGT以外のcharacter"-"をNに変換 (last modified 2013/06/18)
- 前処理 | フィルタリング | ACGT以外の文字数が閾値以下の配列を抽出 (last modified 2013/09/27)
- 前処理 | フィルタリング | 重複のない配列セットを作成 (last modified 2013/06/18)
- 前処理 | フィルタリング | 指定した長さ以上の配列を抽出 (last modified 2014/02/07)
- 前処理 | フィルタリング | 任意のリード(サブセット)を抽出 (last modified 2014/08/21)
- 前処理 | フィルタリング | 指定した
- 前処理 | フィルタリング | 任意のII

前処理 | フィルタリング | 指定した長さ以上の配列を抽出

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

1. multi-FASTAファイル(hoge4.fa)の場合:

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

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

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

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

#本番
obj <- as.logical(width(fasta) >= param)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]                         #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                       #確認してるだけです

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

Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

パッケージ

(Rで)塩基配列解析の推奨インストール手順をおさらい。大まかな手順は、①R本体のインストール、および②CRANおよびBioconductorから提供されている各種パッケージのインストール。(2014年9月のNGS速習コース 3-3の一部)

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

(Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランスクリプ
(last modified 2015/02/15, since 2010)

What's new?

- このウェブページはフリーソフトRと必要者は、1. Rのインストールと起動および
- 門田幸二 著シリ ful R 第7巻
- 「作図 | クラスタリング」周辺の情報を
- 「作図 | ROC曲線」周辺で、発現変動
せています。(2015/02/08) NEW
- 「解析 | 発現変動 | 3群間 | 対応なし |
- 「解析 | シミュレーションカウントデータ
トデータを自在に作成するための項目

- はじめに (last modified 2014/01/30)
- 参考資料 (講義、講習会、本など) (last
- 過去のお知らせ (last modified 2015/02
- Rのインストールと起動 (last modified
- 基本的な利用法 (last modified 2015/0
- サンプルデータ (last modified 2015/02
- バイオインフォマティクス人材育成カリ
- 書籍 | トランスクリプトーム解析 | につい

Rのインストールと起動 NEW

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

1. Windows release版のインストールの場合:

①
1. Rのインストーラを「実行」
2. 聞かれるがままに「次へ」などを押してとにかくインストールを完了させる
3. **Windows Vistaの人**は(パッケージのインストール中に書き込み権限に関するエラーが出るのを避けるために)「コントロールパネル」-「ユーザーアカウント」-「ユーザーアカウント制御の有効化または無効化」で、「ユーザーアカウント制御(UAC)を使ってコンピュータの保護に役立たせる」の**チェックをあらかじめ外しておくことを強くお勧め**します。

②
4. インストールが無事完了したら、デスクトップに出現する「R3.*X.Y* (32 bitの場合; *X*や*Y*中の数値はバージョンによって異なります)」または「R x64 3.*X.Y* (64 bitの場合)」アイコンをダブルクリックして起動
5. 以下を、「Rコンソール画面上」でコピー&ペーストする。10GB程度のディスク容量を要しますが一番お手軽です。(どこからダウンロードするか？と聞かれるので、その場合は自分のいる場所から近いサイトを指定)

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

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

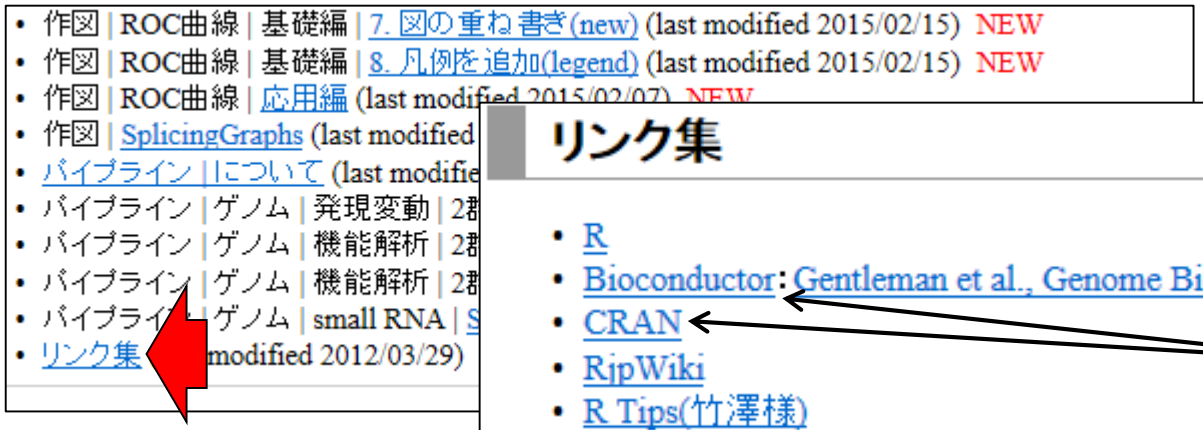
R本体とパッケージの関係

「R本体」と「パッケージ」の関係は、「パソコン」と「ソフト」、「Microsoft EXCEL」と「アドイン」、「Cytoscape」と「プラグイン」のようなものという理解でよい。

- パソコンを購入しただけの状態では、できることが限られています。
  - 通常は、Officeやウイルス撃退ソフトなどをインストールして利用します。
- Linuxをインストールしただけの状態では、できることが限られています。
  - 通常は、マッピングなど各種プログラムをインストールして利用します。
- R本体をインストールしただけの状態では、できることが限られています。
  - NGS解析を行う各種パッケージ（またはライブラリ）をインストールして利用します。

CRANとBioconductor

- R上で利用可能なパッケージの2大リポジトリ(貯蔵庫)
  - CRAN (The Comprehensive R Archive Network):6,328パッケージ
  - Bioconductor:934パッケージ



CRANは生命科学分野に限らず様々な分野で利用されるパッケージを含む。NGS解析は、主にBioconductorから提供されているパッケージを利用します。

パッケージ

- 「(Rで)塩基配列解析」のインストール手順おさらい

Rのインストールと起動 NEW

基本的にはこちらまたはこちらをご覧ください。
よく分からない人でWindowsユーザーの方は以下を参考にしてくださ
14日にアップデートしたMac版のインストール手順こちら(by 孫建強氏
ンストールしたほうがよい」です。

1. Windows release版のインストールの場合:

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

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

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

たった3行のコードで2つのリポジトリから提供されている多くのパッケージ群を一度にインストール。有線LAN接続環境でも数時間程度かかるのは、数千ものパッケージのダウンロードおよびインストールに相当する部分だから。

定期的なバージョンアップをお勧め

- R本体もBioconductorも定期的にバージョンアップがなされている
  - R (http://www.r-project.org/)
    - 2014-10-31にver. 3.1.2をリリース
    - 2014-07-10にver. 3.1.1をリリース
    - 2014-04-10にver. 3.1.0をリリース
    - …
    - 2012-03-30にver. 2.15.0をリリース
    - …
  - Bioconductor (http://bioconductor.org/)は半年ごとにリリース
    - 2014-10にver. 3.0をリリース (R ver. 3.1.1で動作確認)、提供パッケージ数:934
    - 2014-04にver. 2.14をリリース (R ver. 3.1.0で動作確認)、提供パッケージ数:824
    - 2013-10にver. 2.13をリリース (R ver. 3.0で動作確認)、提供パッケージ数:750
    - 2013-04にver. 2.12をリリース (R ver. 3.0で動作確認)、提供パッケージ数:672
    - 2012-10にver. 2.11をリリース (R ver. 2.15.1で動作確認)、提供パッケージ数:608
    - 2012-04にver. 2.10をリリース (R ver. 2.15.0で動作確認)、提供パッケージ数:553
    - 2011-11にver. 2.9をリリース (R ver. 2.14.0で動作確認)、提供パッケージ数:517

2015年2月18日現在のR本体の最新バージョンは**3.1.2**、Biostrings最新バージョンは**2.34.1**。しかし、R ver. **2.15.0**など本体のバージョンが古いと、2014年8月14日にBiostringsパッケージを個別にインストールしてもver. **2.26.3**と昔のバージョンがインストールされる点に注意が必要です。(2014年9月のNGS速習コース 3-4の一部)

(Rで)マイクロアレイデータ解析

(last modified 2014/07/24, since 2005)

- Rの昔のバージョンのインストール

What's new?

- 門田幸二著シリーズ Useful R 第7巻トランスクリプトーム解析刊行(共立出版
  や、ROKU法 (Kadota et al., 2006)、WAD法 (Kadota et al., 2008)などについて
  レイ解析部分のRコードについては、このページの「書籍 | トランスクリプトーム
- お知らせは主に(Rで)塩基配列解析で行っておりますのでそちらをご覧くださ
  ども(Rで)塩基配列解析中の参考資料(講義、講習会、本など)から辿れます。

- はじめに(last modified 2014/05/14)
- 過去のお知らせ(last modified 2014/03/03)
- Rのインストールと起動(last modifie 2014/05/14)
- Rの昔のバージョンのインストール(modified 2012/04/07)
- 使用例(初心者向け)(last modified 1/09/15)
- サンプルデータ (last modified 2014/05/02)

Rの昔のバージョンのインスト

DFW (Chen et al., Bioinformatics, 2007)というAff
りの"階層的クラスタリング結果を導くような遺伝
データ用の正規化法(嫌味ではなくいい方法なん
は、R2.7.2あたりだと正常に動作していましたが、
く動いてくれません。そのような場合でも、Rの昔
ることによって、DFWを利用することができます。
の昔の ージョンをインストールするやり方をR2.

1. ここをクリックして、任意のRのバージョンの
2. R-X.Y.Z-win32.exe(例えばR-2.7.2-win32.
   がままに押す

Previous Releases of R for Windows

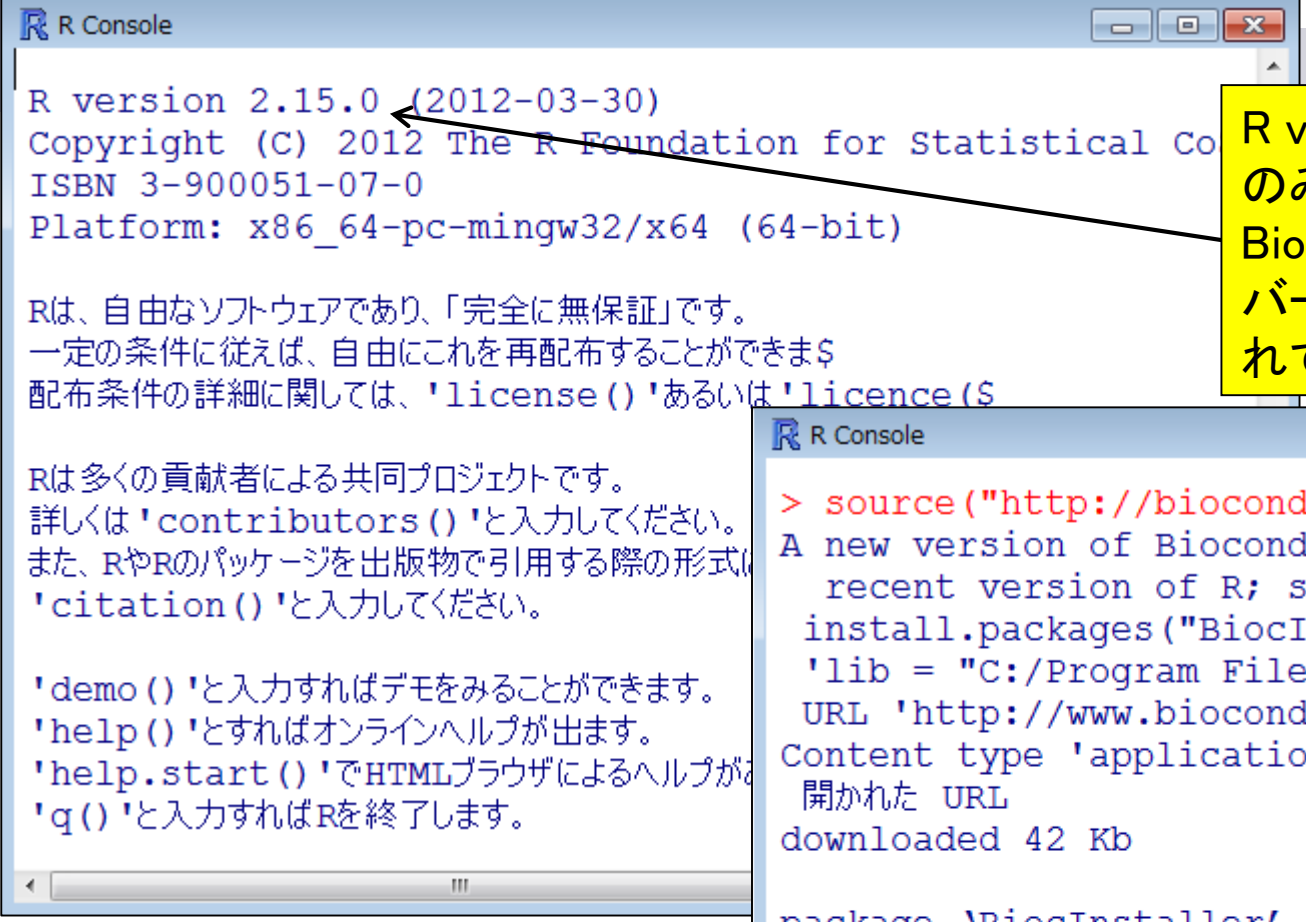
This directory contains previous binary releases of R to run on Windows 95, 98, ME, NT4.0, 2000 and XP or later on Intel/clone chips.

The current release, and links to development snapshots, are available here. Source code for these releases and others is available through the main CRAN page.

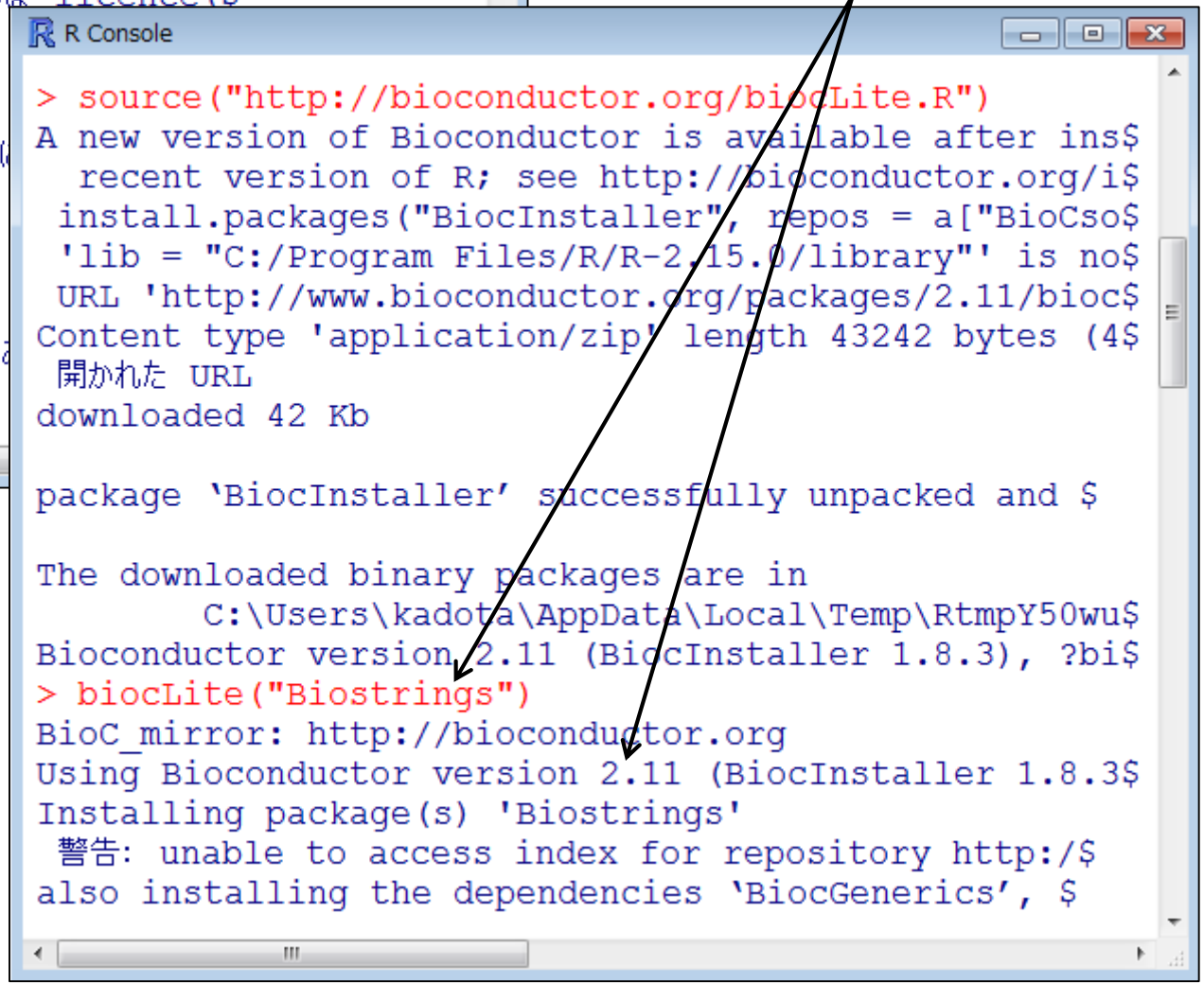
In this directory:

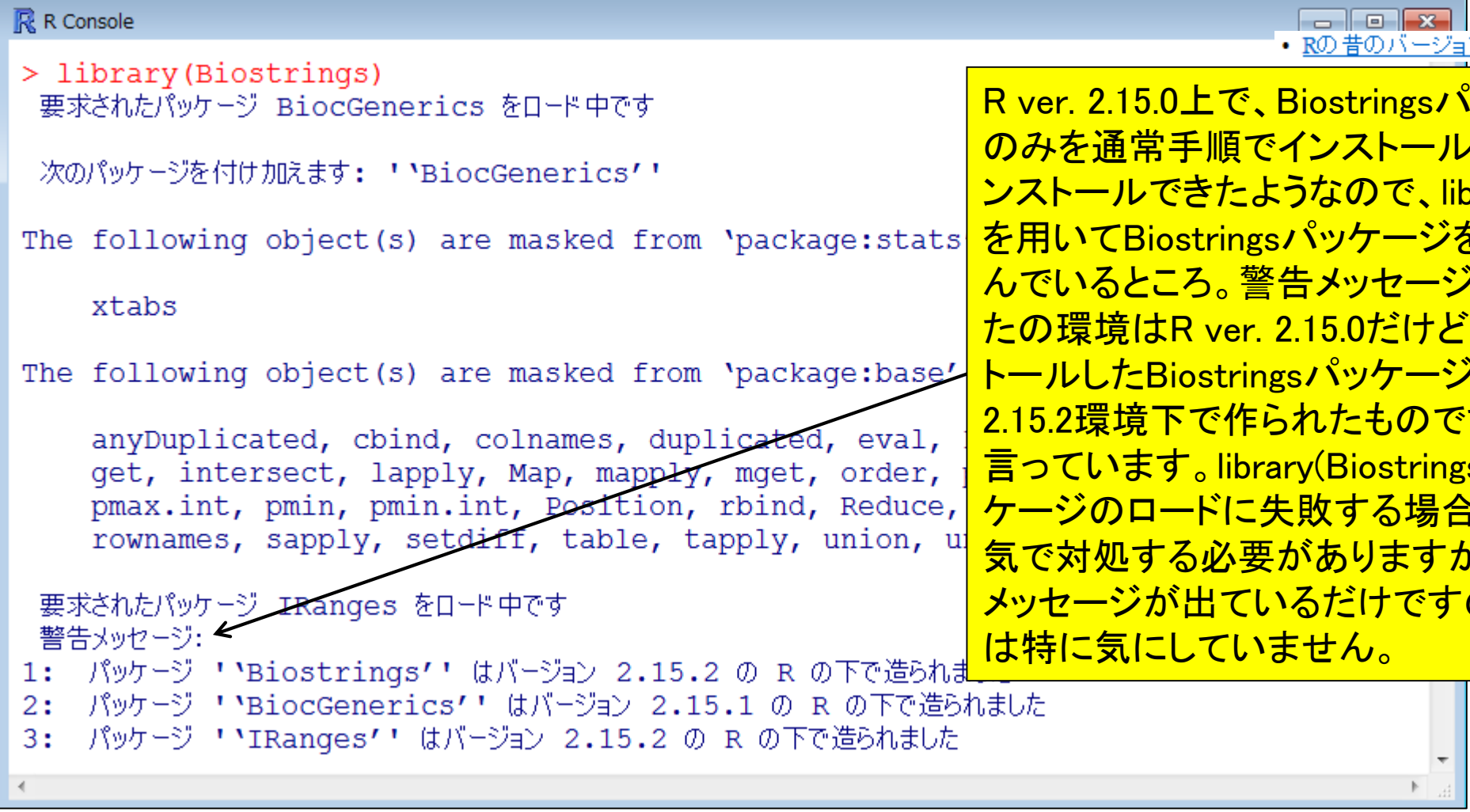
R 3.1.0 (April, 2014)
R 3.0.3 (March, 2014)
R 3.0.2 (September, 2013)
R 3.0.1 (May, 2013)
R 3.0.0 (April, 2013)
R 2.15.3 (March, 2013)
R 2.15.2 (October, 2012)
R 2.15.1 (e, 2012)
R 2.15.0 rch, 2012)
R 2.14.2 (bruary, 2012)
R 2.14.1 (December, 2011)
R 2.14.0 (November, 2011)
R 2.13.2 (September, 2011)
R 2.13.1 (July, 2011)
R 2.13.0 (April, 2011)
R 2.12.2 (February, 2011)
R 2.12.1 (December, 2010)
R 2.12.0 (October, 2010)
R 2.11.1 (May, 2010)
R 2.11.0 (April, 2010)
R 2.10.1 (December, 2009)
R 2.10.0 (October, 2009)

Rの昔のバージョンのインストール手順。Windows版のver. 2.15.0の場合。



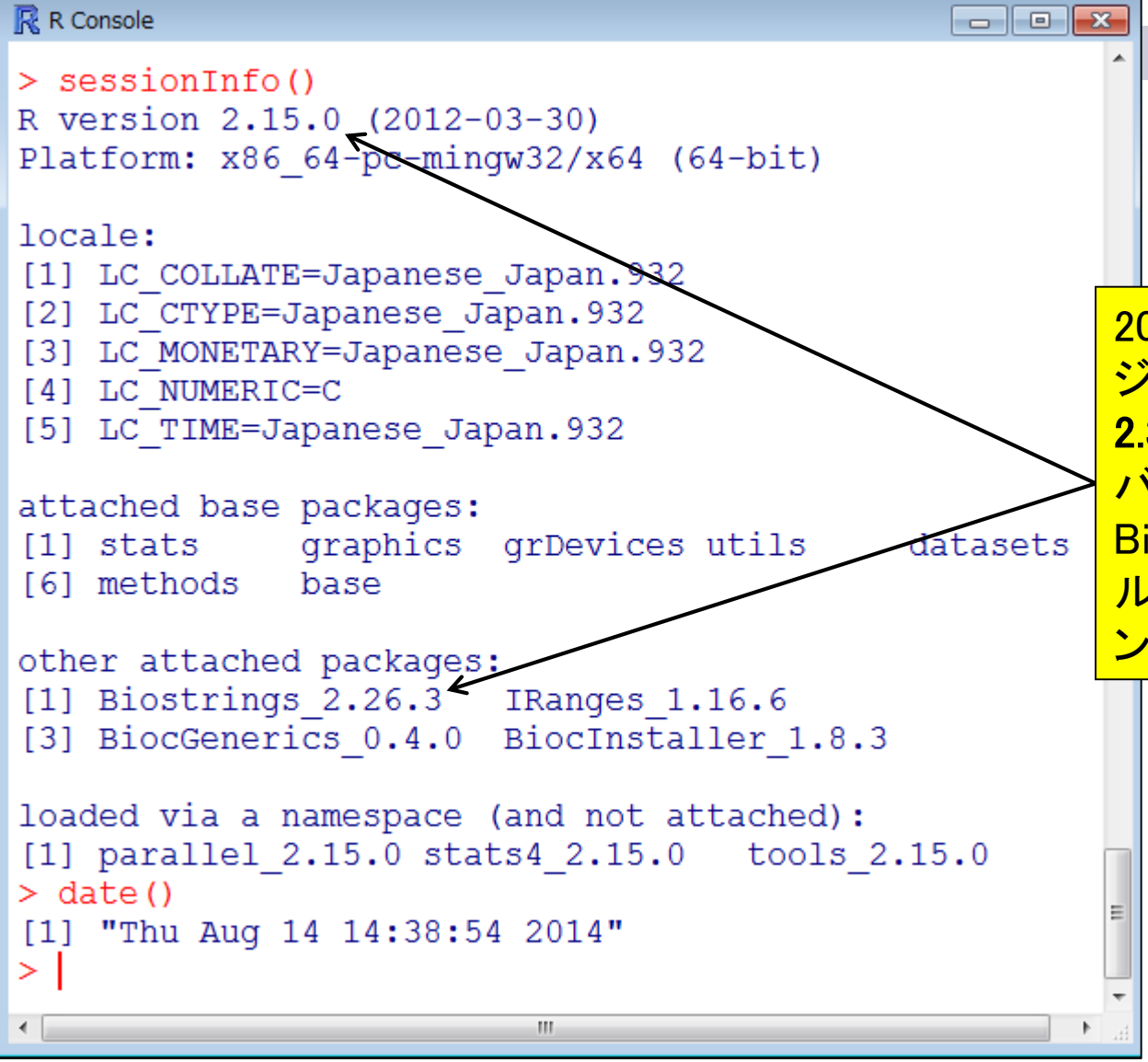
R ver. 2.15.0上で、Biostringsパッケージのみを通常手順でインストール。
Bioconductor ver. 2.11でリリースされたバージョンのBiostringsがインストールされていることが分かる。





- Rの昔のバージョンのインストール

R ver. 2.15.0上で、Biostringsパッケージのみを通常手順でインストール。無事インストールできたようなので、library関数を用いてBiostringsパッケージを読み込んでいるところ。警告メッセージは「あなたの環境はR ver. 2.15.0だけど、インストールしたBiostringsパッケージはR ver. 2.15.2環境下で作られたものです」と言っています。library(Biostrings)でパッケージのロードに失敗する場合には本気で対処する必要がありますが、警告メッセージが出ているだけですので、私は特に気にしていません。



2015年2月18日現在のR本体の最新バージョンは**3.1.2**、Biostrings最新バージョンは**2.34.1**。しかし、R ver. **2.15.0**など本体のバージョンが古いと、2014年8月14日にBiostringsパッケージを個別にインストールしてもver. **2.26.3**と昔のバージョンがインストールされる点に注意が必要です。

定期的なバージョンアップをお勧め

- R本体もBioconductorも定期的にバージョンアップがなされている
  - R (http://www.r-project.org/)
    - 2014-10-31にver. 3.1.2をリリース
    - 2014-07-10にver. 3.1.1をリリース
    - 2014-04-10にver. 3.1.0をリリース
    - …
    - 2012-03-30にver. 2.15.0をリリース
    - …
  - Bioconductor (http://bioconductor.org/)は半年ごとにリリース
    - 2014-10にver. 3.0をリリース (R ver. 3.1.1で動作確認)、提供パッケージ数:934
    - 2014-04にver. 2.14をリリース (R ver. 3.1.0で動作確認)、提供パッケージ数:824
    - 2013-10にver. 2.13をリリース (R ver. 3.0で動作確認)、提供パッケージ数:750
    - 2013-04にver. 2.12をリリース (R ver. 3.0で動作確認)、提供パッケージ数:672
    - 2012-10にver. 2.11をリリース (R ver. 2.15.1で動作確認)、提供パッケージ数:608
    - 2012-04にver. 2.10をリリース (R ver. 2.15.0で動作確認)、提供パッケージ数:553
    - 2011-11にver. 2.9をリリース (R ver. 2.14.0で動作確認)、提供パッケージ数:517

基本的にはバグの修正や新たな機能がどんどん追加されているので最新版の利用をお勧め。毎年5月初めと11月初めごろにBioconductorを覗きにいくとよいだろう。

Contents1

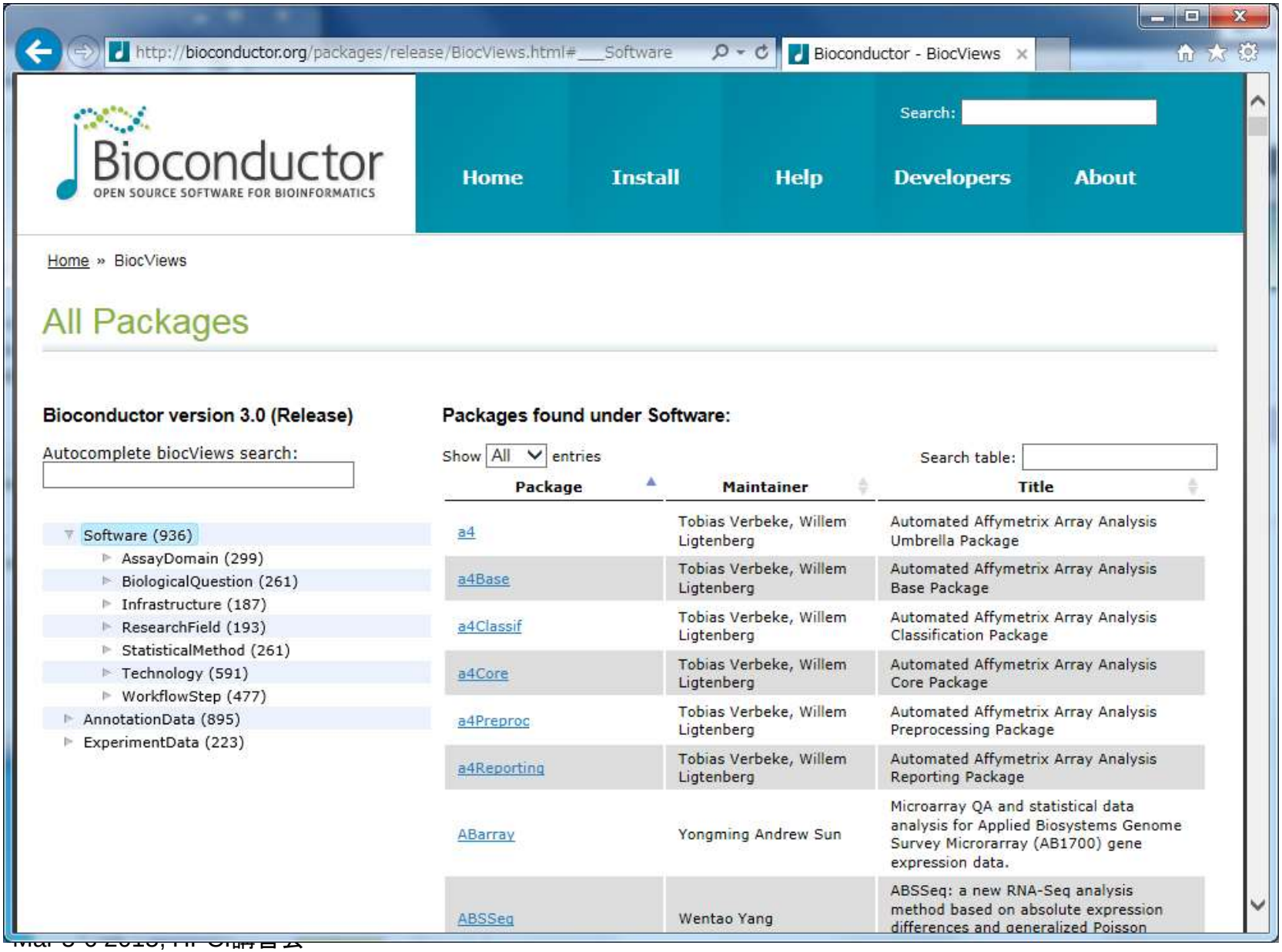
- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

Bioconductor概観

PubMed上で「R Bioconductor」でキーワード検索し原著論文があるパッケージのみ探すのも一つの戦略ですが、原著論文公開前のパッケージも見つかります。

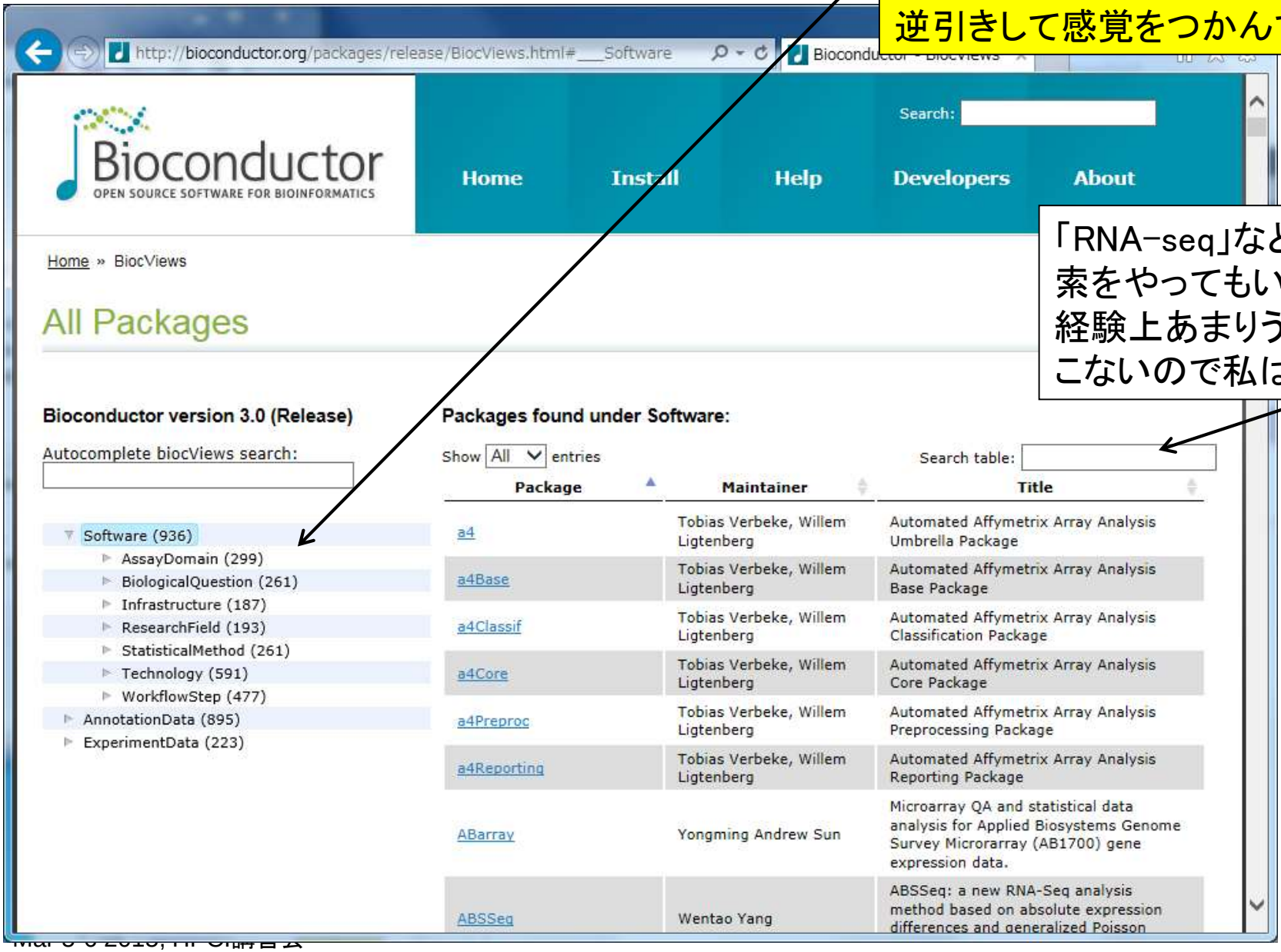
Bioconductor概観

ネットワーク環境にもよりますが、数秒～数分以内にこのような画面になります。



Bioconductor概観

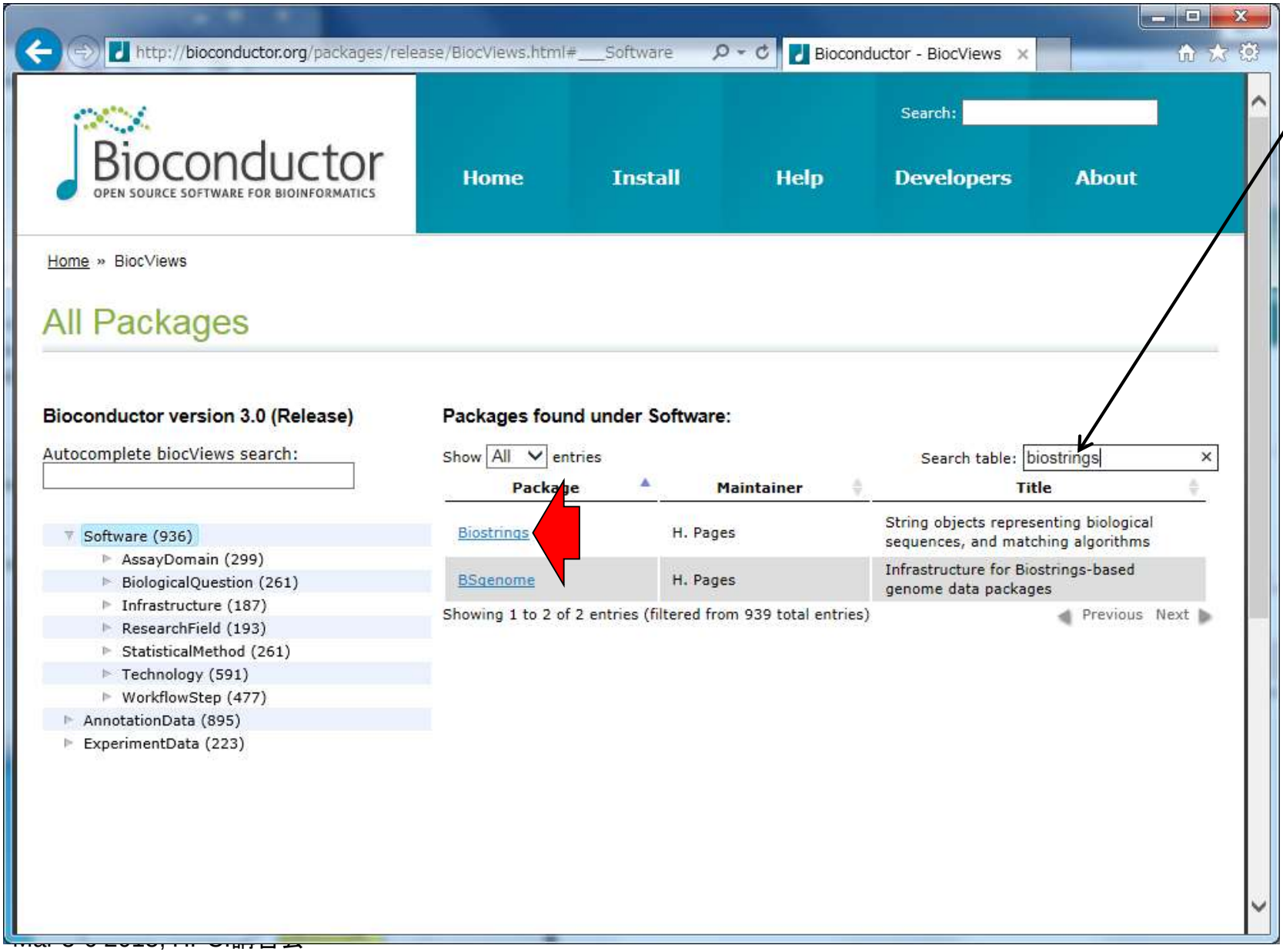
基本的には左側のカテゴリ分けのところを眺めますが、Biostringsなど何を行うパッケージかがある程度分かっているものから逆引きして感覚をつかんでおくとよいでしょう。



「RNA-seq」など、フリーワード検索をやってもいいとは思いますが、経験上あまりうまく引っかかってこないので私はやりません。

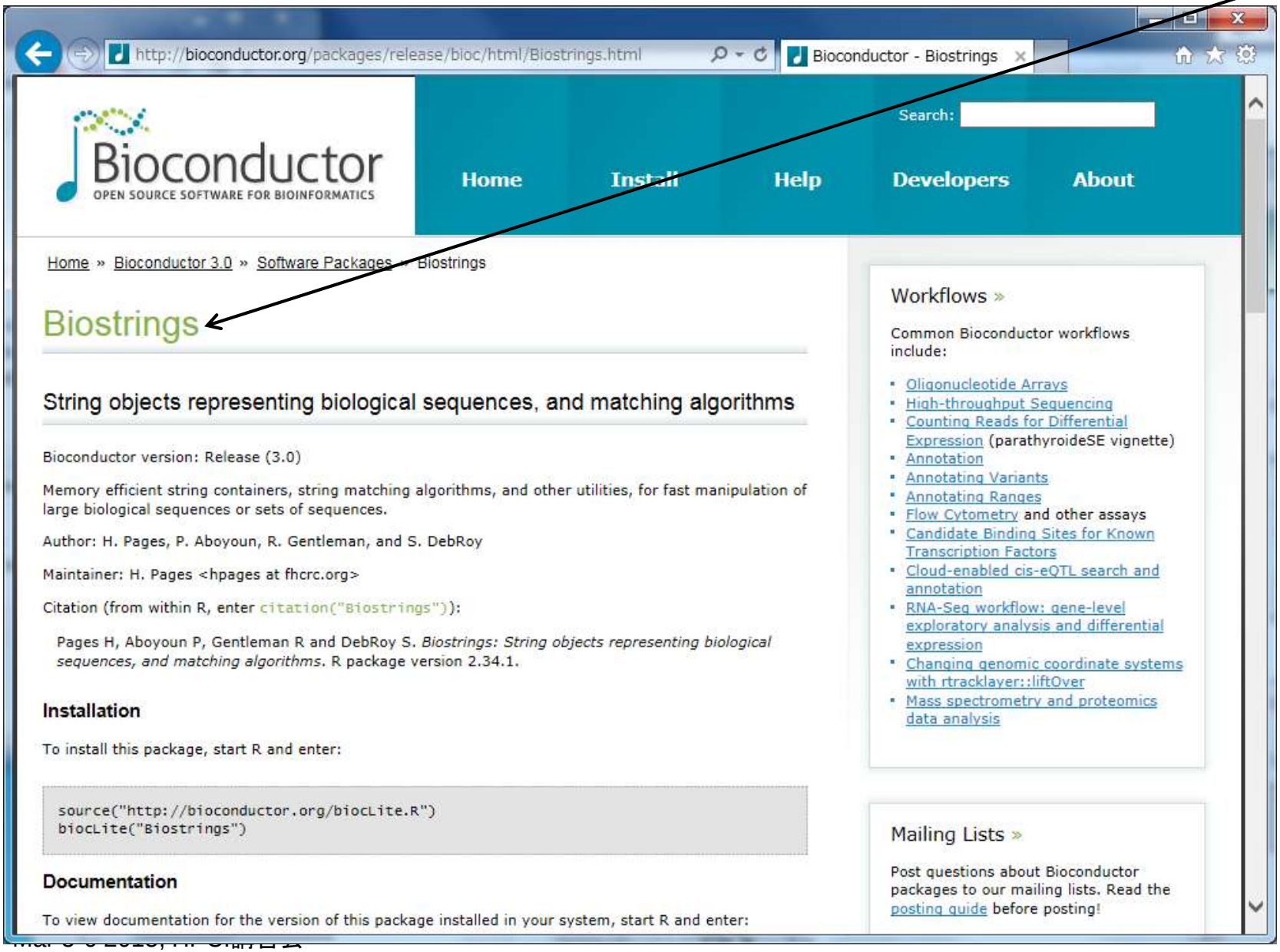
Bioconductor概観

「biostrings」と打つとすぐにリストアップされる。



Bioconductor概観

BioconductorのBiostringsパッケージのページに飛びます。

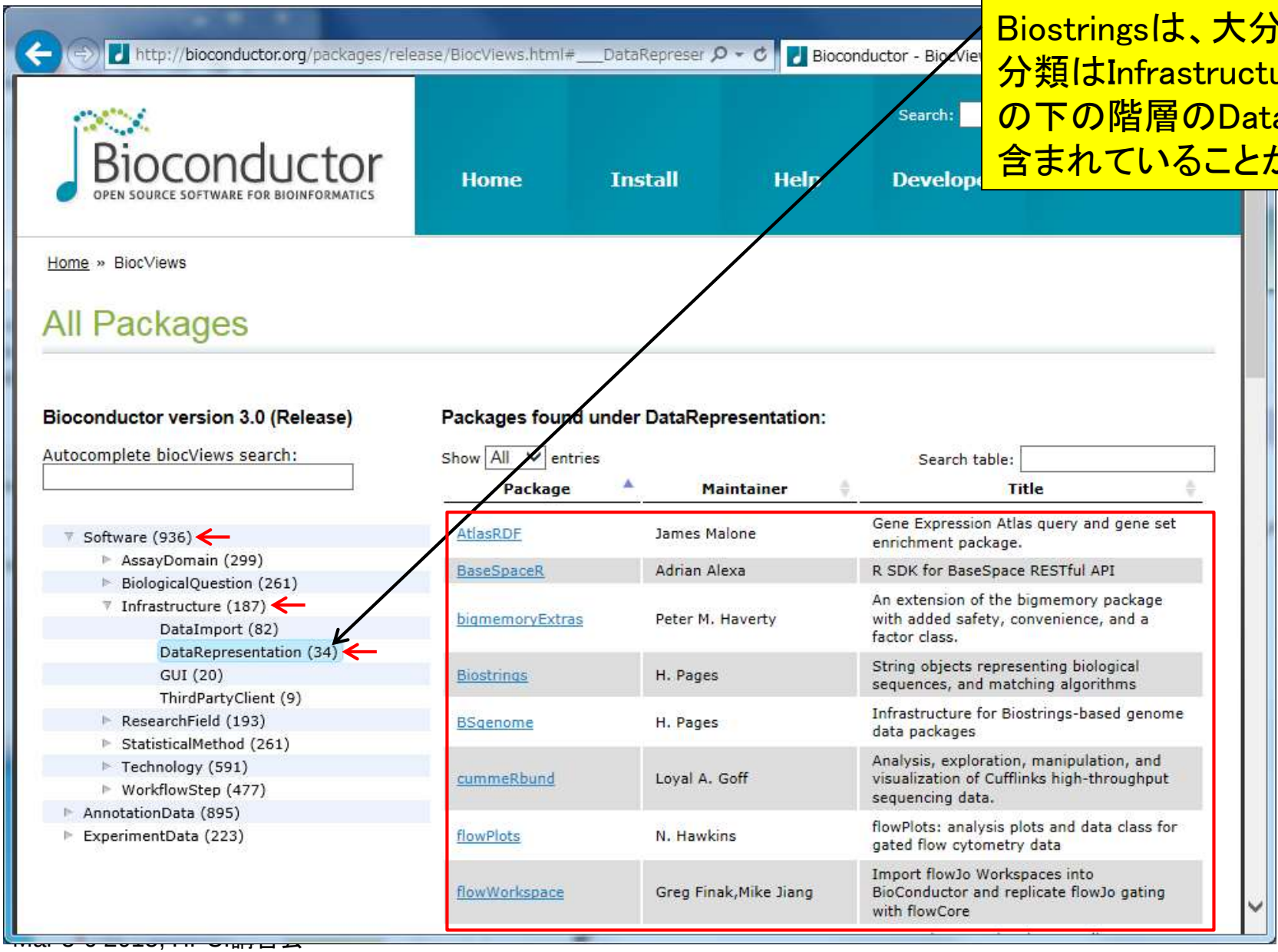


Bioconductor概観

BioconductorのBiostringsパッケージのページで、ちょっと下のほうに移動。biocViewsのところで見えるキーワードっぽいのがさきほどのカテゴリ分けに相当。例えば、DataRepresentationをクリックするとカテゴリ分けの階層関係がわかる。

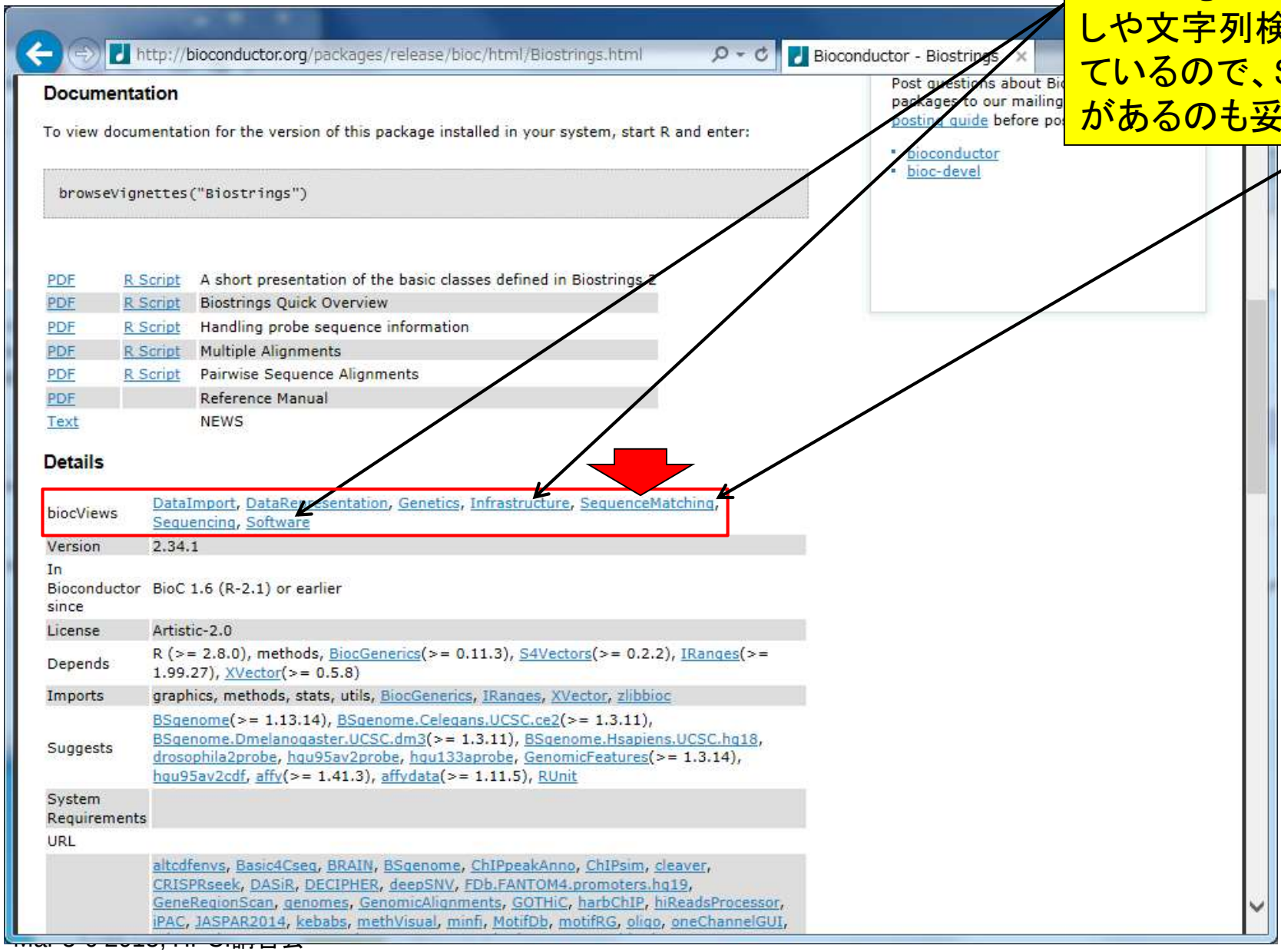
Bioconductor概観

DataRepresentationのカテゴリに含まれるのは34パッケージであることが分かる。赤枠部分がそのリスト。Biostringsは、大分類はSoftware、中分類はInfrastructureとなっており、その下の階層のDataRepresentationに含まれていることがわかる。



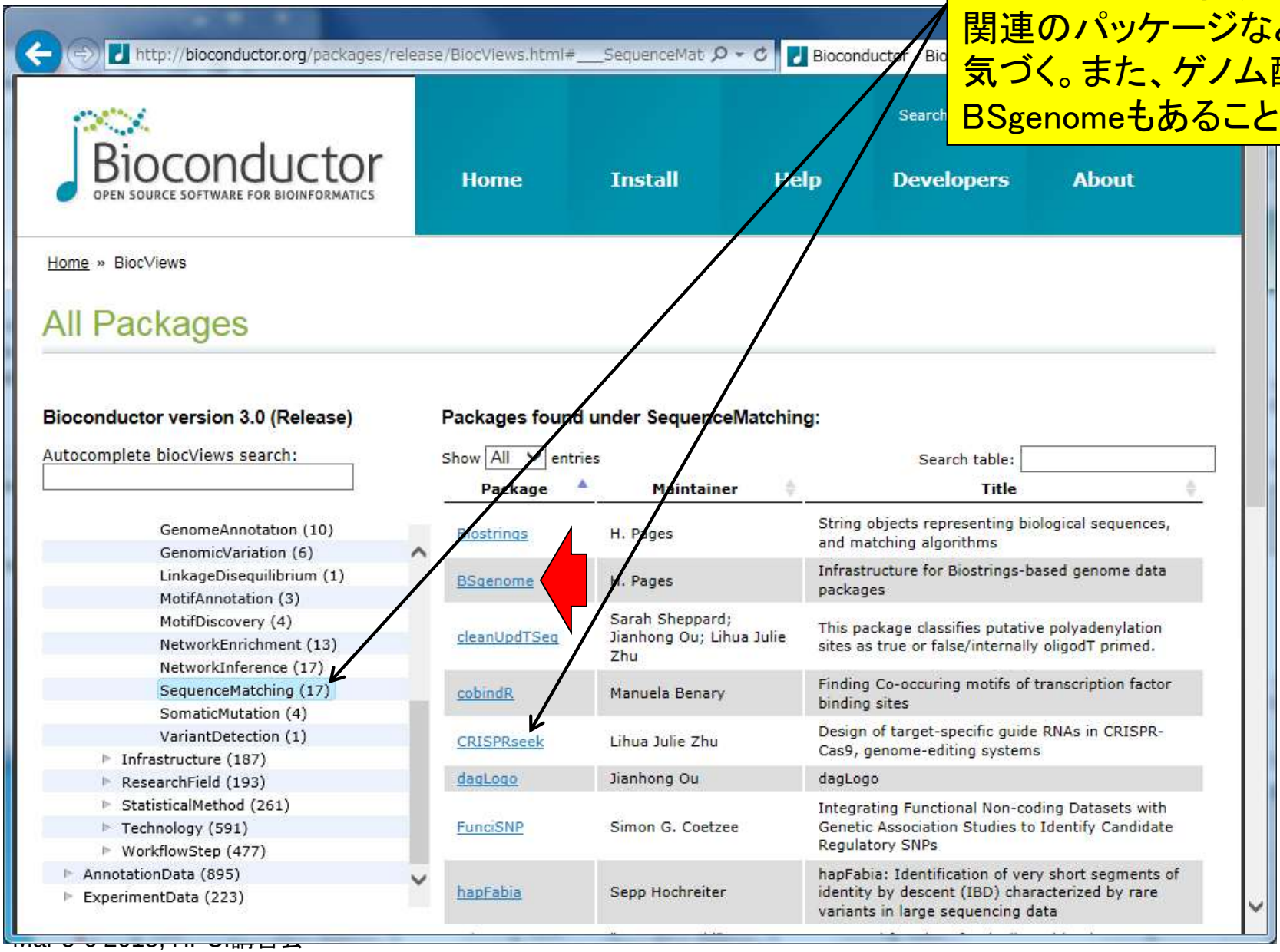
Bioconductor概観

大分類のSoftware、中分類のInfrastructureも存在する。Biostringsは塩基配列の切り出しや文字列検索系関数も提供しているので、SequenceMatchingがあるのも妥当。



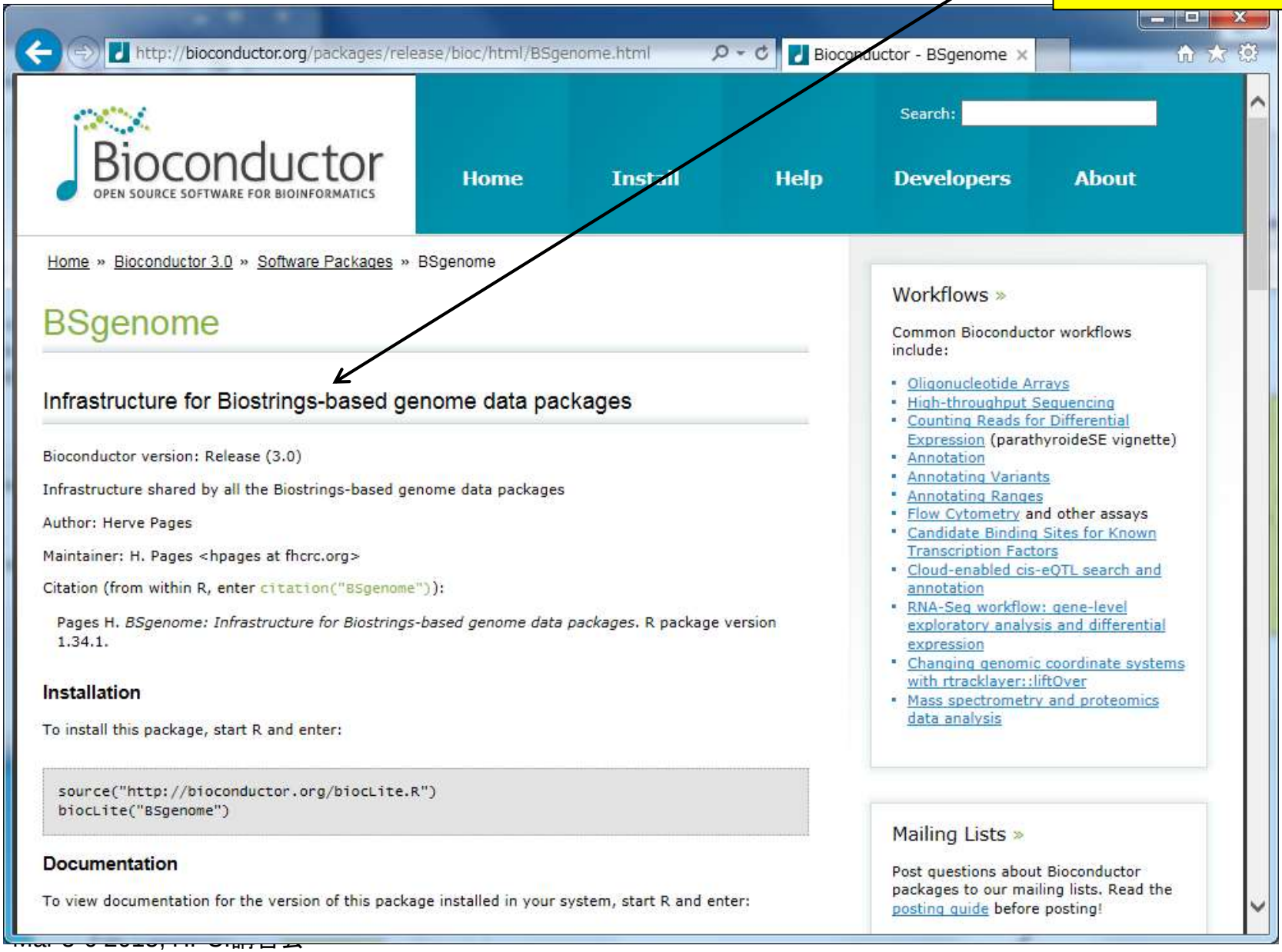
Bioconductor概観

SequenceMatchingに含まれる17パッケージの一部しか表示されていないが、(Rで)**塩基配列解析**中にはないCRISPR関連のパッケージなども存在することに気づく。また、ゲノム配列パッケージBSgenomeもあることがわかる。



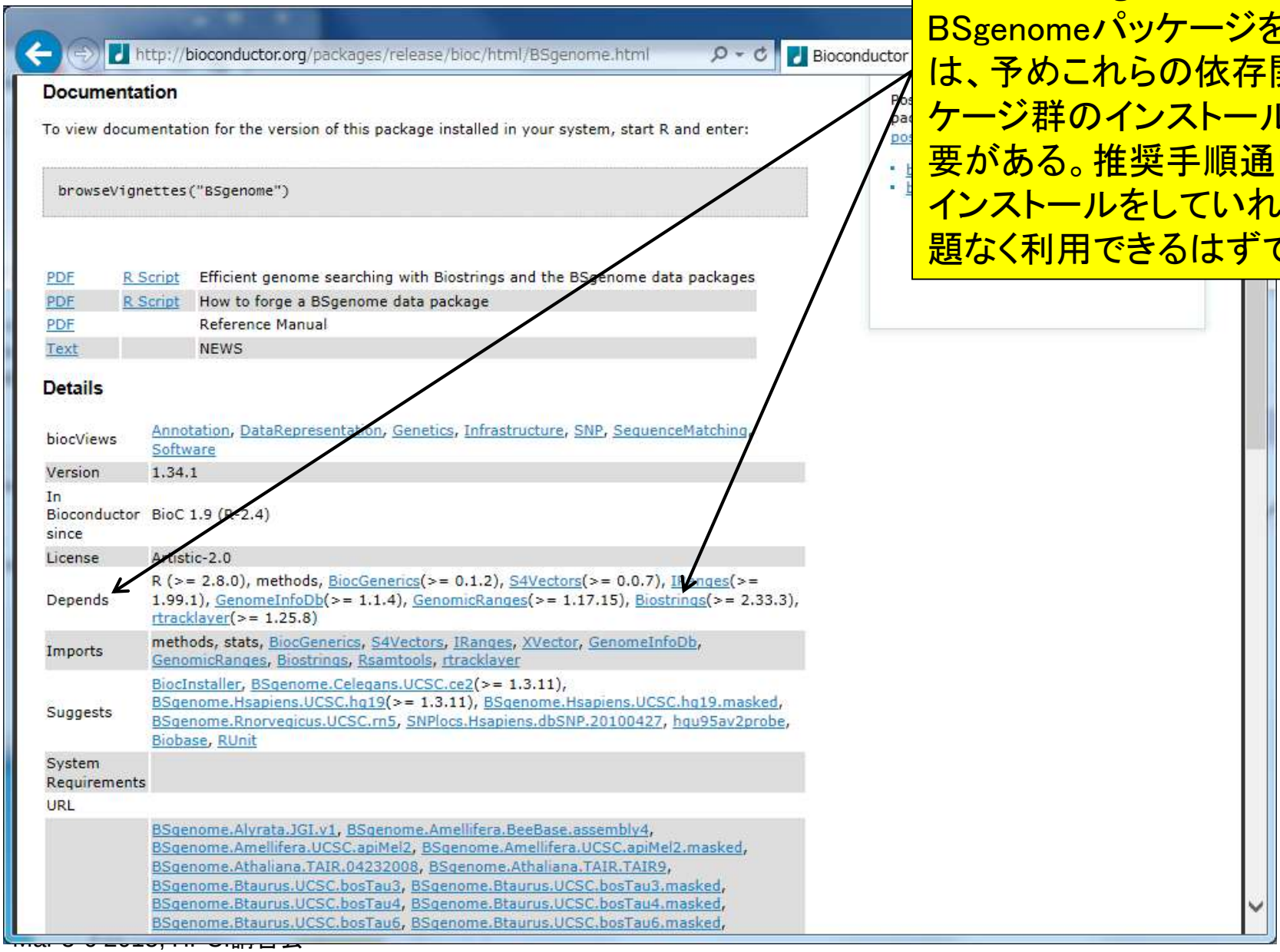
Bioconductor概観

ゲノム配列パッケージBSgenomeは、内部的にBiostringsパッケージを利用していることがわかる。



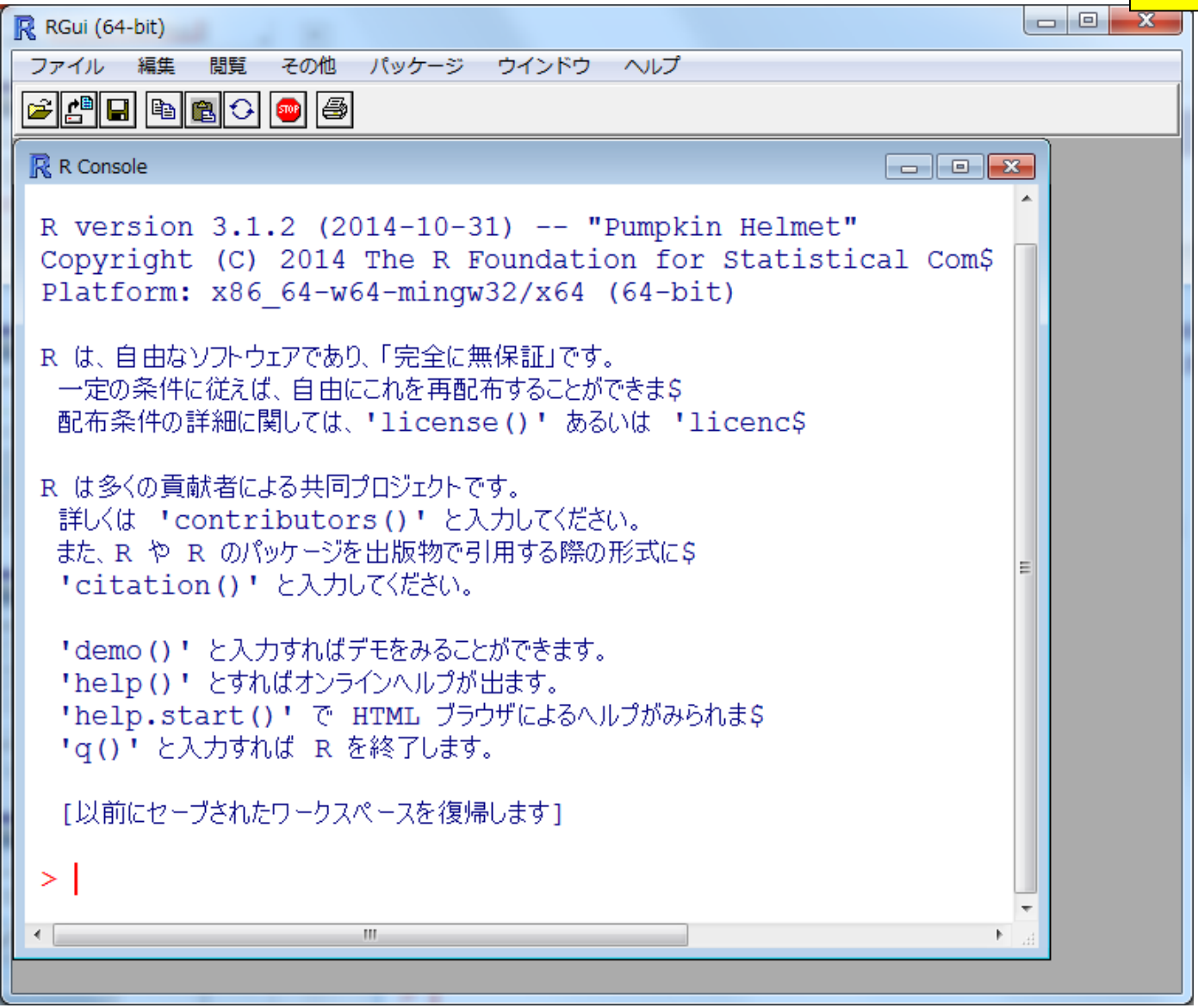
Bioconductor概観

ゲノム配列パッケージBSgenomeのちょっと下のほうに移動。Depends(依存)のところにBiostringsが存在することがわかる。BSgenomeパッケージを利用したい場合には、予めこれらの依存関係のあるパッケージ群のインストールが完了している必要がある。推奨手順通りにパッケージのインストールをしていればBSgenomeを問題なく利用できるはずである。



Bioconductor概観

BSgenomeを問題なく利用できるかは、library(BSgenome)が通るかどうかで判断。R Guiの新規画面を起動。



Bioconductor概観

library(BSgenome)の実行結果中にエラーメッセージが出ていなければOK

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

R Console

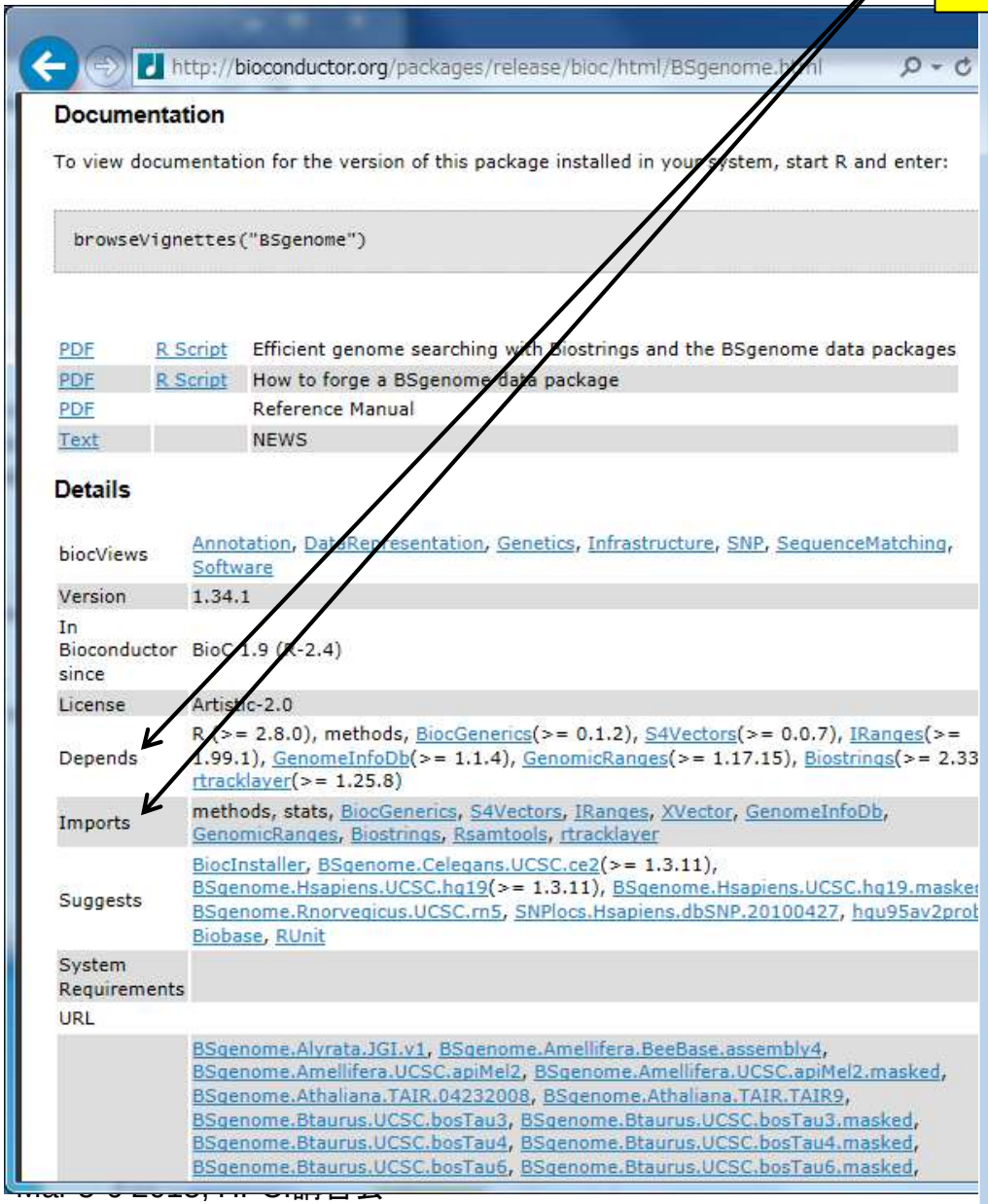
```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statisti
Platform: x86_64-w64-mingw32/x64 (64-bit)

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$
 配布条件の詳細に関しては、'license()' あるいは 'licenc$

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$
 'q()' と入力すれば R を終了します。

 [以前にセーブされたワークスペースを復帰します]

> library(BSgenome)
```

R Console

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

 次のパッケージを付け加えます: 'BiocGenerics'

The following objects are masked from 'package:paralle$

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

The following object is masked from 'package:stats':

    xtabs

The following objects are masked from 'package:base':

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

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

Bioconductor概観

library(XXX)をやったときに、XXXパッケージ内部で利用するDependsやImportsにリストアップされているパッケージも同時にロードしている(読み込んでいる)。



http://bioconductor.org/packages/release/bioc/html/BSgenome.html

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("BSgenome")
```

PDF	R Script	Efficient genome searching with Biostrings and the BSgenome data packages
PDF	R Script	How to forge a BSgenome data package
PDF		Reference Manual
Text		NEWS

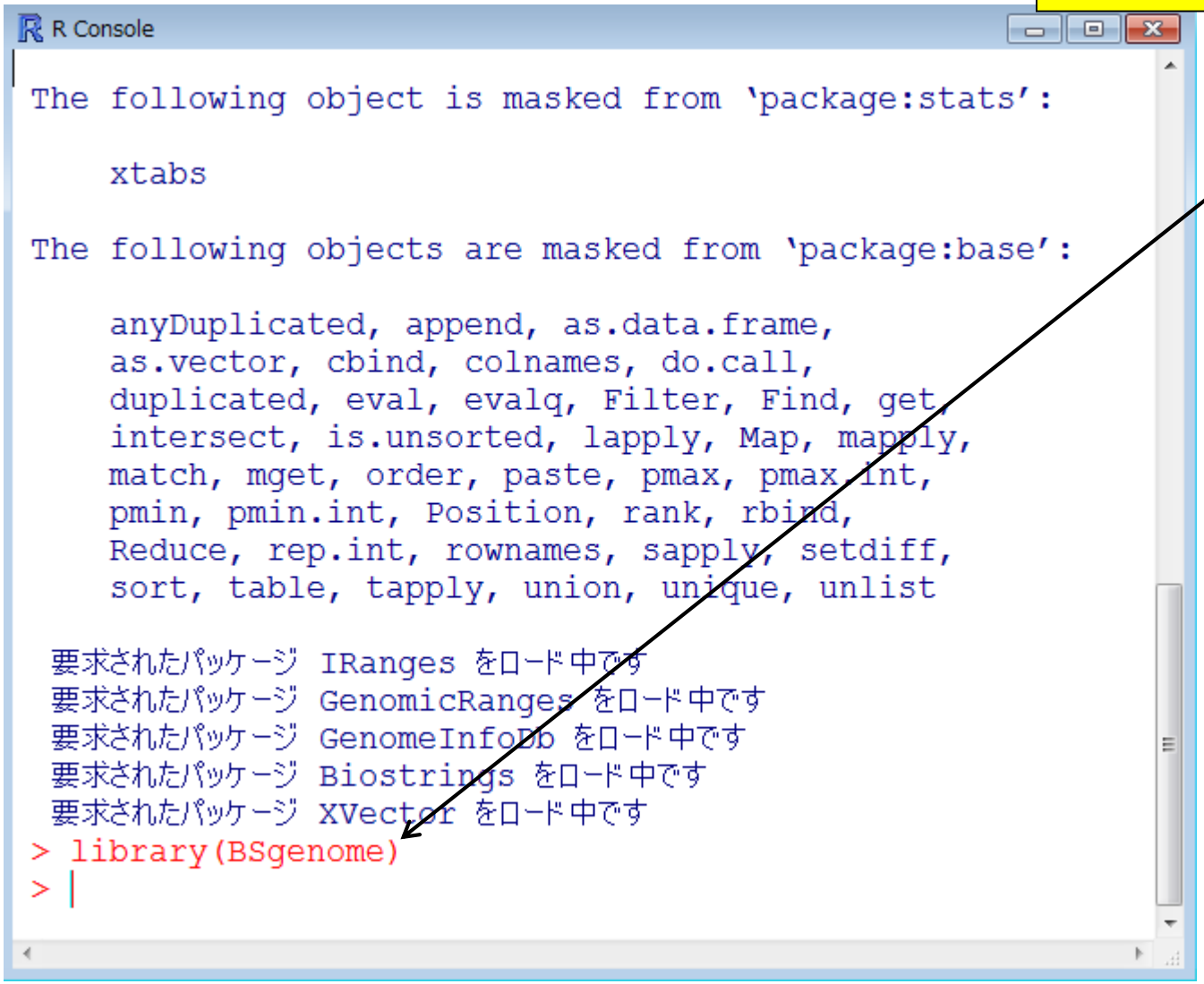
Details

biocViews	Annotation, DataRepresentation, Genetics, Infrastructure, SNP, SequenceMatching, Software
Version	1.34.1
In Bioconductor since	BioC 1.9 (R-2.4)
License	Artistic-2.0
Depends	R (>= 2.8.0), methods, BiocGenerics(>= 0.1.2), S4Vectors(>= 0.0.7), IRanges(>= 1.99.1), GenomeInfoDb(>= 1.1.4), GenomicRanges(>= 1.17.15), Biostrings(>= 2.33 rtracklayer(>= 1.25.8)
Imports	methods, stats, BiocGenerics, S4Vectors, IRanges, XVector, GenomeInfoDb, GenomicRanges, Biostrings, Rsamtools, rtracklayer
Suggests	BiocInstaller, BSgenome.Celegans.UCSC.ce2(>= 1.3.11), BSgenome.Hsapiens.UCSC.hg19(>= 1.3.11), BSgenome.Hsapiens.UCSC.hg19.maske BSgenome.Rnorvegicus.UCSC.rn5, SNPlocs.Hsapiens.dbSNP.20100427, hgu95av2prot Biobase, RUnit
System Requirements	
URL	
	BSgenome.Alyrata.JGI.v1, BSgenome.Amellifera.BeeBase.assembly4, BSgenome.Amellifera.UCSC.apiMel2, BSgenome.Amellifera.UCSC.apiMel2.masked, BSgenome.Athaliana.TAIR.04232008, BSgenome.Athaliana.TAIR.TAIR9, BSgenome.Btaurus.UCSC.bosTau3, BSgenome.Btaurus.UCSC.bosTau3.masked, BSgenome.Btaurus.UCSC.bosTau4, BSgenome.Btaurus.UCSC.bosTau4.masked, BSgenome.Btaurus.UCSC.bosTau6, BSgenome.Btaurus.UCSC.bosTau6.masked,

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

 次のパッケージを付け加えます: 'BiocGenerics'

The following objects are masked from 'package:paralle$

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

The following object is masked from 'package:stats':

    xtabs

The following objects are masked from 'package:base':

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

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

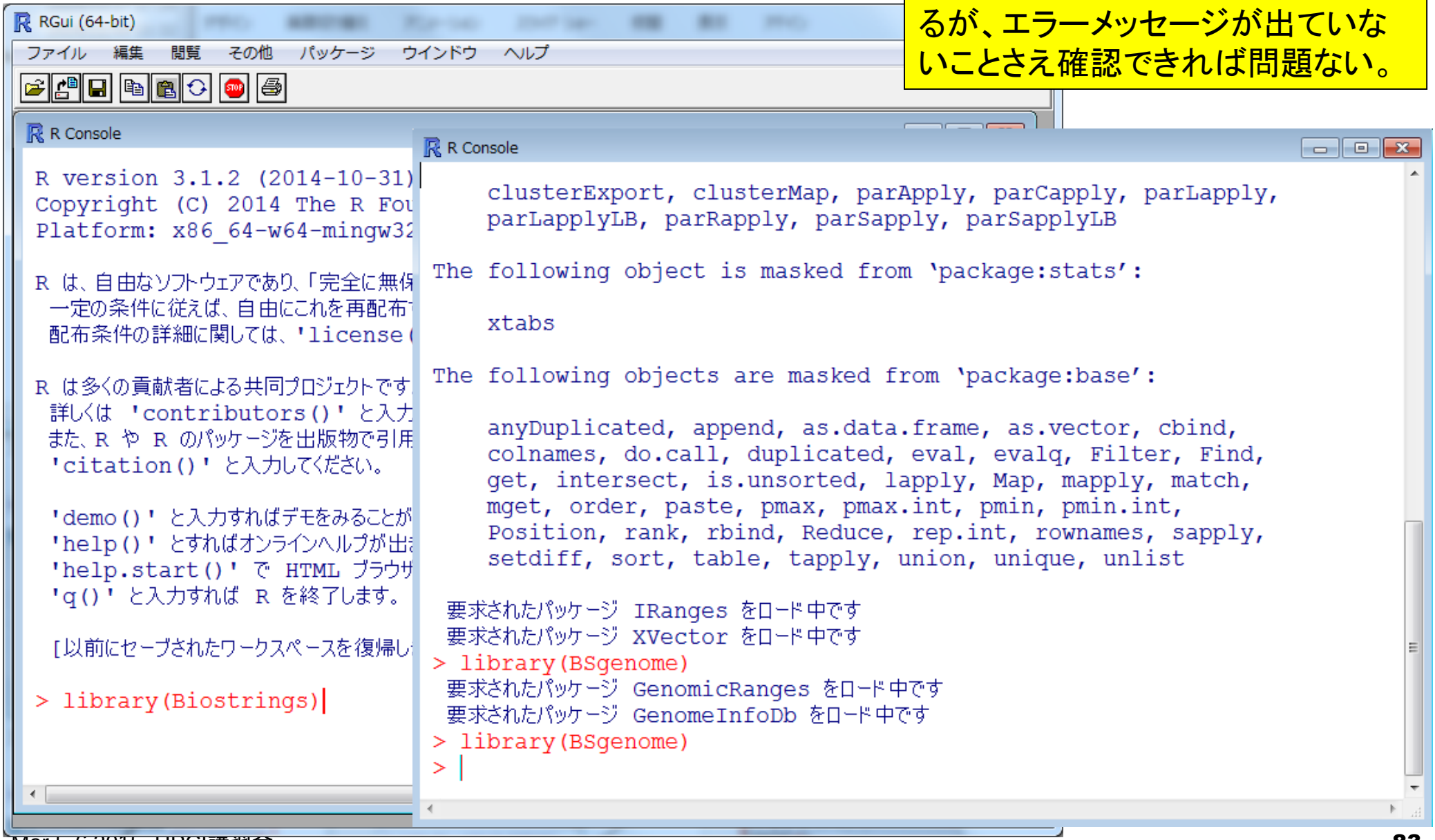
Bioconductor概観

もう一度library(BSgenome)をやってもメッセージは出ません。エラーメッセージが出ていなければ特に気にする必要はありません。



Bioconductor概観

先にBSgenomeが内部的に利用しているBiostringsパッケージのロードを行っておくと、若干見栄えが異なるが、エラーメッセージが出ていないことさえ確認できれば問題ない。



BSgenome利用の意義

ゲノム配列はUCSCやEnsemblなどのウェブサイトから取得するのが一般的ではあるが、Rの生物種ごとに提供されているBSgenomeで取得、あるいは取り扱うことも可能。ChIP-seq用パッケージMEDIPSは内部的にBSgenomeを利用。

- 解析 | 機能解析 | パスウェイ(Pathway)解析 | SeqGSEA(Wang 2014) (last modified 2014/
- 解析 | 菌叢解析 | について (last modified 2014/12/19)
- 解析 | 菌叢解析 | phyloseq(McMurdie 2013) (last modified 2014/05/29)
- 解析 | エクソーム解析 | について (last modified 2
- 解析 | ChIP-seq | について modified 2015/02
- 解析 | ChIP-seq | DiffBind(Ross-Innes 2012) (las
- 解析 | ChIP-seq | ChIPseqR(Humburg 2011) (last
- 解析 | ChIP-seq | chipseq (last modified 2011/12/1
- 解析 | ChIP-seq | PICS(Zhang 2011) (last modifie

解析 | ChIP-seq | について

このあたりはほとんどノータッチです。SraTailor (Oki et al., 2014)は、実験医学2014年12月号の「Close Up実験法」中で日本語による解説記事があります(沖 真弥氏提供情報)。
2015年2月に調査した結果をリストアップします。

R用:

- ChIPsim: Zhang et al., PLoS Comput. Biol., 2008
- PeakSeq法: Rozowsky et al., Nat Biotechnol., 2009
- CSAR: Kaufmann et al., PLoS Biol., 2009
- rMAT: Droit et al., Bioinformatics, 2010
- ChIPpeakAnno: Zhu et al., BMC Bioinformatics, 2010
- PICS: Zhang et al., Biometrics, 2011
- ChIPseqR: Humburg et al., BMC Bioinformatics, 2011
- DiffBind: Ross-Innes et al., Nature, 2012
- MEDIPS: Lienhard et al., Bioinformatics, 2014
- DSS: Feng et al., Nucleic Acids Res., 2014
- methylSig: Park et al., Bioinformatics, 2014

R以外:

- bwtool: Pohl and Beato, Bioinformatics, 2014
- SraTailor: Oki et al., Genes Cells., 2014

Review、ガイドライン、パイプライン系:

- ガイドライン: Bailey et al., PLoS Comput Biol., 2013
- Review: Robinson et al., Front Genet., 2014

BSgenome

プロモーター配列取得ではなく、
ゲノム配列取得のほうです！

- イントロ | 一般 | 任意の長さの連続塩基の出現頻度情報を取得 (last modified 2013/06/14)
- イントロ | 一般 | Tips | 任意の拡張子でファイルを保存 (last modified 2013/09/26)
- イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して保存 (last modified 2013/09/26)
- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから (last modified 2014/05/28)
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome (last modified 2015/02/18) NEW
- イントロ | 一般 | 配列取得 | プロモーター配列 | 公共DBから (last modified 2014/04/02)
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome (last modified 2014/04/25)
- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence 2013) (last modified 2014/04/23)
- イントロ | 一般 | 配列取得 | トランスクリ
- イントロ | 一般 | 配列取得 | トランスクリ
- イントロ | NGS | 様々なプラットフォーム
- イントロ | NGS | qPCRやmicroarrayなど

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。getSeq関数はBSgenomeオブジェクト 中の「single sequences」というあたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」以外に「chr1_random」や「chrUn_random」なども等価に取扱っている点に注意してください。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

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

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

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

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

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

BSgenome

黒枠部分のコードをコピペ。R ver. 3.1.2 (Bioconductor ver. 3.0)で利用可能な生物種のパッケージ名をリストアップ。71個あることが分かる。Rのバージョンが古いとパッケージ数は少なくなる。

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタ
lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬
(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ
(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チ
ンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々
な生物種が利用可能であることがわかります。getSeq関数はBSgenomeオブジェクト 中の「single sequences」という
あたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」
以外に「chr1_random」や「chrUn_random」なども等価に取扱っている
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに

1. 利用可能な生物種とRにインストール済みの生物種をリストアップ

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

#本番（利用可能なパッケージをリストアップ；インストール
available.genomes()                                  #このパッケー

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

#後処理（パッケージ名でだいたいわかるがproviderやversi
installed.genomes(splitNameParts=TRUE) #インストー
```

R Console

```
[56] "BSgenome.Ptroglodytes.UCSC.panTro2"
[57] "BSgenome.Ptroglodytes.UCSC.panTro2.masked"
[58] "BSgenome.Ptroglodytes.UCSC.panTro3"
[59] "BSgenome.Ptroglodytes.UCSC.panTro3.masked"
[60] "BSgenome.Rnorvegicus.UCSC.rn4"
[61] "BSgenome.Rnorvegicus.UCSC.rn4.masked"
[62] "BSgenome.Rnorvegicus.UCSC.rn5"
[63] "BSgenome.Rnorvegicus.UCSC.rn5.masked"
[64] "BSgenome.Scerevisiae.UCSC.sacCer1"
[65] "BSgenome.Scerevisiae.UCSC.sacCer2"
[66] "BSgenome.Scerevisiae.UCSC.sacCer3"
[67] "BSgenome.Sscrofa.UCSC.susScr3"
[68] "BSgenome.Sscrofa.UCSC.susScr3.masked"
[69] "BSgenome.Tgondii.ToxoDB.7.0"
[70] "BSgenome.Tguttata.UCSC.taeGut1"
[71] "BSgenome.Tguttata.UCSC.taeGut1.masked"
> |
```

2013年12月にリリースされたヒトゲノム最新版(GRCh38)のRパッケージも利用可能です。

R Console

```
> available.genomes()
 [1] "BSgenome.Alyrata.JGI.v1"
 [2] "BSgenome.Amellifera.BeeBase.assem
 [3] "BSgenome.Amellifera.UCSC.apiMel2"
 [4] "BSgenome.Amellifera.UCSC.apiMel2.
 [5] "BSgenome.Athaliana.TAIR.04232008"
 [6] "BSgenome.Athaliana.TAIR.TAIR9"
 [7] "BSgenome.Btaurus.UCSC.bosTau3"
 [8] "BSgenome.Btaurus.UCSC.bosTau3.mas
 [9] "BSgenome.Btaurus.UCSC.bosTau4"
[10] "BSgenome.Btaurus.UCSC.bosTau4.mas
[11] "BSgenome.Btaurus.UCSC.bosTau6"
[12] "BSgenome.Btaurus.UCSC.bosTau6.mas
[13] "BSgenome.Celegans.UCSC.ce10"
[14] "BSgenome.Celegans.UCSC.ce2"
[15] "BSgenome.Celegans.UCSC.ce6"
[16] "BSgenome.Cfamiliaris.UCSC.canFam2
[17] "BSgenome.Cfamiliaris.UCSC.canFam2
[18] "BSgenome.Cfamiliaris.UCSC.canFam3
[19] "BSgenome.Cfamiliaris.UCSC.canFam3
[20] "BSgenome.Dmelanogaster.UCSC.dm2"
[21] "BSgenome.Dmelanogaster.UCSC.dm2.m
[22] "BSgenome.Dmelanogaster.UCSC.dm3"
[23] "BSgenome.Dmelanogaster.UCSC.dm3.m
[24] "BSgenome.Drerio.UCSC.danRer5"
[25] "BSgenome.Drerio.UCSC.danRer5.mask
[26] "BSgenome.Drerio.UCSC.danRer6"
[27] "BSgenome.Drerio.UCSC.danRer6.mask
[28] "BSgenome.Drerio.UCSC.danRer7"
```

R Console

```
[29] "BSgenome.Drerio.UCSC.danRer7.masked"
[30] "BSgenome.Ecoli.NCBI.20080805"
[31] "BSgenome.Gaculeatus.UCSC.gasAcu1"
[32] "BSgenome.Gaculeatus.UCSC.gasAcu1.masked"
[33] "BSgenome.Ggallus.UCSC.galGal3"
[34] "BSgenome.Ggallus.UCSC.galGal3.masked"
[35] "BSgenome.Ggallus.UCSC.galGal4"
[36] "BSgenome.Ggallus.UCSC.galGal4.masked"
[37] "BSgenome.Hsapiens.NCBI.GRCh38"
[38] "BSgenome.Hsapiens.UCSC.hg17"
[39] "BSgenome.Hsapiens.UCSC.hg17.masked"
[40] "BSgenome.Hsapiens.UCSC.hg18"
[41] "BSgenome.Hsapiens.UCSC.hg18.masked"
[42] "BSgenome.Hsapiens.UCSC.hg19"
[43] "BSgenome.Hsapiens.UCSC.hg19.masked"
[44] "BSgenome.Mfuro.UCSC.musFur1"
[45] "BSgenome.Mmulatta.UCSC.rheMac2"
[46] "BSgenome.Mmulatta.UCSC.rheMac2.masked"
[47] "BSgenome.Mmulatta.UCSC.rheMac3"
[48] "BSgenome.Mmulatta.UCSC.rheMac3.masked"
[49] "BSgenome.Mmusculus.UCSC.mm10"
[50] "BSgenome.Mmusculus.UCSC.mm10.masked"
[51] "BSgenome.Mmusculus.UCSC.mm8"
[52] "BSgenome.Mmusculus.UCSC.mm8.masked"
[53] "BSgenome.Mmusculus.UCSC.mm9"
[54] "BSgenome.Mmusculus.UCSC.mm9.masked"
[55] "BSgenome.Osativa.MSU.MSU7"
```

BSgenome

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハ
lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.eleg
(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.co
(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.
ンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gond
な生物種が利用可能であることがわかります。getSeq関数はBSgenomeオブジェクト 中の「single se
あたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲ
以外に「chr1_random」や「chrUn_random」なども等価に取扱っている点に注意してください。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. 利用可能な生物種とRにインストール済みの生物種をリストアップし

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

#本番（利用可能なパッケージをリストアップ；インストール済
available.genomes()                        #このパッケー

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

#後処理（パッケージ名でだいたいわかるがproviderやversion
installed.genomes(splitNameParts=TRUE) #インストール済
```

黒枠部分のコードをコピペ。数分程度かかります。実際にインストール済みのものは(このPC環境では)8パッケージであることがわかる。植物のシロイヌナズナ(*Arabidopsis thaliana*)のパッケージは推奨手順通りにインストール作業をしたヒトは存在するはずです。が、私もインストールされてなかったり…しますので、なければ個別インストールで対応してください。

R Console

```
 [67] "BSgenome.Sscrofa.UCSC.susScr3"
 [68] "BSgenome.Sscrofa.UCSC.susScr3.masked"
 [69] "BSgenome.Tgondii.ToxoDB.7.0"
 [70] "BSgenome.Tguttata.UCSC.taeGut1"
 [71] "BSgenome.Tguttata.UCSC.taeGut1.masked"
> installed.genomes()                      #インス$
 [1] "BSgenome.Athaliana.TAIR.TAIR9"
 [2] "BSgenome.Celegans.UCSC.ce2"
 [3] "BSgenome.Celegans.UCSC.ce6"
 [4] "BSgenome.Drerio.UCSC.danRer7"
 [5] "BSgenome.Ecoli.NCBI.20080805"
 [6] "BSgenome.Hsapiens.NCBI.GRCh38"
 [7] "BSgenome.Hsapiens.UCSC.hg19"
 [8] "BSgenome.Mmusculus.UCSC.mm9"
>
```

BSgenome

シロイヌナズナパッケージの個別インストール方法。基本的にはパッケージ名部分を変更すれば、どのパッケージのインストールにも対応可能です。

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。getSeq関数はBSgenomeオブジェクト 中の「single sequences」というあたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」以外に「chr1_random」や「chrUn_random」なども等価に取扱っている点に注意してください。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. 利用可能な生物種

```
#必要なパッケージを
library(BSgenome)

#本番（利用可能なパ
available.genomes(

#本番（インストール
installed.genomes(
```

7. シロイヌナズナ("BSgenome.Athaliana.TAIR.TAIR9")のゲノム情報をRにインストールしたい場合:

Rのパッケージをインストール後、ゲノム配列をmulti-FASTAファイルで保存する一連の手順です。The Arabidopsis Information Resource (TAIR)(Lamesch et al., Nucleic Acids Res., 2012)から得られる最新バージョンはTAIR10ですが、アセンブリ結果自体はTAIR9と同じと明記されています（README_whole_chromosomes.txt）。

```
param <- "BSgenome.Athaliana.TAIR.TAIR9"#パッケージ名を指定

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

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

BSgenome

2013年12月にリリースされたヒトゲノム最新版(GRCh38)のRパッケージを入力、multi-FASTAファイルを出力として得る。作業ディレクトリはどこでもよいが基本はデスクトップ上のhoge。数分かかるが、約3.3GBのファイルが生成される。**決してテキストエディタで開かないで!**

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザ
lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans
(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、
(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.mus
ンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)
な生物種が利用可能であることがわかります。getSeq関数はBSgenomeオブジェクト 中の「single sequences」という
あたりにリストアップされているchr、というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」
以外に「chr1_random
「ファイル」-「ディレクト

1. 利用可能な生物種とR

```
#必要なパッケージを
library(BSgenome)

#本番（利用可能なパ
available.genomes

#本番（インストール
installed.genomes

#後処理（パッケージ
installed.genomes
```

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTAファイルで保存したい場合:

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconductor ver. 2.14以上の環境で実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

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

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

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

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

BSgenome

出力ファイルの内容はfastaオブジェクトに格納されている。慣れればfastaオブジェクトの中身を眺めるほうが全体像をつかみやすい。

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTA

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconductor ver. 2.14以上の環境で実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

R Console

```
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", widt$
> fasta
  A DNAStringSet instance of length 455
          width seq                    names
  [1] 248956422 NNNNNNNNN...NNNNNNNNN 1
  [2] 242193529 NNNNNNNNN...NNNNNNNNN 2
  [3] 198295559 NNNNNNNNN...NNNNNNNNN 3
  [4] 190214555 NNNNNNNNN...NNNNNNNNN 4
  [5] 181538259 NNNNNNNNN...NNNNNNNNN 5
  ...       ... ...
[451]    200773 TCTACTCTC...GGGGAATTC HSCHR19KIR_FH08_B...
[452]    170148 TTTCTTTCT...GGGGAATTC HSCHR19KIR_FH13_A...
[453]    215732 TGTGGTGAG...GGGGAATTC HSCHR19KIR_FH13_B...
[454]    170537 TCTACTCTC...GGGGAATTC HSCHR19KIR_FH15_A...
[455]    177381 GATCTATCT...GGGGAATTC HSCHR19KIR_RP5_B_...
> |
```

BSgenome

1～22番染色体のみ取扱いたい場合。染色体番号の数が大きくなるほど配列長が短くなっている傾向が一目瞭然ですね。

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTAファイ

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconductor ver. 2.14以上の環境で実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

R Console

```
[454]    170537 TCTACTCTC...GGGGAATTC HSCHR19KIR_FH15_A...
[455]    177381 GATCTATCT...GGGGAATTC HSCHR19KIR_RP5_B_...
> fasta[1:22]
  A DNAStringSet instance of length 22
      width seq                    names
 [1] 248956422 NNNNNNNNNN...NNNNNNNNN 1
 [2] 242193529 NNNNNNNNNN...NNNNNNNNN 2
 [3] 198295559 NNNNNNNNNN...NNNNNNNNN 3
 [4] 190214555 NNNNNNNNNN...NNNNNNNNN 4
 [5] 181538259 NNNNNNNNNN...NNNNNNNNN 5
 ...       ... ...
[18]  80373285 NNNNNNNNNN...NNNNNNNNN 18
[19]  58617616 NNNNNNNNNN...NNNNNNNNN 19
[20]  64444167 NNNNNNNNNN...NNNNNNNNN 20
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
>
```

BSgenome

X, Y, およびミトコンドリア配列も含めたい場合。配列の並びの確認は試行錯誤。最初からわかっていたわけではありません。R画面上で眺めるほうが、全体像を把握しやすい。

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTAファイル

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconductor v
実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

R Console

```
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
> fasta[1:25]
  A DNAStringSet instance of length 25
         width seq                    names
 [1] 248956422 NNNNNNNNNN...NNNNNNNNN 1
 [2] 242193529 NNNNNNNNNN...NNNNNNNNN 2
 [3] 198295559 NNNNNNNNNN...NNNNNNNNN 3
 [4] 190214555 NNNNNNNNNN...NNNNNNNNN 4
 [5] 181538259 NNNNNNNNNN...NNNNNNNNN 5
 ...       ... ...
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
[23] 156040895 NNNNNNNNNN...NNNNNNNNN X
[24]  57227415 NNNNNNNNNN...NNNNNNNNN Y
[25]     16569 GATCACAGGT...ATCACGATG MT
>
```

BSgenome

X, Y, およびミトコンドリア配列までのサブセットをhoge10.fastaで保存したい場合。①上矢印キーを何回か押して、ファイルに保存するためのコマンドを出し、水色下線部分を変更すればよい。

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTAフ

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconduct
実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

①

R Console

```
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
> fasta[1:25]
  A DNAStringSet instance of length 25
      width seq                    names
 [1] 248956422 NNNNNNNNNN...NNNNNNNNN 1
 [2] 242193529 NNNNNNNNNN...NNNNNNNNN 2
 [3] 198295559 NNNNNNNNNN...NNNNNNNNN 3
 [4] 190214555 NNNNNNNNNN...NNNNNNNNN 4
 [5] 181538259 NNNNNNNNNN...NNNNNNNNN 5
 ...       ... ...
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
[23] 156040895 NNNNNNNNNN...NNNNNNNNN X
[24]  57227415 NNNNNNNNNN...NNNNNNNNN Y
[25]     16569 GATCACAGGT...ATCACGATG MT
> writeXStringSet(fasta, file=out_f, format="fasta", widt$
```

BSgenome

実行後にhoge9.fastaよりも若干ファイルサイズの小さいhoge10.fastaが生成されていることが確認できるはず。**決してテキストエディタで開かないで!**

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmulti-FASTAファイル

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.0とBioconductor
実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

R Console

```
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
> fasta[1:25]
  A DNAStringSet instance of length 25
         width seq                    names
 [1] 248956422 NNNNNNNNNN...NNNNNNNNN 1
 [2] 242193529 NNNNNNNNNN...NNNNNNNNN 2
 [3] 198295559 NNNNNNNNNN...NNNNNNNNN 3
 [4] 190214555 NNNNNNNNNN...NNNNNNNNN 4
 [5] 181538259 NNNNNNNNNN...NNNNNNNNN 5
 ...       ... ...
[21]  46709983 NNNNNNNNNN...NNNNNNNNN 21
[22]  50818468 NNNNNNNNNN...NNNNNNNNN 22
[23] 156040895 NNNNNNNNNN...NNNNNNNNN X
[24]  57227415 NNNNNNNNNN...NNNNNNNNN Y
[25]     16569 GATCACAGGT...ATCACGATG MT
> writeXStringSet(fasta[1:25], file="hoge10.fasta", forma$
>
```

参考

様々な記述形式があります。やらなくていいです。
決してテキストエディタで開かないで!

BSgenome

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome NEW

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyr… セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、ボ… (C… (G.a… ンパン… な生物… あたり… 以外に… 「ファイ…

10. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列のmulti-FASTAファイルで保存したい場合:

一部を抽出して保存するやり方です。このパッケージ中の染色体の並びが既知(chr1, 2, ..., chr22, chrX, chrY, and MT)であるという前提です。

```
out_f <- "hoge10.fasta"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定
param_range <- 1:25                        #抽出したい範囲を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中…
#tmp <- unlist(strsplit(param, "…
tmp <- ls(paste("package", param,…
genome <- eval(parse(text=tmp))
genome

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

#後処理(フィルタリング)
obj <- param_range
fasta <- fasta[obj]
fasta
```

1. 利用…

```
#必…
libr…

#本…
avai…

#本…
inst…

#後…
inst…
```

R Console

```
          width seq                                    names
 [1] 248956422 NNNNNNNNNNN...NNNNNNNNNN 1
 [2] 242193529 NNNNNNNNNNN...NNNNNNNNNN 2
 [3] 198295559 NNNNNNNNNNN...NNNNNNNNNN 3
 [4] 190214555 NNNNNNNNNNN...NNNNNNNNNN 4
 [5] 181538259 NNNNNNNNNNN...NNNNNNNNNN 5
 ...       ... ...
[21]  46709983 NNNNNNNNNNN...NNNNNNNNNN 21
[22]  50818468 NNNNNNNNNNN...NNNNNNNNNN 22
[23] 156040895 NNNNNNNNNNN...NNNNNNNNNN X
[24]  57227415 NNNNNNNNNNN...NNNNNNNNNN Y
[25]     16569 GATCACAGGTC...CATCACGATG MT
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=$
>
```

BSgenome

26番目以降の配列は、ヒトゲノムの一部ではあるものの、まだ割り当てられる染色体が定まっていないものたちです。メタゲノム解析などでヒトゲノムにマップされないリードのみ取扱いたい場合には、利用可能な全配列をマッピング時のリファレンスとして用いるのが自然だと思います。

9. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")のゲノム配列をmul

2013年12月にリリースされたGenome Reference Consortium GRCh38です。R ver. 3.1.
実行可能です。

```
out_f <- "hoge9.fasta"                    #出力ファイル名を指定してou
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)

#前処理(paramで指定したパッケージ中のオ
#tmp <- unlist(strsplit(param, "."),
tmp <- ls(paste("package", param, sep
genome <- eval(parse(text=tmp))
genome

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

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

R Console

```
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", widt$
> fasta
  A DNAStringSet instance of length 455
          width seq                    names
  [1] 248956422 NNNNNNNNN...NNNNNNNNN 1
  [2] 242193529 NNNNNNNNN...NNNNNNNNN 2
  [3] 198295559 NNNNNNNNN...NNNNNNNNN 3
  [4] 190214555 NNNNNNNNN...NNNNNNNNN 4
  [5] 181538259 NNNNNNNNN...NNNNNNNNN 5
  ...       ... ...
[451]    200773 TCTACTCTC...GGGGAATTC HSCHR19KIR_FH08_B...
[452]    170148 TTTCTTTCT...GGGGAATTC HSCHR19KIR_FH13_A...
[453]    215732 TGTGGTGAG...GGGGAATTC HSCHR19KIR_FH13_B...
[454]    170537 TCTACTCTC...GGGGAATTC HSCHR19KIR_FH15_A...
[455]    177381 GATCTATCT...GGGGAATTC HSCHR19KIR_RP5_B_...
> |
```

Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

Rで調べることができます

ヒトゲノム中のCpG出現確率は低い

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

2連続塩基の出現頻度

全選択はCTRLとALTキーを押しながらコードの枠内で**左クリック**。hoge4.faが作業ディレクトリ中にあることを確認してコピペ。

イントロ | 一般 | 2連続塩基の出現頻度情報を取得 NEW

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

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

タイトル通りの出現頻度です。

```
in_f <- "hoge4.fa"                          #入
out_f <- "hoge1.txt"                        #出

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

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

#本番
out <- dinucleotideFrequency(fasta)         #2連

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

R Console

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで$
> fasta                                          #確認してるだけ$
  A DNAStringSet instance of length 4
    width seq                                    names          $
[1]    24 CGGACAGCTCCTCGGCATCCGGAT               contig_1
[2]   103 GTCTGCCTCAAGCG...CAGCACACCCTGTC       contig_2
[3]    65 TGTAGGAGAAGGGC...TATGAGGTCGGGCA       contig_3
[4]    49 CGTGCTGATTCCAC...TACCTATGAACATG       contig_4
>
> #本番
> out <- dinucleotideFrequency(fasta)     #2連続塩基の出$
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out)             #保存したい情報$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, r$
>
```

2連続塩基の出現頻度

出力ファイルは、配列ごと(この場合コンティグごと)に16種類の2連続塩基の出現頻度をカウントしたものです。

イントロ | 一般 | 2連続塩基の出現頻度情報を取得 NEW

multi-FASTA形式ファイルを読み込んで、"AA", "AC", "AG", "AT
"GT", "TA", "TC", "TG", "TT" の計$4^2 = 16$通りの2連続塩基の出
で"CG"の割合が期待値よりも低い(Lander et al., 2001; Saxonov et
「ファイル」ー「ディレクトリの変更」で解析したいファイルを置いてあ

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

タイトル通りの出現頻度です。

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

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

#ファイルに保存
tmp <- cbind(na
write.table(tmp
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

出力:hoge1.txt

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

2連続塩基の出現確率

出力ファイルは、配列ごと(この場合コンティグごと)に16種類の2連続塩基の出現確率をカウントしたものです。

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

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"                          #入力ファ
out_f <- "hoge2.txt"                        #出力ファ

#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
out <- dinucleotideFrequency(fasta, as.prob=T)#2

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

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

出力:hoge2.txt

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

2連続塩基の出現確率

ヒトゲノムRパッケージを入力とすることもできます。一見ややこしいですが、fastaオブジェクトの作成までを「お約束の手順」だと思えばいいのです。

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

出現頻度ではなく、出現確率を得るやり方です。

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

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

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

#本番
out <- dinucleotideFrequenc

#ファイルに保存
tmp <- cbind(names(fasta),
write.table(tmp, out_f, sep
```

7. BSgenomeパッケージ中のヒトゲノム配列("BSgenome.Hsapiens.NCBI.GRCh38")の場合:

2013年12月にリリースされたGenome Reference Consortium GRCh38です。出力は出現確率です。

```
out_f <- "hoge7.txt"                          #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定

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

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利用可能なオブジェク
genome <- eval(parse(text=tmp))               #文字列tmpをRオブジェクトとしてgenomeに格納(パッケー
fasta <- getSeq(genome)                       #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome)              #description情報を追加している
fasta                                         #確認してるだけです

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

#ファイルに保存
tmp <- cbind(names(fasta), out)               #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
```

2連続塩基の出現確率

2分強かかります。CGの連続塩基が他に比べて確かに低いことがわかります。

出力：hoge7.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
1	9.5%	5.0%	7.1%	7.4%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.9%	3.8%	4.5%	5.0%	7.3%	5.8%	7.1%	10.6%
5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.8%	5.1%	6.9%	5.9%	7.2%	10.3%
6	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
7	9.8%	5.0%	7.0%	7.7%	7.3%	5.1%	1.0%	7.0%	6.0%	4.2%	5.1%	5.1%	6.5%	5.9%	7.3%	10.0%
8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	6.0%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
9	9.7%	5.1%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	6.0%	4.3%	5.3%	5.0%	6.4%	6.0%	7.3%	9.7%
10	9.6%	5.0%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.0%	4.4%	5.3%	5.1%	6.3%	6.0%	7.4%	9.7%
11	9.5%	5.1%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.1%	4.3%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
12	9.8%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	6.0%	7.2%	9.9%
13	10.5%	5.0%	6.8%	8.4%	7.1%	4.5%	0.9%	6.7%	5.9%	3.8%	4.6%	5.0%	7.2%	5.8%	7.1%	10.6%
14	9.7%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
15	9.4%	5.1%	7.1%	7.3%	7.3%	5.4%	1.1%	7.1%	6.0%	4.5%	5.5%	5.1%	6.1%	6.0%	7.4%	9.5%
16	8.6%	5.1%	7.3%	6.7%	7.5%	6.1%	1.4%	7.2%	6.1%	5.0%	6.1%	5.1%	5.4%	6.1%	7.6%	8.8%
17	8.5%	5.1%	7.3%	6.4%	7.4%	6.3%	1.5%	7.4%	6.2%	5.1%	6.4%	5.0%	5.2%	6.1%	7.5%	8.6%
18	10.1%	5.1%	7.0%	7.9%	7.2%	4.7%	0.9%	6.9%	6.1%	4.0%	4.9%	5.1%	6.7%	5.9%	7.3%	10.3%

2連続塩基の出現頻度と確率

染色体ごとではなく、全てをひとまとめにするやり方です。連続塩基の出現頻度順にソートしてCGが少ないことを確かめています。

8. BSgenomeパッケージ中のヒトゲノム配列("BSgenome.Hsapiens.NCBI.GRCh38")の場合:

全配列を合算して、連続塩基ごとの出現頻度(frequency)と出現確率(probability)を出力するやり方です。dinucleotideFrequency関数中の「simplify.as="collapsed"」オプションでも一応実行できますが、桁が多くなりすぎて「整数オーバーフロー」問題が起きたのでやめてます。

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

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利用可能なオブジ
genome <- eval(parse(text=tmp))        #文字列
fasta <- getSeq(genome)                #ゲノム
names(fasta) <- seqnames(genome)       #descr
fasta                                  #確認し

#本番
hoge <- dinucleotideFrequency(fasta, as.prob=
frequency <- colSums(hoge)             #列ごと
probability <- frequency / sum(frequency)#出現
frequency                              #中身を
sort(frequency, decreasing=F)          #値の小

#ファイルに保存
tmp <- cbind(names(frequency), frequency, pro
write.table(tmp, out_f, sep="\t", append=F, q
```

R Console

```
       TT
299351073
> sort(frequency, decreasing=F)            #値の小さい$
       CG        GC        AC        GT        CC
 30979743 130065644 153830681 154194068 158048073
       GG        TC        GA        TA        CT
159302235 181782675 182772932 197567087 213517855
       AG        CA        TG        AT        AA
213785914 221181041 222266728 233904527 296763858
       TT
299351073
>
> #ファイルに保存
> tmp <- cbind(names(frequency), frequency, probabil$
> write.table(tmp, out_f, sep="\t", append=F, quote=$
>
```

k連続塩基解析

2連続塩基の解析は、k=2のときのk連続塩基の解析(k-mer解析)と同じです。

- 比較ゲノム解析
  - k=3 or 4付近の値を用いてゲノムごとの頻度情報を取得し、類似性尺度として利用
- アセンブル(ゲノムやトランスクリプトーム)
  - k=25～50付近の値を用いてde Bruijnグラフを作成
  - k-mer頻度グラフを作成して眺め、Heterozygosityの有無などを調査
- モチーフ解析
  - 転写開始点の上流配列解析。古細菌の上流50塩基に絞ってk=4で出現頻度解析すると、おそらくTATAが上位にランクイン
- 発現量推定
  - RNA-seq解析で、リファレンスにリードをマップしてリード数をカウントするのが主流だが、マッピング作業をすっ飛ばしてk-merに基づく方法で定量。Sailfish (Patro et al., *Nat Biotechnol.*, 2014)やRNA-Skim (Zhang and Wang, *Bioinformatics*, 2014)。

Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

(遺伝子)アノテーション

遺伝子ごとに、どの染色体のどの座標上に存在するのかなどの情報を含むタブ区切りテキストファイル。
GFF/GTF形式ファイルが有名です。

- イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013) (last modified 2014/06/26)
- イントロ | NGS | 配列取得 | シミュレーションデータ | について (last modified 2015/01/18)
- イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成から (last modified 2015/01/18)
- イントロ | NGS | アノテーション情報取得 | について (last modified 2014/03/26)
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル (last modified 2014/04/11)
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル (last modified 2013/09/25)
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck 2009) (last modified 2013/09/26)
- イントロ | NGS | アノテーション
- イントロ | NGS | アノテーション
- イントロ | NGS | アノテーション
- イントロ | NGS | アノテーション
- イントロ | NGS | 読み込み | F
- イントロ | NGS | 読み込み | F
- イントロ | NGS | 読み込み | F
- イントロ | NGS | 読み込み | F

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

多くの生物種についてEnsembl (Flicek et al., Nucleic Acids Res., 2014) のFTPサイトからGTF形式(GFF ver. 2)の遺伝子アノテーションファイルを得ることができます。GTFはGeneral Transfer FormatまたはGene Transfer Formatの略で、GTFの派生版としてGTF2というフォーマットもあるようです。また、General Feature Format ver. 3 (GFF3)という形式も存在するなど、GFF/GTF形式として総称されている中で様々なバリエーションがあります。いずれもrefFlat形式同様、どの領域にどの遺伝子があるのかという座標(Coordinates)情報を含みます。ゲノム配列のバージョンと同じであることを確認した上で用いましょう。

- Ensembl (Flicek et al., Nucleic Acids Res., 2014)
  圧縮(gzip)ファイル形式です。基本はFTPサイトです。代表的なものを以下にリストアップしています。
  - ヒト ; Human(H.sapiens)
  - ラット ; Rat(R.norvegicus)
  - ネコ; Cat(F.catus)
  - ウサギ ; Rabbit(O.cuniculus)
  - ニワトリ; Chicken(G.gallus)
  - イヌ; Dog(C.familiaris)
  - ウマ; Horse(E.caballus)
  - ゼブラフィッシュ ; Zebrafish (D.rerio)
  - ...
- イネ : RAP-DB (Sakai et al., Plant Cell Physiol., 2013)
  - 「ダウンロード」-「Gene set」-「Gene structure and function information in GFF format」-「Download」。IRGSP-1.0_representative_2014-03-05.tar.gz (12.4MB程度)の圧縮ファイルが得られます。
- シロイヌナズナ : The Arabidopsis Information Resource (TAIR) (Lamesch et al., Nucleic Acids Res., 2012)
  - 「ダウンロード」-「Genes」-「TAIR10_genome_release」-「TAIR10_gff3」の TAIR10_GFF3_genes.gff (42MB程度)

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

(遺伝子)アノテーション

現状では不具合が多く存在します。RではTranscriptDbという形式のオブジェクトとして利用するらしい、という程度の認識でいいと思います。

- イントロ | NGS | アノテーション情報取得 | について (last modified 2014/03/26)
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル (last modified 2014/04/11)
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル (last modified 2013/09/25)
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck 2009) (last modified 2013/09/26)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | について (last modified 2014/03/28)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*から (last modified 2015/02/19) NEW
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | 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 | TranscriptDb | GFF/GTF形式ファイルから (last modified 2014/04/01)
- イントロ | NGS | 読み込
- イントロ | NGS | 読み込

イントロ | NGS | アノテーション情報取得 | TranscriptDb | について

アノテーション情報を取り扱うためにRでよく利用されるオブジェクトは、TranscriptDb以外に、RangedData、GRangesListなどなどが挙げられますが、このウェブページでは、TranscriptDbというオブジェクトをアノテーション情報の基本オブジェクトとし、txdbというオブジェクト名で統一して取り扱っています。このtxdbに対して、GenomicFeaturesで利用可能なtranscripts, exons, cds, genes, promoters, disjointExons, microRNAs, tRNAsなどの関数を適用して、GRanges形式のオブジェクトを得ることができます。出力情報を制限したい場合には、transcriptsByOverlaps, exonsByOverlaps, cdsByOverlapsなどの関数が利用可能です。GRanges形式やGRangesList形式オブジェクトの取り扱いはGenomicRangesに記載されています。

- GenomicFeatures: Lawrence et al., PLoS Comput. Biol., 2013
- GenomicRanges: Lawrence et al., PLoS Comput. Biol., 2013

(遺伝子)アノテーション

目的のアノテーション情報がTxDb.から始まる名前のRパッケージとして提供されている場合はそれを利用するのが基本。

イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*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、"TranscriptDb"という形式のオブジェクトを利用する必要があります。TranscriptDbオブジェクトは、makeTranscriptDbFromGFF関数を用いてGTF形式ファイルを入力として作成することも可能ですが、最も手っ取り早いやり方はTranscriptDbオブジェクト形式で格納されている"TxDb.*"という名前のパッケージを利用することです。利用可能なTxDb.*パッケージはここにリストアップされているものたちです。

(1)「全パッケージリスト(All Packages)」中の、
(2)「AnnotationDataの左側のさんかく」、
(3)「PackageTypeの左側のさんかく」、
(4)「TxDb」からも辿れます。

ここでは、いくつかのパッケージの読み込みまでを示します。

1. TxDb.Hsapiens.UCSC.hg19.knownGene(ヒト)の場合:

hg19 (Genome Reference Consortium GRCh37のことらしい)です。

```
param <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定

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

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

(遺伝子)アノテーション

2015年2月現在、ヒト、マウス、ラット、酵母、線虫など、主要なモデル生物についてのみTxDbパッケージが提供されています。

イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*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、"TranscriptDb"という形式のオブジェクトを利用する必要があります。TranscriptDbオ
形式ファイルを入力として作成することも可能ですが、
納されている"TxDb.*"という名前のパッケージを利用す
アップされているものたちです。
(1)「全パッケージリスト(All Packages)」中の、
(2)「AnnotationDataの左側のさんかく」、
(3)「PackageTypeの左側のさんかく」、
(4)「TxDb」からも辿れます。
こ いくつかのパッケージの読み込みまでを示し

1. TxDb.Hsapiens.UCSC.hg19.knownGene(ヒト)の場合

hg19 (Genome Reference Consortium GRCh37のことら

```
param <- "TxDb.Hsapiens.UCSC.hg19.knownG

#必要なパッケージをロード
library(param, character.only=T)

#前処理
#tmp <- unlist(strsplit(param, ".", fixe
tmp <- ls(paste("package", param, sep=":
txdb <- eval(parse(text=tmp))
txdb
```

All Packages

Bioconductor version 3.0 (Release)

Autocomplete biocViews search:

- CustomDBSchema (11)
- FunctionalAnnotation (13)
- Organism (532)
- PackageType (524)
  - BSgenome (69)
  - cdf (126)
  - ChipDb (155)
  - db0 (19)
  - InparanoidDb (8)
  - MeSHDb (3)
  - OrganismDb (3)
  - OrgDb (19)
  - probe (104)
  - SNPlocs (1)
  - TxDb (17)
- SequenceAnnotation (3)
- ExperimentData (223)

Packages found under TxDb:

Show All entries Search table:

Package	Maintainer	Title
FDb.UCSC.tRNAs	Biocore Data Team	Annotation package for FeatureDb object(s)
TxDb.Athaliana.BioMart.plantsmart22	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Celegans.UCSC.ce6.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Dmelanogaster.UCSC.dm3.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg18.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg38.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm9.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn4.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn5.refGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer2.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)

(遺伝子)アノテー

イントロ | NGS | アノテーション情報取得 |

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
オブジェクトを利用する必要があります。TranscriptDbオブジェク
形式ファイルを入力として作成することも可能ですが、最も手っ
納されている"TxDb.*"という名前のパッケージを利用することで
アップされているものたちです。
(1)「全パッケージリスト(All Packages)」中の、
(2)「AnnotationDataの左側のさんかく」、
(3)「PackageTypeの左側のさんかく」、
(4)「TxDb」からも辿れます。
ここでは、いくつかのパッケージの読み込みまでを示します。

1. TxDb.Hsapiens.UCSC.hg19.knownGene(ヒト)の場合:

hg19 (Genome Reference Consortium GRCh37のことらしい)で

```
param <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#
#必要なパッケージをロード
library(param, character.only=T)          #param
#前処理
#tmp <- unlist(strsplit(param, ".", fixed=TRUE
tmp <- ls(paste("package", param, sep=":"))#pa
txdb <- eval(parse(text=tmp))             #文字列
txdb                                      #確認し
```

Package		
FDb.UCSC.tRNAs		
TxDb.Athaliana.BioMart.plantsmart22		
TxDb.Celegans.UCSC.ce6.ensGene		
TxDb.Dmelanogaster.UCSC.dm3.ensGene	Package Maintainer	TxDb object(s)
TxDb.Hsapiens.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg18.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg38.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm9.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn4.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn5.refGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer2.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)

ヒトゲノムhg19に対応するUCSC known genesのアノテーションパッケージを読み込んでtxdbという名前のTranscriptDb形式のオブジェクトを得る。ここで重要なのは、**エラーなく読み込めることを確認**すること。この一点のみです。

(遺伝子)アノテーション

1. TxDb.Hsapiens.UCSC.hg19.knownGene(ヒト)の場合:

hg19 (Genome Reference Consortium GRCh37のこと

```
param <- "TxDb.Hsapiens.UCSC.hg19.know

#必要なパッケージをロード
library(param, character.only=T)

#前処理
#tmp <- unlist(strsplit(param, ".", fi
tmp <- ls(paste("package", param, sep=
txdb <- eval(parse(text=tmp))
txdb
```

R Console

```
> #tmp <- unlist(strsplit(para
> tmp <- ls(paste("package", p
> txdb <- eval(parse(text=tmp)
> txdb
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Organism: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: GRCh37
| transcript_nrow: 82960
| exon_nrow: 289969
| cds_nrow: 237533
| Db created by: GenomicFeatures package from Bioconduct$
| Creation time: 2014-03-17 16:15:59 -0700 (Mon, 17 Mar $
| GenomicFeatures version at creation time: 1.15.11
| RSQLite version at creation time: 0.11.4
| DBSCHEMAVERSION: 1.0
>
```

全選択はCTRLとALTキーを押しながらコードの枠内で**左クリック**。以下のようにTranscriptDbオブジェクトの中身が見られる状態になっていればOK。ヒトゲノムの最新版(hg38)ではないが、hg19は長らく利用されてきたバージョンであるため、推奨手順通りにやるとデフォルトでインストールされている。

(遺伝子)アノテーション

遺伝子IDはEnsembl Gene ID, RefSeq Gene ID, Entrez Gene IDなどいろいろある。knownGeneというのはEntrez Gene IDのことなのだろうと解釈する。また、転写物数は82,960個なのだろうと妄想する。

1. TxDb.Hsapiens.UCSC.hg19.knownGeneの場合:

hg19 (Genome Reference Consortium GRCh37のこと

```
param <- "TxDb.Hsapiens.UCSC.hg19.know

#必要なパッケージをロード
library(param, character.only=T)

#前処理
#tmp <- unlist(strsplit(param, ".", fi
tmp <- ls(paste("package", param, sep=
txdb <- eval(parse(text=tmp))
txdb
```

R Console

```
> #tmp <- unlist(strsplit(para
> tmp <- ls(paste("package", param, sep=":"))  #paramで指$
> txdb <- eval(parse(text=tmp))                #文字列tmpをR才$
> txdb                                         #確認してるだけ$
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Organism: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: GRCh37
| transcript_nrow: 82960
| exon_nrow: 289969
| cds_nrow: 237533
| Db created by: GenomicFeatures package from Bioconduct$
| Creation time: 2014-03-17 16:15:59 -0700 (Mon, 17 Mar $
| GenomicFeatures version at creation time: 1.15.11
| RSQLite version at creation time: 0.11.4
| DBSCHEMAVERSION: 1.0
>
```

参考

(遺伝子)アノテー

ラットゲノムrn5に対応したRefSeq Gene IDに基づくアノテーションパッケージを読み込んでみる。同じR Console画面上でコピペしてよい。

2. TxDb.Rnorvegicus.UCSC.rn5.refGene(ラット)の場合:

2015年2月現在、Rで取得可能なラットの最新版です。

```
param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#

#必要なパッケージをロード
library(param, character.only=T)        #param

#前処理
#tmp <- unlist(strsplit(param, ".", fixed=TRU
tmp <- ls(paste("package", param, sep=":"))#pa
txdb <- eval(parse(text=tmp))           #文字列
txdb                                    #確認し
```

Package		
FDb.UCSC.tRNAs	Bioco	
TxDb.Athaliana.BioMart.plantsmart22	Bioco Pack	
TxDb.Celegans.UCSC.ce6.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Dmelanogaster.UCSC.dm3.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg18.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Hsapiens.UCSC.hg38.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm10.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Mmusculus.UCSC.mm9.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.BioMart.igis	Gabriel Becker	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn4.ensGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Rnorvegicus.UCSC.rn5.refGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer2.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
TxDb.Scerevisiae.UCSC.sacCer3.sgdGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)

R Console

参考

```
> param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名を指定
>
> #必要なパッケージをロード
> library(param, character.only=T)          #paramで指定したパッケージの読み込$
 以下にエラー library(param, character.only = T) :
   'TxDb.Rnorvegicus.UCSC.rn5.refGene' という名前のパッケージはありません
>
> #前処理
> #tmp <- unlist(strsplit(param, ".", fixed=TRUE))[2]#paramで指定した文
> tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利$
 以下にエラー as.environment(pos) :
   検索リストに "package:TxDb.Rnorvegicus.UCSC.rn5.refGene" という項目はあ$
> txdb <- eval(parse(text=tmp))             #文字列tmpをRオブジェクトとしてtxd$
> txdb                                      #確認してるだけです
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Organism: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: GRCh37
| transcript_nrow: 82960
| exon_nrow: 289969
| cds_nrow: 237533
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2014-03-17 16:15:59 -0700 (Mon, 17 Mar 2014)
| GenomicFeatures version at creation time: 1.15.11
| RSQLite version at creation time: 0.11.4
| DBSCHEMAVERSION: 1.0
>
```

①

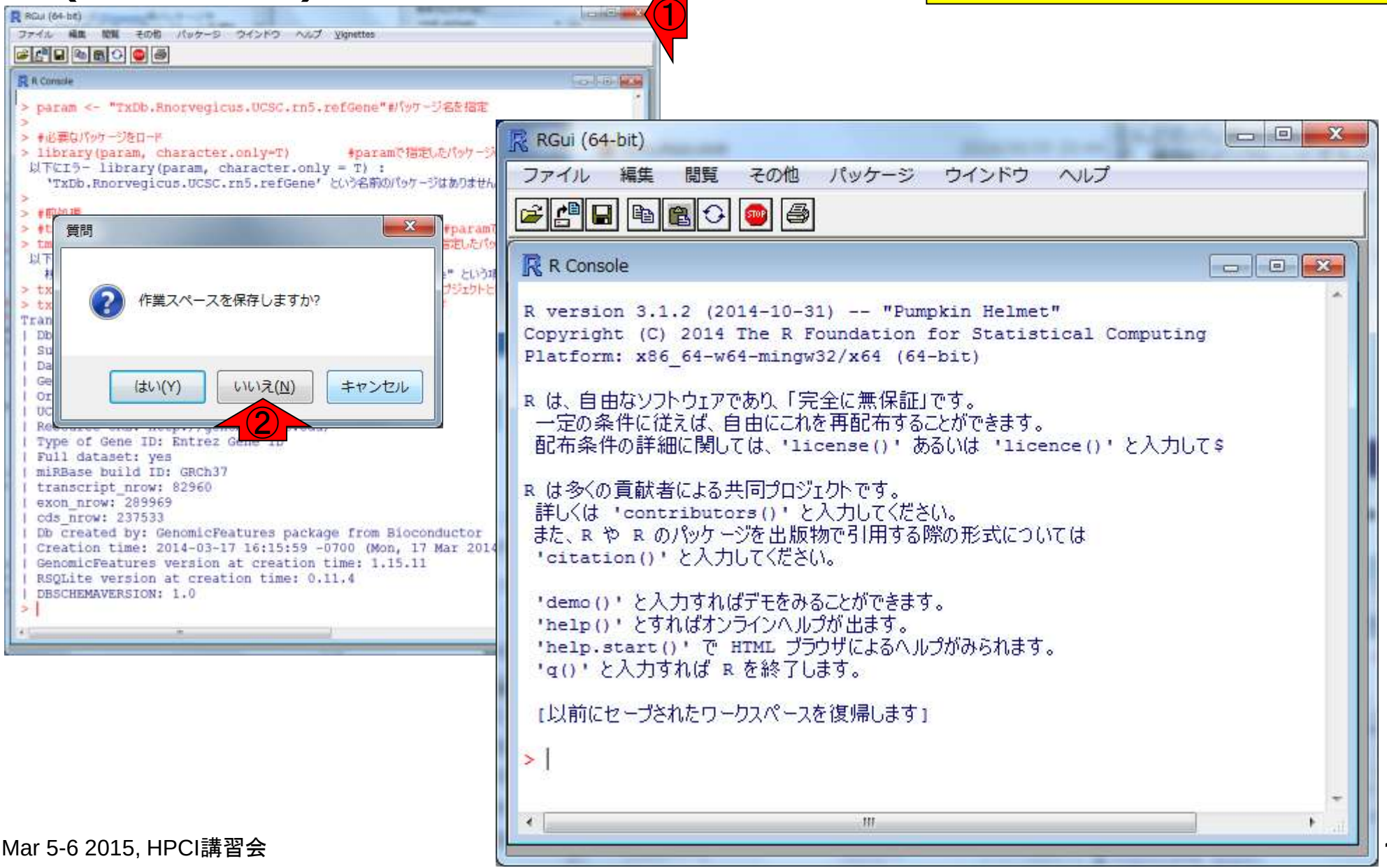
②

①パッケージのインストールができていないので、エラーが出ていることがわかる。②txdbオブジェクトの中身を見ることはできているが、これは以前に得たヒトのアノテーション情報なので別物です。この種の失敗を防ぐための一番手っ取り早い方法は、Rを一旦終了させて何もなかった状態で再度コピペすることです。

参考

もう少し真面目なオブジェクトの消去のやり方はNGS速習コースの3-4を参照(2014年9月9日の資料)。

(遺伝子)アノテーション



(遺伝子)アノテーション

真っ新な状態での実行結果。
確かにtxdbオブジェクトが作成されていないことがわかる。

2. TxDb.Rnorvegicus.UCSC.rn5.refGene(ラット)の場合:

2015年2月現在、Rで取得可能なラットの最新版です。

```
param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名を指定

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

#前処理
#tmp <- unlist(strsplit(para
tmp <- ls(paste("package", p
txdb <- eval(parse(text=tmp)
txdb
```

R Console

```
> param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名$
>
> #必要なパッケージをロード
> library(param, character.only=T)          #paramで指定したパ$
 以下にエラー library(param, character.only = T) :
   'TxDb.Rnorvegicus.UCSC.rn5.refGene' という名前のパッケ$
>
> #前処理
> #tmp <- unlist(strsplit(param, ".", fixed=TRUE))[2]#param$
> tmp <- ls(paste("package", param, sep=":"))#paramで指定し$
 以下にエラー as.environment(pos) :
   検索リストに "package:TxDb.Rnorvegicus.UCSC.rn5.refGene"$
> txdb <- eval(parse(text=tmp))                #文字列tmpをRオブ$
 以下にエラー parse(text = tmp) : オブジェクト 'tmp' があ$
> txdb                                         #確認してるだけです
 エラー: オブジェクト 'txdb' がありません
>
```

Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

参考

個別パッケージのインストール

ここでの目的は TxDb.Rnorvegicus.UCSC.rn5.refGeneパッケージのインストール。講習会では使わないので、インストールしないでください。

- はじめに (last modified 2014/01/30)
- 参考資料 (講義、講習会、本など) (last modified 2015/01/17)
- 過去のお知らせ (last modified 2015/02/15) NEW
- Rのインストールと起動 (last modified 2014/09/08)
- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified ...) NEW
- バイオインフォマティクス人材
- 書籍 |トランスクリプトーム解

個別パッケージのインストール NEW

Rのインストールと起動に従って推奨手順通りに行えば、ほとんどのパッケージは自動でインストールされます。しかし、多くのBSgenome系パッケージや TxDb系のパッケージは自動ではインストールされませんので、個別にインストールする必要があります。ここでは、それらのパッケージのインストール手順を示します。

1. ゼブラフィッシュゲノムのパッケージ("BSgenome.Drerio.UCSC.danRer7")をインストールしたい場合:

400MB程度あります....。

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

#本番
source("http://bioconductor.org/biocLite.R")#おまじない
biocLite(param, suppressUpdates=TRUE)  #おまじない
```

2. TxDb.Rnorvegicus.UCSC.rn5.refGeneパッケージのインストールをしたい場合:

```
param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名を指定

#本番
source("http://bioconductor.org/biocLite.R")#おまじない
biocLite(param, suppressUpdates=TRUE)  #おまじない
```

参考

個別パッケージのインストール

インストール所要時間は約5分。エラーは出ていないので、該当パッケージを利用可能になっているはず。

2. TxDb.Rnorvegicus.UCSC.rn5.refGeneパッケージのインストールをしたい場合:

```
param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名を指定

#本番
source("http://bioconductor.o
biocLite(param, suppressUpdat
```

R Console

```
> param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名$
>
> #本番
> source("http://bioconductor.org/biocLite.R")#おまじない
Bioconductor version 2.14 (BiocInstaller 1.14.3),
  ?biocLite for help
A newer version of Bioconductor is available for
  this version of R, ?BiocUpgrade for help
> biocLite(param, suppressUpdates=TRUE)  #おまじない
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.14 (BiocInstaller
  1.14.3), R version 3.1.2.
Installing package(s)
  'TxDb.Rnorvegicus.UCSC.rn5.refGene'
 URL 'http://bioconductor.org/packages/2.14/data/annotation$
Content type 'appli
 開かれた URL
downloaded 6.3 Mb


The downloaded bina
        C:\Users\kadota\AppData\Local\Temp\RtmpOsgiLZ\downl$
>
```

71% downloaded

URL: ... ows/contrib/3.1/TxDb.Rnorvegicus.UCSC.rn5.refGene_2.14.0.zip

(遺伝子)アノテーション

2. TxDb.Rnorvegicus.UCSC.rn5.refGene(ラット)の場合:

2015年2月現在、Rで取得可能なラットの最新

```
param <- "TxDb.Rnorvegicus.UCSC.r

#必要なパッケージをロード
library(param, character.only=T)

#前処理
#tmp <- unlist(strsplit(param, ".
tmp <- ls(paste("package", param,
txdb <- eval(parse(text=tmp))
txdb
```

R Console

```
> #前処理
> #tmp <- unlist(strsplit(pa
> tmp <- ls(paste("package",
> txdb <- eval(parse(text=tr
> txdb
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: rn5
| Organism: Rattus norvegicus
| UCSC Table: refGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: NA
| transcript_nrow: 18567
| exon_nrow: 160612
| cds_nrow: 151698
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2014-03-17 16:30:25 -0700 (Mon, 17 Mar 201$
| GenomicFeatures version at creation time: 1.15.11
| RSQLite version at creation time: 0.11.4
| DBSCHEMAVERSION: 1.0
>
```

確かにtxdbオブジェクトがちゃんとできている。しかし2015年2月現在の最新版はBioconductor ver. 3.0なのに、なぜかver. 2.14というメッセージがあった。そして、実際ちょっと情報が古いという疑いを持つ。**結論としては、R ver. 3.1.2インストール後に、古いバージョンのR本体(R ver. 3.1.0)をインストールしたから起きていた通常利用環境下では起きない問題。**以降の数枚のスライドを行って得た結論。

TxDb.Rnorvegicus.UCSC.rn5.refGene

参考

Annotation package for TxDb object(s)

Bioconductor version: Release (3.0)

Exposes an annotation databases generated from UCS

Author: Marc Carlson

Maintainer: Bioconductor Package Maintainer <mainta

Citation (from within R, enter citation("TxDb.Rnorv

Carlson M. *TxDb.Rnorvegicus.UCSC.rn5.refGene: Ar*
version 3.0.0.

Installation

To install this package, start R and enter:

```
source("http://bioconductor.org/biocLite.R")
biocLite("TxDb.Rnorvegicus.UCSC.rn5.refGene
```

Documentation

PDF		Reference Manual

Details

biocViews	AnnotationData, Genetics, Ratt
Version	3.0.0
License	Artistic-2.0
Depends	GenomicFeatures(>= 1.17.17)
Imports	GenomicFeatures, AnnotationD
Suggests	
System Requirements	
URL	
Depends On Me	Rattus.norvegicus
Imports Me	
Suggests Me	BSgenome.Rnorvegicus.UCSC.

R Console

```
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: rn5
| Organism: Rattus norvegicus
| UCSC Table: refGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: NA
| transcript_nrow: 18567
| exon_nrow: 160612
| cds_nrow: 151698
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2014-03-17 16:30:25 -0700 (Mon, 17 Mar 201$
| GenomicFeatures version at creation time: 1.15.11
| RSQLite version at creation time: 0.11.4
| DBSCHEMAVERSION: 1.0
> packageVersions(param)
 エラー:  関数 "packageVersions" を見つけることができません$
> packageVersion(param)
[1] '2.14.0'
> param
[1] "TxDb.Rnorvegicus.UCSC.rn5.refGene"
>
```

packageVersion関数を用いて、ロードしたパッケージのバージョン情報を知る。最新バージョン(3.0.0)のインストールができていないという確定診断がこの段階で下される。

• 個別パッケージのインストール 参考

個別パッケージのインス

2. TxDb.Rnorvegicus.UCSC.rn5.refGeneパッケージのインストールをしたい場合:

```
param <- "TxDb.Rnorvegicus.UCSC.rn5.refGene"#パッケージ名を指定

#本番
source("http://bioconductor.o
biocLite(param, suppressUpdat
```

改めてパッケージをインストールしたときのことを思い出す。このPCのR本体のバージョンは3.1.2なので、Bioconductor 3.0になっているはずだが、なぜかBioconductor 2.14時に配布されたパッケージのバージョンになっている。門田の別のPCでは、正しく「Bioconductor version 3.0 (BiocInstaller 1.16.1),」となっていたものもあった。

R Console

```
> param <- "TxDb.Rnorvegic
>
> #本番
> source("http://bioconductor.org/biocLite.R")#おまじない
Bioconductor version 2.14 (BiocInstaller 1.14.3),
  ?biocLite for help
A newer version of Bioconductor is available for
  this version of R, ?BiocUpgrade for help
> biocLite(param, suppressUpdates=TRUE)   #おまじない
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.14 (BiocInstaller
  1.14.3), R version 3.1.2.
Installing package(s)
  'TxDb.Rnorvegicus.UCSC.rn5.refGene'
 URL 'http://bioconductor.org/packages/2.14/data/annotation$
Content type 'application/zip' length 6635703 bytes (6.3 Mb)
 開かれた URL
downloaded 6.3 Mb


The downloaded binary packages are in
        C:\Users\kadota\AppData\Local\Temp\RtmpOsgiLZ\downl$
>
```

原因究明

参考

Rを再起動して、該当部分をもう一度実行しても同じメッセージが出ることを確認。メッセージに従って、?BiocUpgradeをやってみる。何を書いているのかよく分からないが赤枠部分をコピペすればよさそう。

R Console

```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical C
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

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

 [以前にセーブされたワークスペースを復帰します]

> source("http://bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.3), ?bi
A newer version of Bioconductor is available for this
  ?BiocUpgrade for help
> ?BiocUpgrade
starting httpd help server ... done
>
```

BiocUpgrade {BiocInstaller} R Documentation

Upgrade Bioconductor to the latest version available for this version of R

Description

Downloads the latest version of the BiocInstaller package, and upgrades all currently installed packages to the latest repositories **for this version of R**.

To upgrade, use:

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

See Also

biocLite Installs/updates Bioconductor/CRAN packages.

chooseBioCmirror lets you choose from a list of all public Bioconductor mirror URLs.

chooseCRANmirror lets you choose from a list of all public CRAN mirror URLs.

biocinstallRepos returns the Bioconductor and CRAN repositories used by biocLite.

install.packages installs the packages themselves.

Examples

```
## Not run:
source("http://bioconductor.org/biocLite.R")
biocLite("BiocUpgrade")

## End(Not run)
```

[Package *BiocInstaller* version 1.14.3 Index]

参考 赤枠部分をコピペ。

BiocUpgrade {BiocInstaller} R Documentation

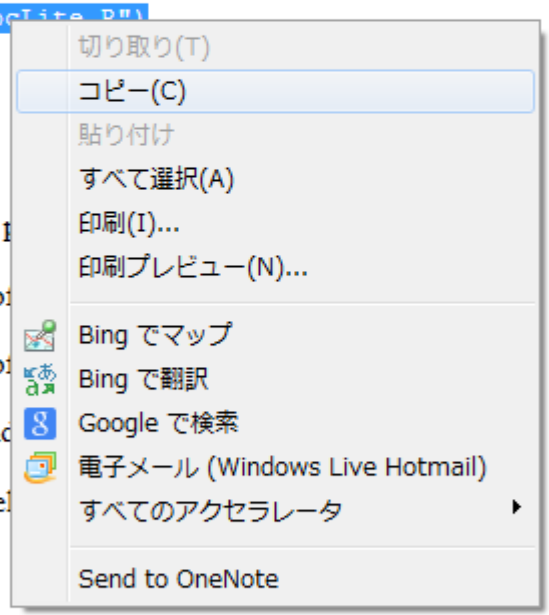
Upgrade Bioconductor to the latest version available for this version of R

Description

Downloads the latest version of the BiocInstaller package, and upgrades all currently installed packages to the latest repositories **for this version of R**.

To upgrade, use:

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



See Also

biocLite Installs/updates Bioconductor/CRAN

chooseBioCmirror lets you choose from a list of

chooseCRANmirror lets you choose from a list of

biocinstallRepos returns the Bioconductor and

install.packages installs the packages themsel

Examples

```
## Not run:
source("http://bioconductor.org/biocLite.R")
biocLite("BiocUpgrade")

## End(Not run)
```

[Package *BiocInstaller* version 1.14.3 Index]

BiocUpgrade {BiocInstaller} R Documentation

Upgrade Bioconductor to the for this ver

Description

Downloads the latest version of the BiocInstaller installed packages to the latest repositories **for t**

To upgrade, use:

```
source("http://bioconductor.org/bio
biocLite("BiocUpgrade")
```

See Also

biocLite Installs/updates Bioconductor/CRAN

chooseBioCmirror lets you choose from a list c

chooseCRANmirror lets you choose from a list c

biocinstallRepos returns the Bioconductor an

install.packages installs the packages themse

Examples

```
## Not run:
source("http://bioconductor.org/biocLit
biocLite("BiocUpgrade")

## End(Not run)
```

[Package *BiocInstaller* ve

参考

R Console

```
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()' あるいは 'licence()' $

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

[以前にセーブされたワークスペースを復帰します]

> source("http://bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.3), ?biocLite f$
A newer version of Bioconductor is available for this version$
  ?BiocUpgrade for help
> ?BiocUpgrade
starting httpd help server ... done
> source("http://bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.3),
  ?biocLite for help
A newer version of Bioconductor is available for this
  version of R, ?BiocUpgrade for help
>     biocLite("BiocUpgrade")
Upgrade all packages to Bioconductor version 3.0? [y/n]:    y
```

バージョンアップするかどうかを聞かれているのでyを押してリターン。

参考

原因究明

全パッケージのアップグレードなのでそれなりの時間がかかる。赤下線のフォルダ中にRの個別のパッケージがインストールされることがわかる。

R Console

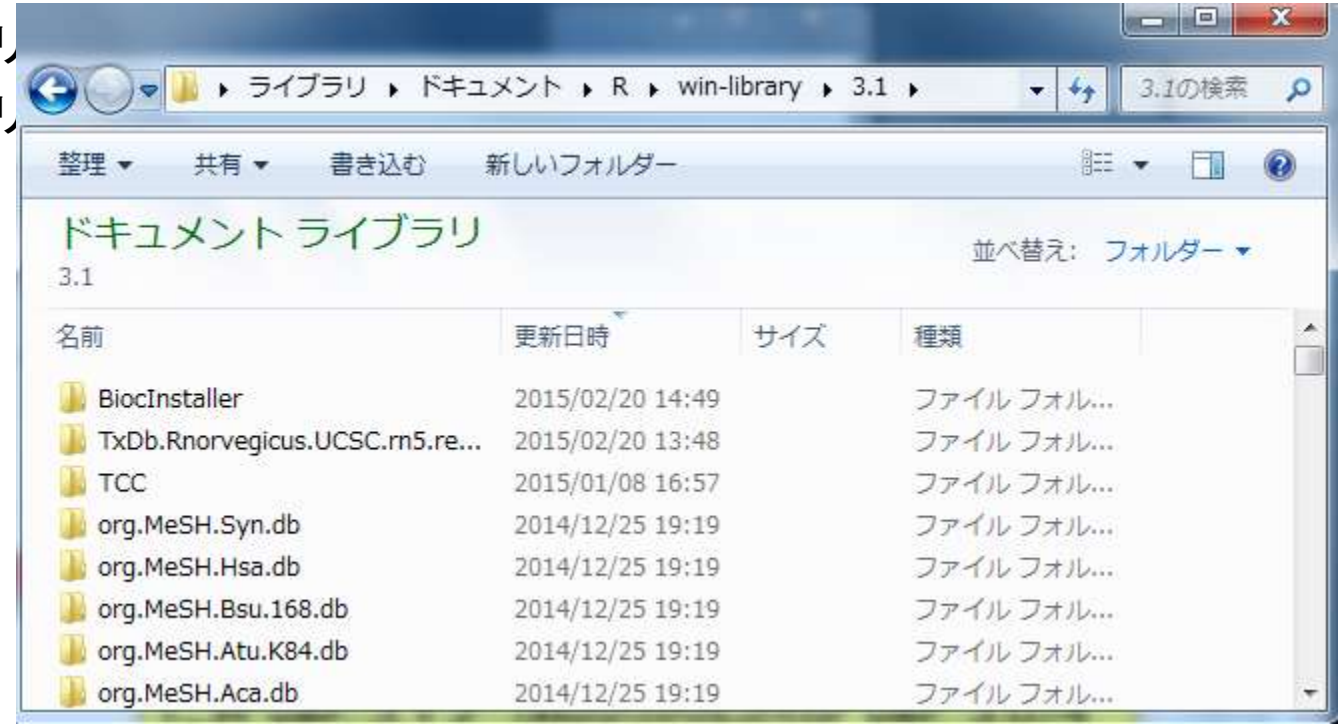
```
> ?BiocUpgrade
starting httpd help server ... done
> source("http://bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.3),
  ?biocLite for help
A newer version of Bioconductor is available for this
  version of R, ?BiocUpgrade for help
>     biocLite("BiocUpgrade")
Upgrade all packages to Bioconductor version 3.0? [y/n]:    y
Upgrade all packages to Bioconductor version 3.0? [y/n]: y
Installing package into 'C:/Users/kadota/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
 URL 'http://bioconductor.org/packages/3.0/bioc/bin/windows/contrib/3.1/BiocIns$
Content type 'application/zip' length 109541 bytes (106 Kb)
 開かれた URL
downloaded 106 Kb

package 'BiocInstaller' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\kadota\AppData\Local\Temp\RtmpGYwNH1\downloaded_packages
Bioconductor version 3.0 (BiocInstaller 1.16.1), ?biocLite
  for help
'BiocInstaller' changed to version 1.16.1
BioC_mirror: http://bioconductor.org
Using Bioconductor version 3.0 (BiocInstaller 1.16.1), R
  version 3.1.2.
```

参考

R本体とBioconductor

- Bioconductorは半年ごとにリリース
  - 2014-10にver. 3.0をリリース (R ver. 3.1.1で動作確認
  - 2014-04にver. 2.14をリリース (R ver. 3.1.0で動作確
  - 2013-10にver. 2.13をリリース (R ver. 3.0で動作確認
  - 2013-04にver. 2.12をリリース (R ver. 3.0で動作確認
  - 2012-10にver. 2.11をリリース (R ver. 2.15.1で動作確認)、提供パッケージ数:608
  - 2012-04にver. 2.10をリリース (R ver. 2.15.0で動作確認)、提供パッケージ数:553
  - 2011-11にver. 2.9をリリ
  - 2011-04にver. 2.8をリリ

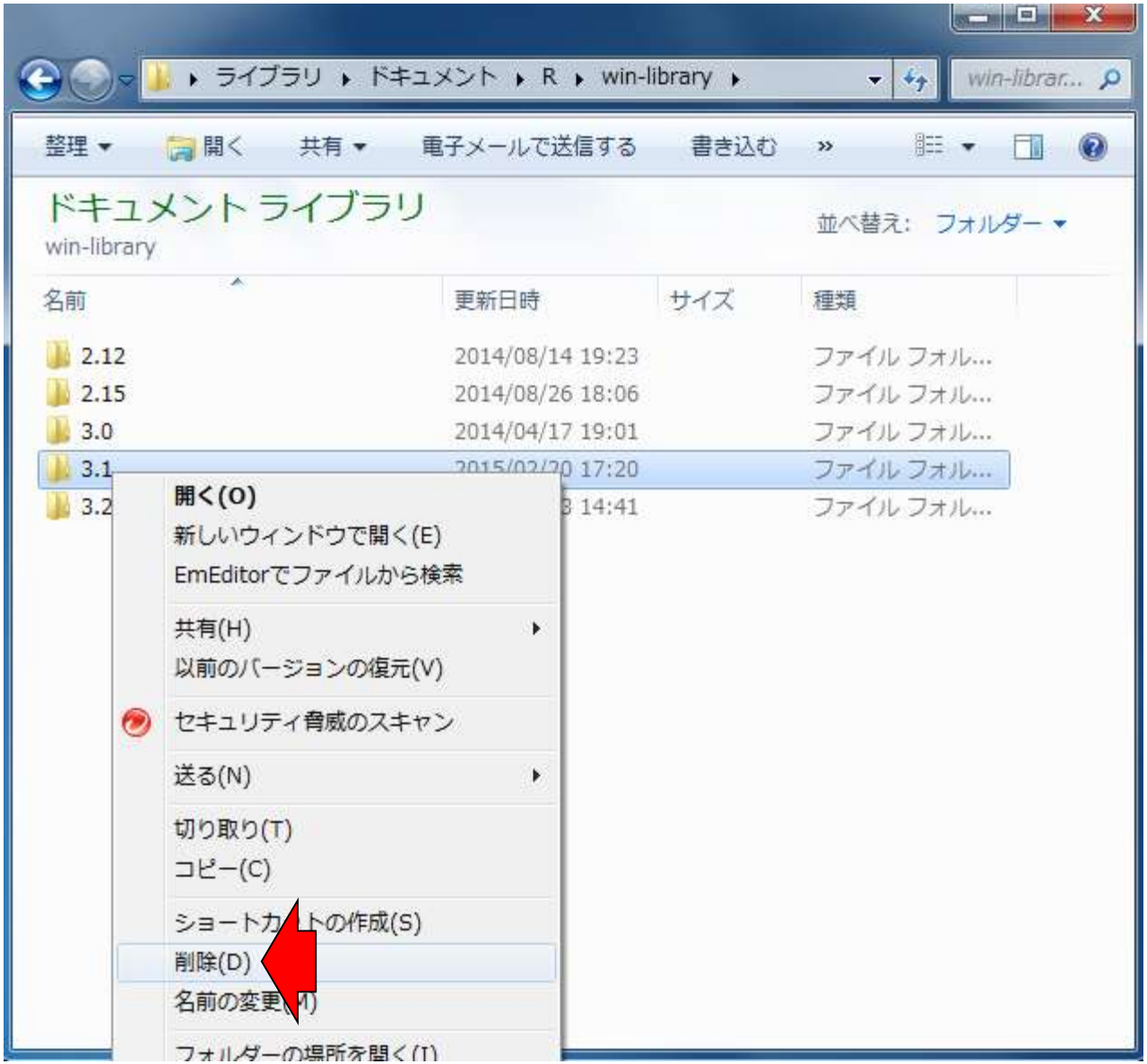
門田のPCは、R ver. 3.1.2 (Bioconductor ver. 3.0)インストール後に、R ver. 3.1.0 (Bioconductor ver. 2.14)をインストールしている。これらはともにR ver. 3.1.Xなので、同じフォルダ(…/win-library/3.1)にBioconductor ver. 2.14由来パッケージが上書きインストールされた結果である。複数のバージョンのR本体をインストールする場合にはインストール先のフォルダ名にも気を付けよう。



参考

通常の門田のやり方

アップグレード系は「すでにパッケージがあるけど上書きするか？」などいろいろ聞かれるのが面倒なので、一旦全部消します。そして最初から全部のインストールをやり直します。



Contents1

- イントロダクション(教材最新情報)
  - (Rで)塩基配列解析、アグリバイオインフォマティクス教育研究プログラム
  - バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)
  - 講習会PC環境
- ゲノム解析
  - 塩基配列解析基礎
    - multi-FASTA形式の塩基配列ファイルを読み込んで自在に解析する(Biostrings)
  - パッケージ(CRANとBioconductor)
  - Bioconductor概観 → ゲノム配列パッケージ(BSgenome)
  - 2連続塩基出現頻度解析(CpG解析)、k-mer解析
  - アノテーション(TxDb, GenomicFeatures)
  - 個別パッケージのインストール
  - プロモーター配列取得

プロモーター配列取得

ゲノム配列とアノテーション情報があれば、任意の領域の配列を取り出すことができます。

イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenomeとTxDbから

NEW

ゲノム配列(BSgenome)パッケージとアノテーション情報(TxDb)パッケージを用いて様々な生物種のプロモーター配列(転写開始点近傍配列;上流配列)を取得するやり方を示します。2014年4月リリースのBioconductor 2.14以降の推奨手順では、ゲノムのパッケージ(例:BSgenome.Hsapiens.UCSC.hg19)と対応するアノテーションパッケージ(例:TxDb.Hsapiens.UCSC.hg19.knownGene)の両方を読み込ませる必要がありますので、2015年2月に記述内容を大幅に変更しました。ヒトなどの主要なパッケージ以外はおそらくデフォルトではインストールされていませんので、「パッケージがインストールされていません」的なエラーが出た場合は、個別パッケージのインストールを参考にして予め利用したいパッケージのインストールを行ってから再挑戦してください。出力はmulti-FASTAファイルです。現状では、ゼブラフィッシュ(danRer7)はゲノムパッケージ(BSgenome.Drerio.UCSC.danRer7)は存在しますが、対応するTxDbパッケージが存在しないので、どこかからGFFファイルを取得してmakeTranscriptDbFromGFF関数などを利用してTranscriptDbオブジェクトを得るなどする必要があります。

プロモーター配列取得

作業ディレクトリはどこでもよいが基本はデスクトップ上のhoge。転写開始点上流1000塩基を取得したい場合。width列は配列長に相当。出力ファイルを見なくてもうまくいっているだろうと思える。

1. ヒト(hg19)の場合:

ゲノムパッケージ(BSgenome.Hsapiens.UCSC.hg19)と対応するアノテーションパッケージ(TxDb.Hsapiens.UCSC.hg19.knownGene)を読み込んで、転写開始点上流1000塩基分を取得するやり方です。

```
out_f <- "hoge1.fasta"                          #出力ファイル名を指定してout_fに格納
param_bsgenome <- "BSgenome.Hsapiens.UCSC.hg19"#パッケージ名を指定(BSgenome系のゲノムパ
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーシ
param_upstream <- 1000                          #転写開始点上流の塩基配列数を指定

#前処理(指定したパッケージ中のオブジェクト
library(param_bsgenome, character.only
tmp <- ls(paste("package", param_bsger
genome <- eval(parse(text=tmp))

library(param_txdb, character.only=T)
tmp <- ls(paste("package", param_txdb,
txdb <- eval(parse(text=tmp))

#本番
gn <- sort(genes(txdb))
hoge <- flank(gn, width=param_upstream
fasta <- getSeq(genome, hoge)
fasta

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

R Console

```
> fasta <- getSeq(genome, hoge)                 #指定した範囲の$
> fasta                                         #確認してるだけ$
  A DNAStringSet instance of length 23056
        width seq                                   names
    [1]  1000 ACACATGCTA...TTTCGTTAA 100287102
    [2]  1000 GCTATTATCA...AATAACTCT 79501
    [3]  1000 GAATTAGGCT...AAGGCGGGG 643837
    [4]  1000 CGGGGAGCCC...CTTGGGCCA 148398
    [5]  1000 CGGCGGGGCT...AGCGGCGGG 339451
     ...   ... ...
[23052]  1000 GGTGAGCCAA...ATCTCAGCC 283788
[23053]  1000 AGCCCTCCAC...CTCTCCAAC 100507412
[23054]  1000 CGGGGCCCAG...CCTGGCTGC 728410
[23055]  1000 CGGGGCCCAG...CCTGGCTGC 100653046
[23056]  1000 CAGGCTGAGC...CTCACCGCG 100288687
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", wid$
>
```

プロモーター配列取得

線虫はBSgenomeとTxDbの両方が提供されています。予め2つのパッケージを個別にインストールしておけば、上流500～下流20塩基の範囲の配列を取得できる。

3. 線虫(ce6)の場合:

ゲノムパッケージ(BSgenome.Celegans.UCSC.ce6)と 対応するアノテーションパッケージ(TxDb.Celegans.UCSC.ce6.ensGene)を読み込んで、転写開始点上流500塩基から下流20塩基までの範囲を取得するやり方です。

```
out_f <- "hoge3.fasta"                    #出力ファイル名を指定してout_fに格納
param_bsgenome <- "BSgenome.Celegans.UCSC.ce6"#パッケージ名を指定(BSgenome系のゲノムパッ
param_txdb <- "TxDb.Celegans.UCSC.ce6.ensGene"#パッケージ名を指定(TxDB系のアノテーション
param_upstream <- 500
param_downstream <- 20

#前処理(指定したパッケージ中のオブジェクト
library(param_bsgenome, character.only
tmp <- ls(paste("package", param_bsger
genome <- eval(parse(text=tmp))

library(param_txdb, character.only=T)
tmp <- ls(paste("package", param_txdb,
txdb <- eval(parse(text=tmp))

#本番
gn <- sort(genes(txdb))
hoge <- promoters(gn, upstream=param_u
          downstream=param_downstream
fasta <- getSeq(genome, hoge)
fasta
```

R Console

```
> fasta <- getSeq(genome, hoge)            #指定した範囲の$
> fasta                                    #確認してるだけ$
  A DNAStringSet instance of length 27928
        width seq                    names
    [1]   520 TTGCGCGTAA...CCGCGTCAC Y74C9A.2
    [2]   520 GCAGATAATT...ACGATGGAA Y74C9A.1
    [3]   520 AGAGGAATTT...CGAGACAAA Y48G1C.12
    [4]   520 TTACCTCCAG...CCCGATCCC Y48G1C.4
    [5]   520 CTTCATTCCC...GCAGCAGCA Y48G1C.2
    ...   ... ...
[27924]   520 GGGTGTTACA...ATAAAAAAG MTCE.29
[27925]   520 ATATCTGCAG...ATAATTCAG MTCE.30
[27926]   520 GTAGTATTTT...TTATATTAA MTCE.32
[27927]   520 TTTATATTAT...TGTTTAAAT MTCE.33
[27928]   520 GATTAATATT...TTAATAAAA MTCE.36
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", wid$
>
```

プロモーター配列取得

multi-FASTA形式のゲノム配列ファイルとGFF3形式などの一般的なアノテーションファイルがあれば、それを利用することももちろんできます。入力ファイルは「デスクトップ – hoge – Lcasei」中にあります。

- イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して保存 (last modified 2013/09/26)
- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから (last modified 2014/05/28)
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome (last modified 2015/02/19) NEW
- イントロ | 一般 | 配列取得 | プロモーター配列 | 公共DBから (last modified 2014/04/02)
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenomeとTxDbから (last modified 2015/02/20) NEW
- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence_2013) (last modified 2015/02/20)
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | 公共DBから (last modified 2014/04/02)

イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence_2013) NEW

GenomicFeaturesパッケージを主に用いてプロモーター配列(転写開始点近傍配列)を得るやり方を示します。
ここでは、イントロ | 一般 | 配列取得 | ゲノム配列(BSgenomeから)で指定可能なゲノムとイントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatures(Lawrence_2013)で作成可能なTranscriptDbオブジェクトを入力として、getPromoterSeqという関数を用いて、転写開始点から任意の[上流xxx塩基, 下流yyy塩基]分の塩基配列を取得して、FASTA形式ファイルで保存するやり方を示しています。multi-FASTA形式のゲノム配列ファイルとGFF3形式のアノテーションファイルのみからプロモータ配列を取得するやり方も示しています。

7. 乳酸菌ゲノム(Lactobacillus casei 12A)の[上流500塩基, 下流20塩基]のプロモーター配列を取得する場合:

Ensembl (Flicek et al., Nucleic Acids Res., 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.chromosome.Chromosome.gff3)と GFF3形式のアノテーションファイル(Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa) を読み込むやり方です。
makeTranscriptDbFromGFF関数実行時に用いているuseGenesAsTranscripts=Tは、原核生物(prokaryotes)に代表される「転写物＝遺伝子」の場合に指定します。デフォルトはFです。

```
in_f1 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入力ファイル名を指
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"#入力ファイル名を指定
out_f <- "hoge7.fasta"                    #出力ファイル名を指定してout_fに格納
param_upstream <- 500                     #転写開始点上流の塩基配列数を指定
param_downstream <- 20                    #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(Biostrings)                       #パッケージの読み込み
library(GenomicFeatures)                  #パッケージの読み込み
```

プロモーター配列取得

まだちゃんと検証はできてはいませんが、見た感じうまくとれている印象をうけます。

7. 乳酸菌ゲノム(Lactobacillus casei 12A)の[上流500塩基, 下流20塩基]のプロモーター配列を取得する場合:

Ensembl (Flicek et al., Nucleic Acids Res., 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.chromosome.Chromosome.gff3)と GFF3形式のアノテーションファイル (Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa) を読み込むやり方です。
makeTranscriptDbFromGFF関数実行時に用いているuseGenesAsTranscripts=Tは、原核生物(prokaryotes)に代表される「転写物＝遺伝子」の場合に指定します。デフォル

```
in_f1 <- "Lactobacillus_case
in_f2 <- "Lactobacillus_case
out_f <- "hoge7.fasta"
param_upstream <- 500
param_downstream <- 20

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

#前処理(アノテーション情報を取
txdb <- makeTranscriptDbFrom
          format="gff3", us
txdb

#前処理(欲しい領域の座標情報取
gn <- sort(genes(txdb))
hoge <- promoters(gn, upstre
          downstream=param_
hoge
obj <- (ranges(hoge)@start >
```

R Console

```
> #本番(配列取得)
> fasta <- getSeq(FaFile(in_f1), hoge)    #指定した範囲の塩基配列情$
> names(fasta) <- names(hoge)             #description情報を追加して$
> fasta                                   #確認してるだけです
  A DNAStringSet instance of length 2680
        width seq                                         names
    [1]   520 CTTGAAAGCCTTGAAA...ATTTACGATCACCCG gene:LCA12A_0618
    [2]   520 GAACTGATAATGTGCG...AACAATCGAAATCAC gene:LCA12A_0619
    [3]   520 GTTTCACGATCAATCG...ACTGGATCACTTGGT gene:LCA12A_0620
    [4]   520 CCACAGCATTGGCTGT...GGACAAGAAAGAAAC gene:LCA12A_0621
    [5]   520 ACTGATCATTATGACC...TGATCGCCAAGAAAG gene:LCA12A_0622
    ...   ... ...
[2676]   520 AAGTGACCGTTGCTTA...TGCATCTATCACTGC gene:LCA12A_0612
[2677]   520 ACACAGAAACTTTATG...AACCTTTCAAGGCAG gene:LCA12A_0613
[2678]   520 TGCGCAAGTCATATCG...AAATCGTCAAATCAA gene:LCA12A_0614
[2679]   520 CGAATTTCCTGTGATT...TGCGCGTTTCGGCGG gene:LCA12A_0615
[2680]   520 GCGCCAATTCACGCTC...AACCAAACGGACTTT gene:LCA12A_0616
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fast$
>
```

プロモーター配列取得

コピペ前後の様子。コピペ後にhoge7.fastaという出力ファイルが得られていることがわかる。

コピペ前

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

コピペ後

```
R Console
[2678]    520 TGCGCAAGTCATATCGGATCAA...TTGTGAAAAATCGTCAAATCAA gene:LCA12A_0614
[2679]    520 CGAATTTCCTGTGATTGGTTTG...GCTTGTTTGCGCGTTTCGGCGG gene:LCA12A_0615
[2680]    520 GCGCCAATTCACGCTCTGTAAA...CAATGACAACCAAACGGACTTT gene:LCA12A_0616
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/Lcasei"
> list.files()
[1] "hoge7.fasta"
[2] "Lactobacillus_casei_12a.GCA_000309565.2.25.chromosome.Chromosome.gff3"
[3] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"
[4] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.fai"
>
```

Contents2

「イントロ、公共DB、データ取得」周辺情報は、下記のR本、2014年6月18日のアグリバイオ大学院講義資料、乳酸菌学会誌の連載第3回などにも情報あり。

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

↑教科書(参考書)

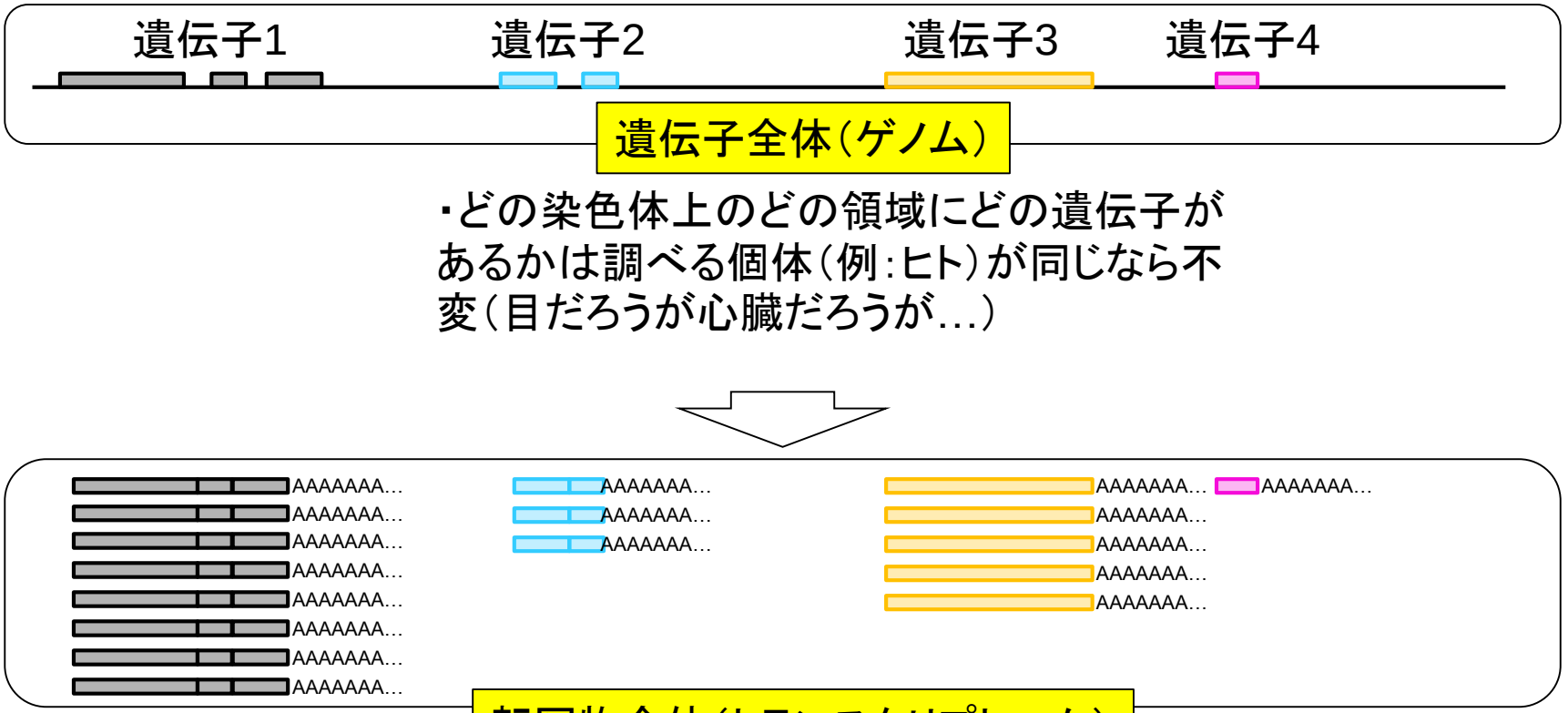
参考 調べたい組織でどの遺伝子がどの程度発現しているのかを一度に観察

イントロダクション

- トランスクリプトームとは
  - ある特定の状態の組織や細胞中に存在する全RNA（転写物、transcripts）の総体
- 様々なトランスクリプトーム解析技術
  - マイクロアレイ
    - DNAマイクロアレイ、Affymetrix GeneChip、タイリングアレイなど
  - 配列決定に基づく方法
    - EST、SAGE、CAGE、次世代シーケンサ(RNA-seq)など
  - 電気泳動に基づく方法
    - Differential Display、AFLP、HiCEPなど

トランスクリプトームとは

- ある状態のあるサンプル（例：目）のあるゲノムの領域

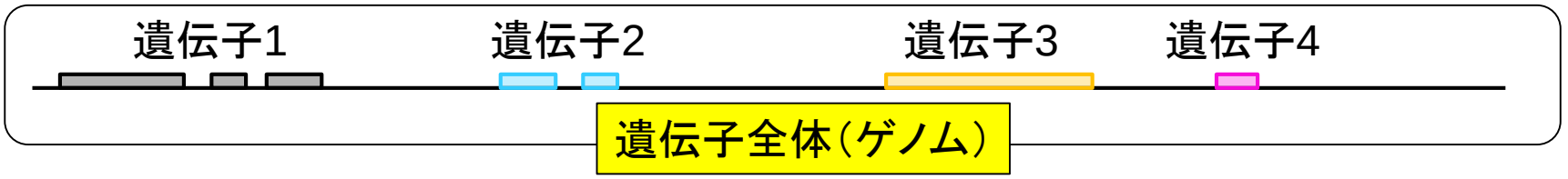


・どの染色体上のどの領域にどの遺伝子があるかは調べる個体（例：ヒト）が同じなら不変（目だろうが心臓だろうが…）

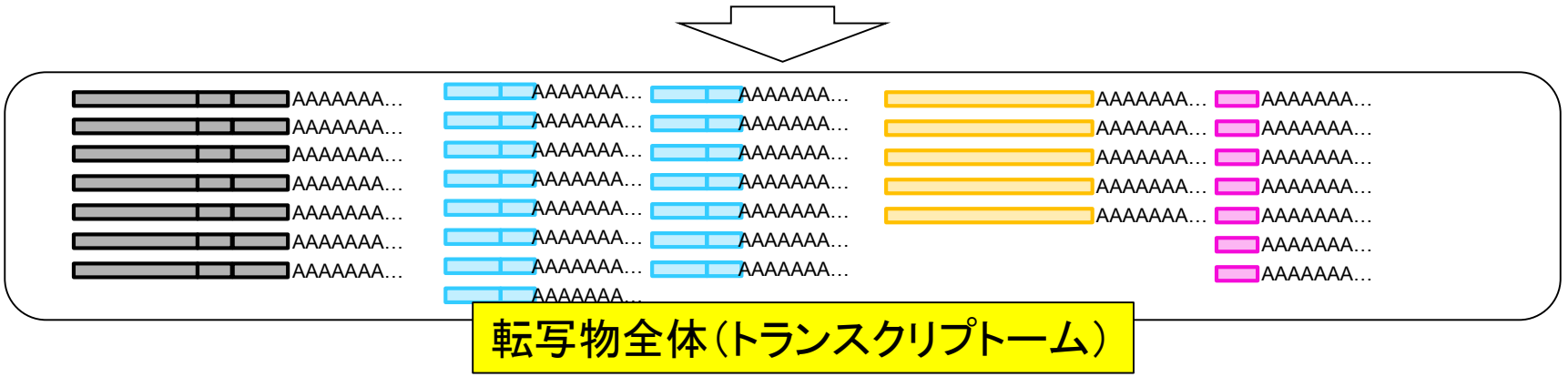
・遺伝子1は沢山転写されている（発現している）
・遺伝子4はごくわずかしか転写されてない
・…

トランスクリプトームとは

- 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…）



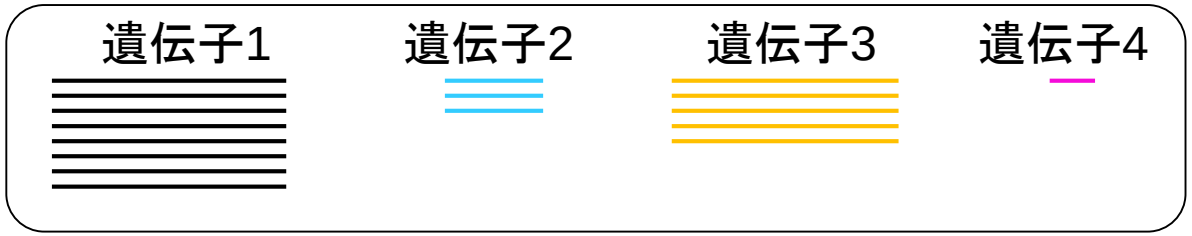
・遺伝子2は光刺激に応答して発現亢進
・遺伝子4も光刺激に応答して発現亢進

参考

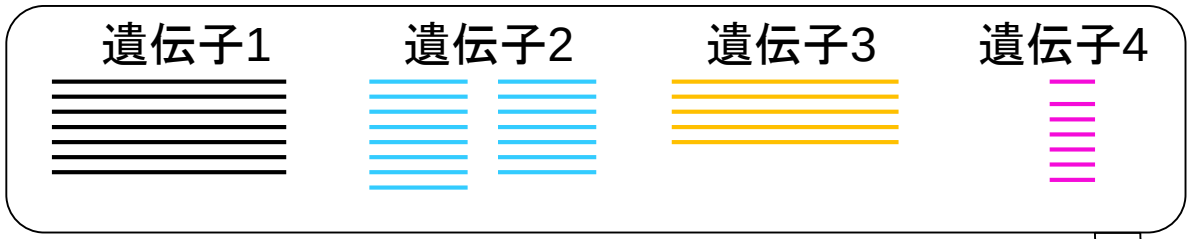
教科書p9の図1-8に示してあるように、実際には「遺伝子 = 転写物」とは限らない点に注意!

トランスクリプトーム取得手段

- 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」

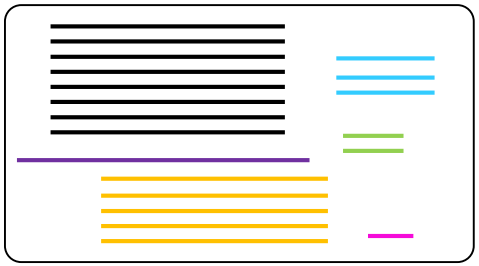
	T1	T2
遺伝子1	8	7
遺伝子2	3	15
遺伝子3	5	5
遺伝子4	1	7
…	…	…

・マイクロアレイ
・RNA-seq

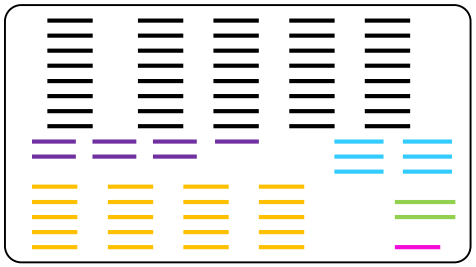
トランスクリプトーム取得（NGS）

- 次世代シーケンサー（Illumina社の場合）

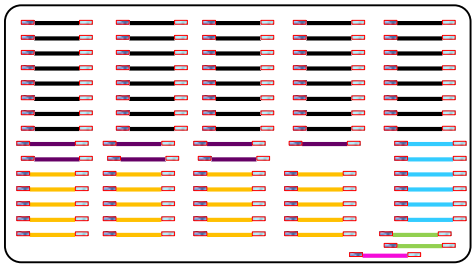
光刺激前（T1）の目のトランスクリプトーム



数百塩基程度に断片化



アダプター配列を両末端に付加



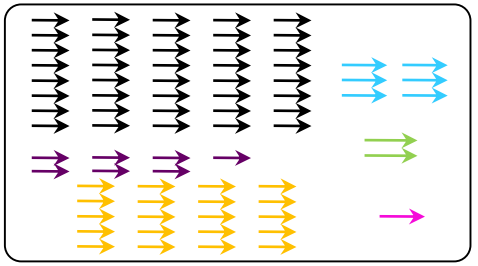
配列決定

・ペアードエンド法

断片配列の両末端が数百塩基以内の対の2種類の配列が得られる

・シングルエンド法

シングルエンド法の場合



参考

様々なNGSプラットフォーム

- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence 2013)(last modified 2015/02/20) NEW
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | 公共DBから(last modified 2014/04/02)
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | biomaRt(Durinck 2009)(last modified 2015/02/20) NEW
- イントロ | NGS | 様々なプラットフォーム (last modified 2014/06/10)
- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2014/11/12)
- イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25)
- イントロ | NGS | 配列
- イントロ | NGS | 配列
- イントロ | NGS | 配列
- イントロ | NGS | 配列

イントロ | NGS | 様々なプラットフォーム

NGS機器(プラットフォーム)もいくつかあります:

- 会社名:製品名
- Illumina: MiSeq, NextSeq 500, HiSeq 2500, HiSeq X Ten, ...
- Roche: GS FLX+ System, GS Junior+ System
- Life Technologies: SOLiD
- Life Technologies: Ion PGM System
- Pacific Biosciences: PacBio RS II System
- Dover社など「POLONATOR G.007」
- ...

Pacific Biosciences (PacBio)について:

PacBio RS II Systemは最長で20,000bp以上(平均は4,500 bp)読めるようですが配列のqualityが若干(85%程度)劣るようです。しかしエラーの入る場所がランダムなようで多数決ルール(majority rule)でエラー補正がかなりうまくいくらしいです。このロングリードでトランスクリプトーム配列決定(新規アイソフォームの発見)をヒト(Sharon et al., 2013)やニワトリ(Thomas et al., 2014)で行った論文などが出始めています。Smart-seq2の実験手順(Picelli et al., Nat Protoc., 2014)なども出ているようですね。葉緑体ゲノムでのアセンブリ性能評価(Ferrarini et al., BMC Genomics, 2013)もなされています。

NGS機器同士の比較:

いくつか性能評価論文も出ているようです。

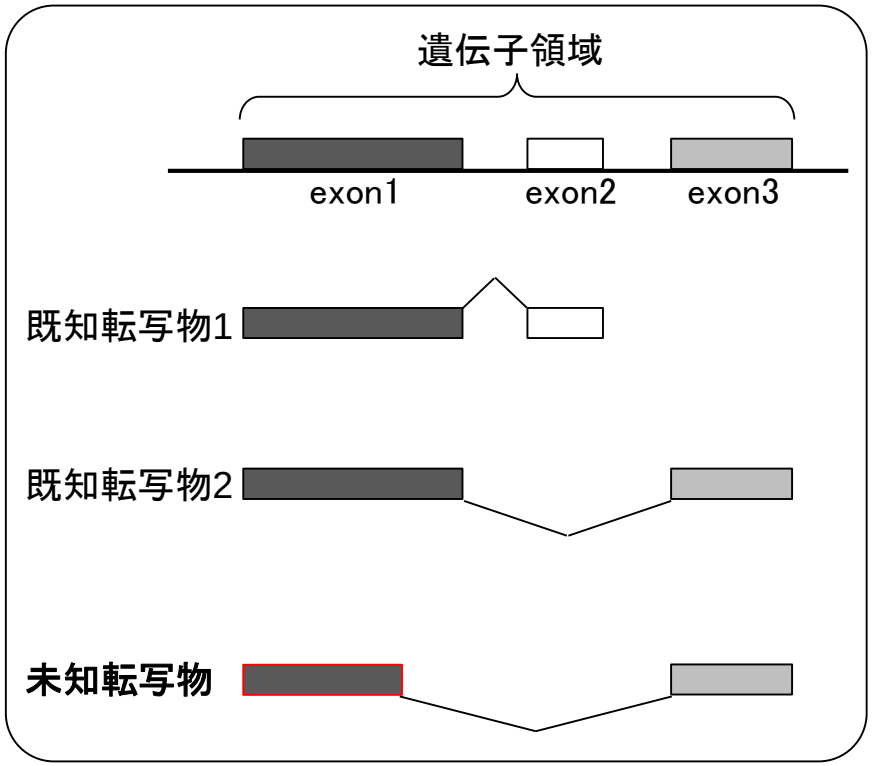
Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

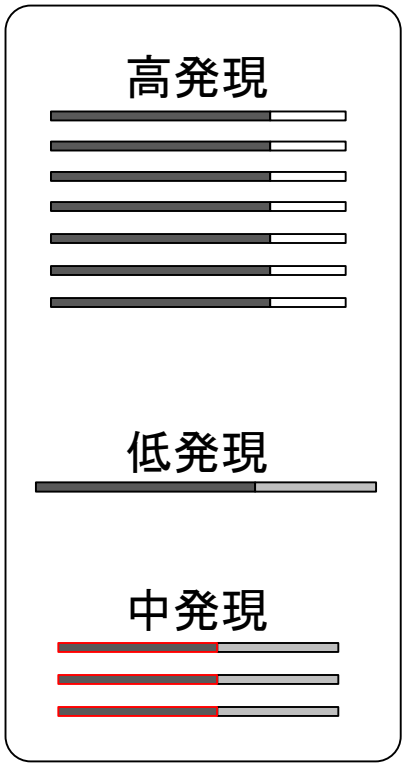
基本イメージ

どの転写物由来か分からない塩基配列情報のみがRNA-seqによって得られる。これをもとに真の転写物の配列や発現情報を得るのがRNA-seq解析の目的。

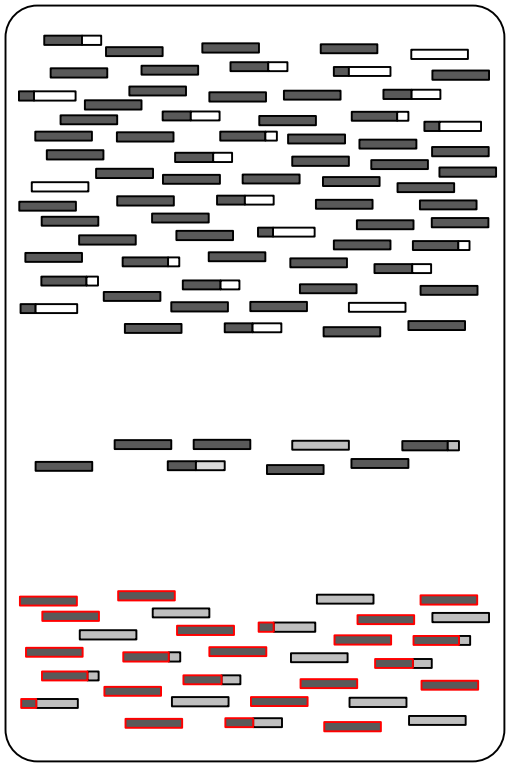
- 真の転写物情報：ある遺伝子領域中に既知転写物は2つ、**未知転写物**も1つ!
- 真の発現情報：既知転写物1（高発現）、既知転写物2（低発現）、未知転写物（中発現）



真の転写物情報



真の発現情報

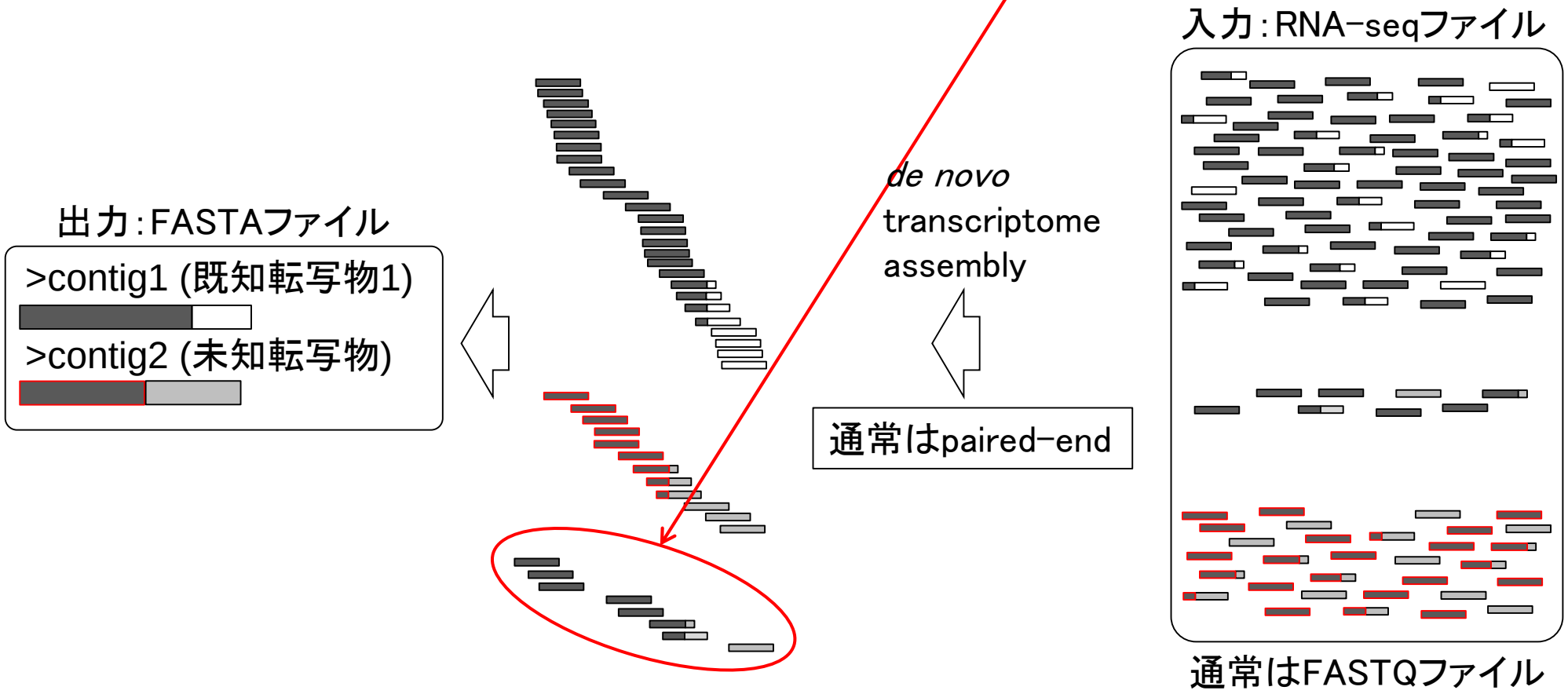


RNA-seqで得られるリード情報
（色は不明；single-endの場合）

基本イメージ

ターゲットサンプル中でそれほど発現していない転写物はアセンブルが原理的に困難。

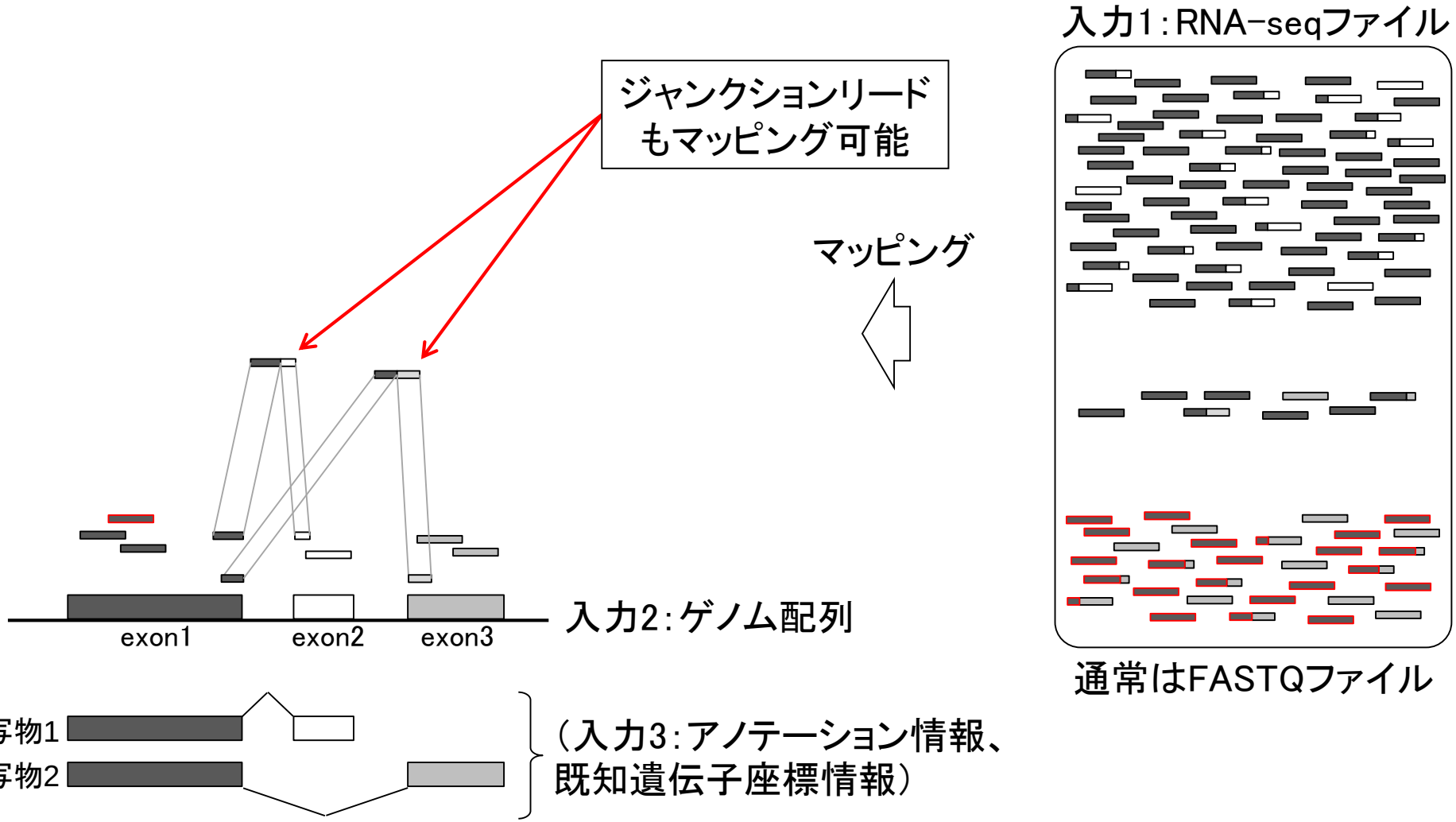
- RNA-seqデータのみしか手元にない場合：トランスクリプトーム配列取得



基本イメージ

リファレンス配列を利用することで
低発現転写物の遺伝子構造推定が
de novo assemblyに比べて容易に

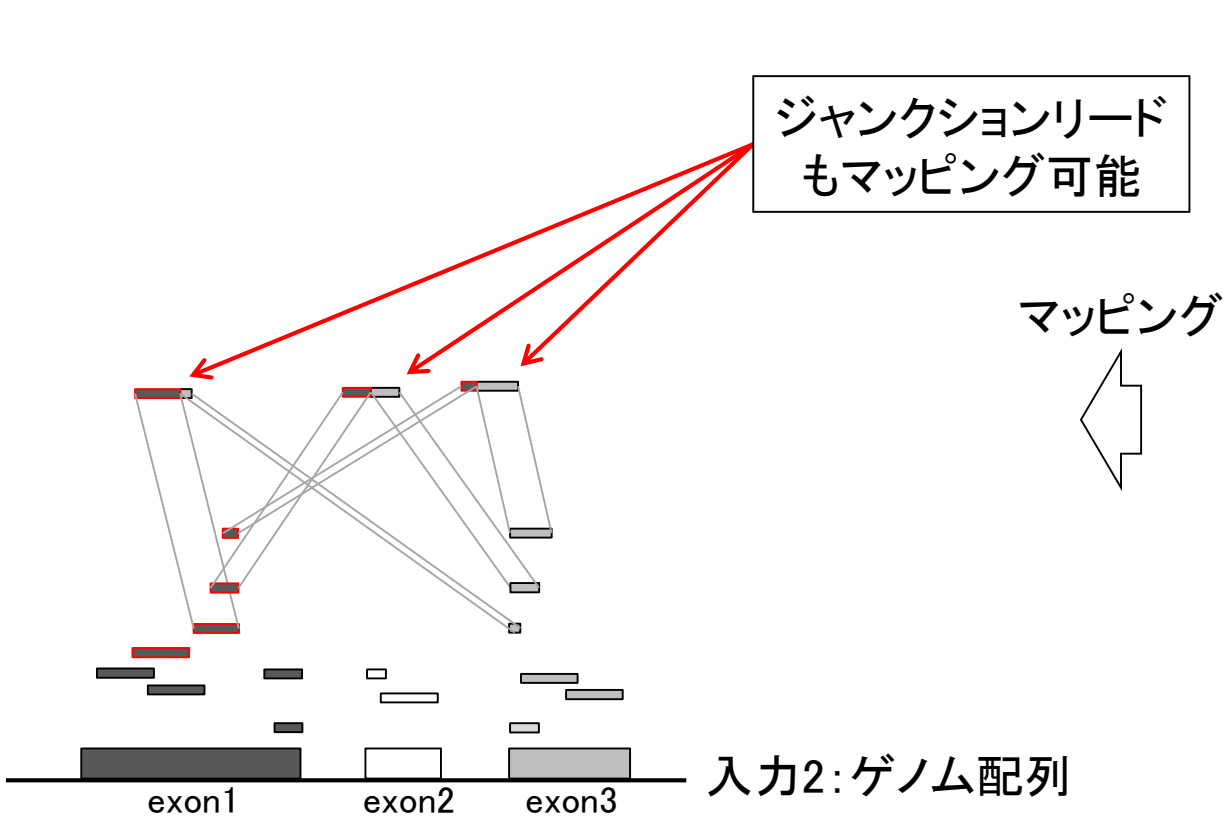
- リファレンスとしてゲノム配列が利用可能な場合：新規転写物の同定

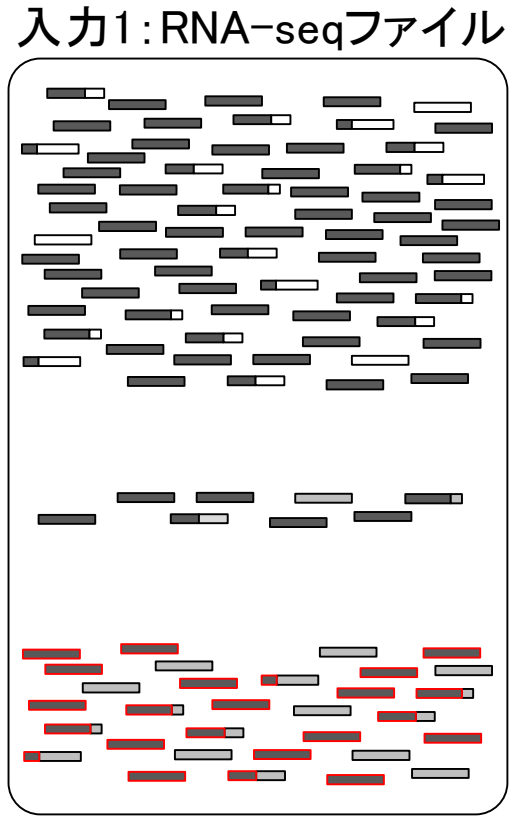


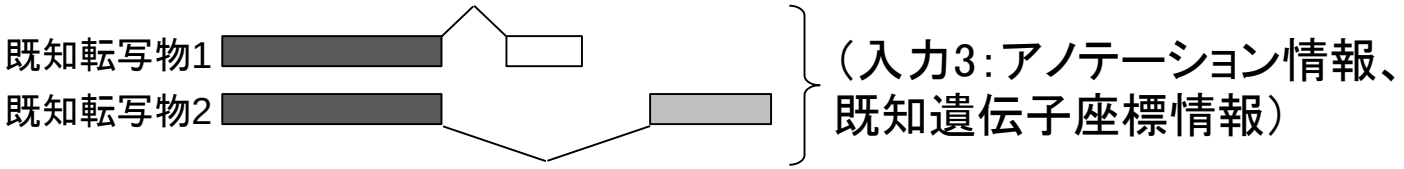
基本イメージ

リファレンス配列を利用することで
低発現転写物の遺伝子構造推定が
de novo assemblyに比べて容易に

- リファレンスとしてゲノム配列が利用可能な場合：新規転写物の同定



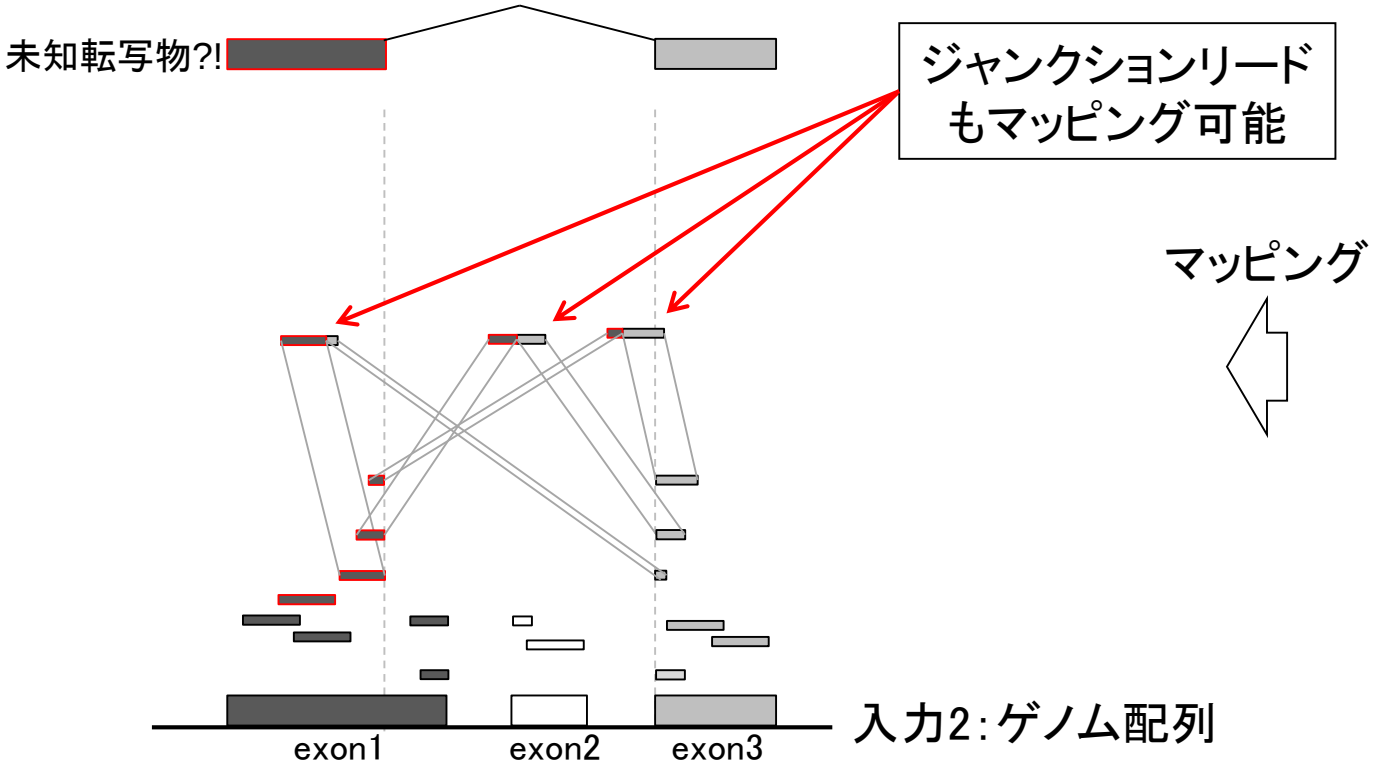


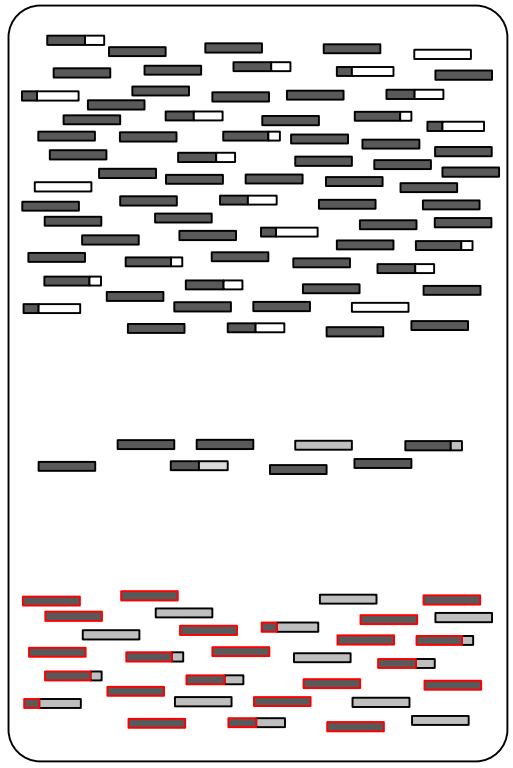


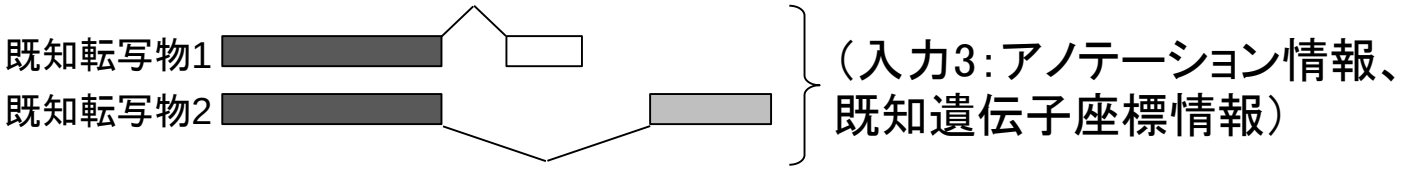
基本イメージ

未知転写物同定が醍醐味。全般的な最新情報は2014年7月22日のイルミナウェビナー資料を参照のこと。

- リファレンスとしてゲノム配列が利用可能な場合：新規転写物の同定







様々な解析目的

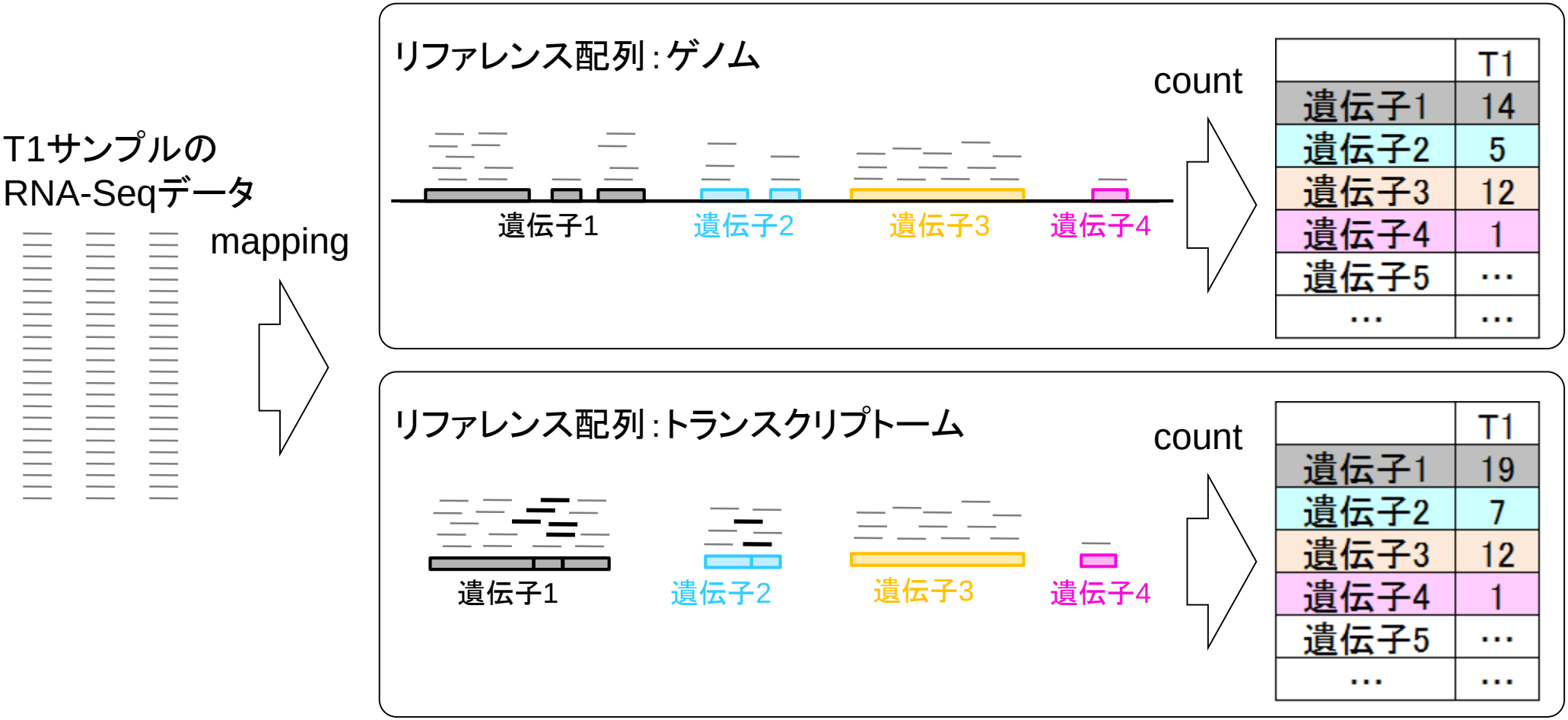
各種補正に関する詳細情報は、R本、2014年7月2日のアグリバイオ大学院講義資料などを参照

- トランスクリプトーム配列取得
  - ゲノム配列既知の場合：Cufflinksなどを用いて遺伝子構造推定（アノテーション）
  - ゲノム配列未知の場合：Trinityなどのトランスクリプトーム用アセンブラを実行
- 遺伝子または転写物(isoform)ごとの発現量の正確な推定
  - RSEMなどを利用して発現量情報を得る
  - ある特定のサンプル内での遺伝子間の発現量の大小関係を知りたい
  - 配列長やGC biasなどの各種補正がポイント
- 比較するサンプル間で発現変動している遺伝子または転写物の同定
  - TCCパッケージなどを利用して発現変動遺伝子（DEG）を得る
  - ライブラリサイズ（総リード数）や発現している遺伝子の組成の補正がポイント
  - （GO解析など）DEG結果を用いる多くの下流解析結果に影響を及ぼす

マッピングの基本イメージ

「マップされたリード数 = 発現量」ではないが、マップされたリード数のカウント情報は、発現量推定の基本情報です

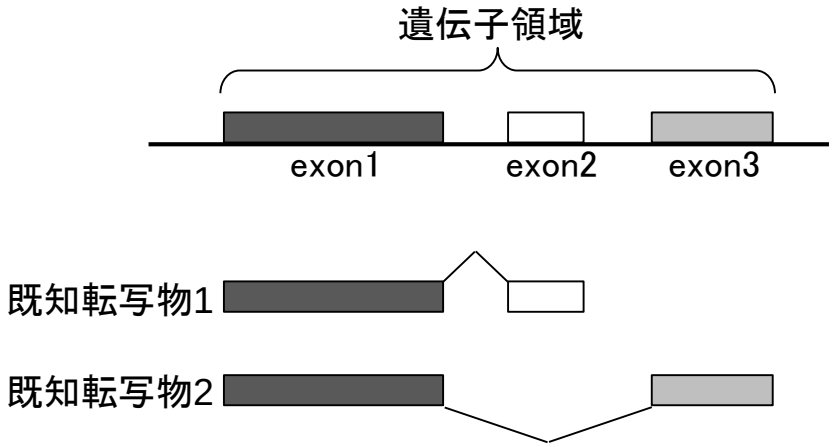
- 基本的なマッピングプログラム（bowtieなど）を用いた場合

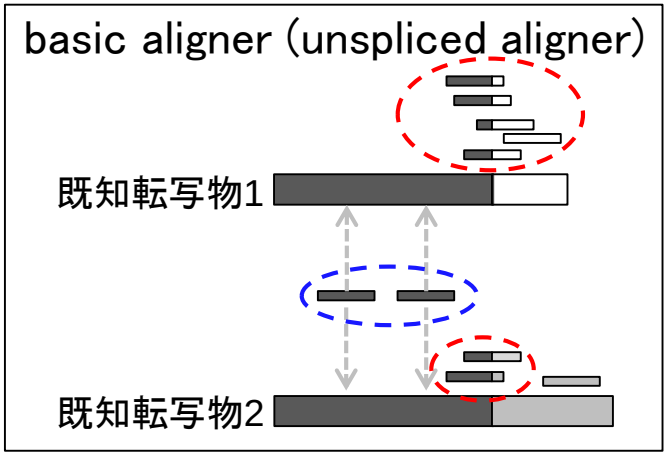


実習では…

- Neyret-Kahn et al., *Genome Res.*, **23**: 1563-1579, 2013のヒトRNA-seqデータのFASTQファイルを取得
  - 原著論文中の記述はGSE42213のみ。これはChIP-seqデータ(GSE42211)とRNA-seqデータ(GSE42212; SRP017142)をまとめたID
- ヒトゲノムにbasic alignerでマップ(QuasR, Rbowtie)
  - アノテーション情報を利用して遺伝子レベル、exonレベルのカウントデータを取得
- データ解析:クラスタリング、発現変動遺伝子(DEG)検出、機能解析

Basic alignerの場合は、ジャンクションリードはマップされない。遺伝子レベル、exonレベルのカウントデータ取得の場合は、shared exon云々の話とは基本無関係。





Rのみで一通り解析が可能に

アセンブル以外ならWindows（のR）上でどうにかなる時代

- SRAdb (Zhuら, *BMC Bioinformatics*, **14**: 19, 2013)
  - 公共DBからのRNA-seqデータ（FASTQファイル）取得
- QuasR (Gaidatzisら, *Bioinformatics*, 2014)
  - リファレンス配列（ゲノム or トランスクリプトーム）へのマッピング
    - Bowtie (Langmeadら, 2009) or SpliceMap (Auら, 2010)を選択可能
    - 出力はBAM形式ファイル、QCレポートも
  - 遺伝子アノテーション情報をもとにカウントデータ取得
    - GenomicFeatures (Lawrenceら, 2013)で得られるTranscriptDbオブジェクトを利用
    - UCSC known genesやEnsembl genesのカウントデータなど
- TCC (Sunら, *BMC Bioinformatics*, **14**: 219, 2013)
  - 内部的にedgeR (Robinsonら, 2010)やDESeq (Anders, 2010)などを用いて頑健な発現変動解析を実行。DESeq2 (Loveら, 2014)にも対応済み。

Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

NGSデータ取得

SRA, DRA, ENAが公式?!なNGS data repositoryかもしれないが、私は論文中のIDでそのままググります。

- 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) NEW
- イントロ | NGS | 様々なプラットフォーム (last modified 2014/06/10)
- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2014/11/12)
- イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25)
- イントロ | NGS | 配列取得 | FASTQ or SRA | 公共DBから (last modified 2015/02/23) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRA | SRAdb(Zhu 2013) (last modified 2014/06/26)
- イントロ | NGS | 配列取得 | シミュ
- イントロ | NGS | 配列取得 | シミュ
- イントロ | NGS | アノテーション情
- イントロ | NGS | アノテーション情
- イントロ | NGS | アノテーション情
- イントロ | NGS | アノテーション情
- イントロ | NGS | アノテーション情
- イントロ | NGS | アノテーション情

イントロ | NGS | 配列取得 | FASTQ or SRA | 公共DBから NEW

次世代シーケンサ(NGS)から得られる塩基配列データを公共データベースから取得する際には以下を利用します。マイクロアレイデータ取得のときと同様、NGSデータもArrayExpress経由でダウンロードするのがいいかもしれません。メタデータの全貌を把握しやすいこと、生データ(raw data)だけでなく加工済みのデータ(processed data)がある場合にはその存在がすぐにわかることなど、操作性の点で他を凌駕していると思います。上記でも触れているようにFASTQファイルのダウンロードからマッピングまでを行うのはエンドユーザレベルでは大変ですが、submitterが提供してくれている場合は(まだまだ少ないようですが)リファレンス配列へのマップ後のデータ、つまりBAM形式ファイルの提供もすでに始まっているようです。2014年6月26日に知りました(DDBJ児玉さんありがとうございますm(_ _)m)。

データの形式は基本的にSanger typeのFASTQ形式です。FASTA形式はリードあたり二行(idの行と配列の行)で表現します。FASTQ形式はリードあたり4行(@から始まるidの行と配列の行、および+から始まるidの行とbase callの際のqualityの行)で表現します。FASTQ形式は、Sangerのものがデファクトスタンダード(業界標準)です。かつてIlluminaのプラットフォームから得られるのはFASTQ-like formatという表現がなされていたようです(Cock et al., Nucleic Acids Res., 2010)。しかし少なくとも2013年頃には、IlluminaデータもBaseSpaceやCASAVA1.8のconfigureBclToFastq.plなどを用いることで業界標準のFASTQ形式(つまりSanger typeのデータ)に切り替えられるようですし、NCBI SRAなどの公共DBから取得するデータは全てはSanger typeのデータになっていたと思います(Kibukawa E., テクニカルサポートウェビナー, 2013)。

- DDBJ Sequence Read Archive (DRA): Kodama et al., Nucleic Acids Res., 2012
- European Nucleotide Archive (ENA): Silvester et al., Nucleic Acids Res., 2015
- NCBI Sequence Read Archive (SRA): NCBI Resource Coordinators., Nucleic Acids Res., 2014
- ArrayExpress: Kolesnikov et al., Nucleic Acids Res., 2015
- DBCLS SRA: Nakazato et al., PLoS One, 2013

NGSデータ取得

原著論文中の記述はGSE42213のみ。重要なのは全体像の理解です。

- 「Neyret-Kahn et al., *Genome Res.*, **23**: 1563-1579, 2013」のRNA-seqデータのFASTQファイルを取得

NGSデータ取得

原著論文中の記述はGSE42213のみ。重要なのは全体像の理解です。これはChIP-seq(GSE42211)とRNA-seq(GSE42212)をまとめたID。

NCBI

Gene Expression Omnibus

HOME | SEARCH | SITE MAP | GEO Publications | FAQ | MIAME | Email GEO

NCBI > GEO > **Accession Display**

Not logged in | Login

Scope: Self | Format: HTML | Amount: Quick | GEO accession: GSE42213 | GO

Series GSE42213 Query DataSets for GSE42213

Status	Public on Jul 24, 2013
Title	SUMO is an Integral and Instructive Compone and Senescence
Organism	Homo sapiens
Experiment type	Genome binding/occupancy profiling by high th Expression profiling by high throughput sequen
Summary	This SuperSeries is composed of the SubSeries
Overall design	Refer to individual Series
Citation(s)	Neyret-Kahn H, Benhamed M, Ye T, Le Gras S e governs coordinated repression of a transcriptio growth and proliferation. *Genome Res* 2013 Oc PMID: 23893515
Submission date	Nov 09, 2012
Last update date	Feb 17, 2015
Contact name	Tao YE
Organization name	IGBMC (CNRS/INSERM/UDS)
Street address	1 rue Laurent Fries
City	Illkirch
ZIP/Postal code	67404
Country	France

Platforms (2)	GPL10999	Illumina Genome Analyzer IIx (Homo sapiens)
	GPL11154	Illumina HiSeq 2000 (Homo sapiens)
Samples (26)	GSM1035423	prolif_input_DNA
More...	GSM1035424	prolif_SUMO1_rep1_ChIPSeq
	GSM1035425	prolif_SUMO1_rep2_ChIPSeq

This SuperSeries is composed of the following SubSeries:

GSE42211	Genome wide occupancy of SUMO machinery in proliferative and Ras-induced senescent human primary fibroblasts
GSE42212	Quantitative analysis of proliferative and Ras-induced senescent human primary fibroblasts transcriptomes

Relations

BioProject PRJNA179295

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE42213_RAW.tar	2.8 Gb	(http)(custom)	TAR (of BED, WIG)

NGSデータ取得

Samples (26)
Less...

GSM1035423	prolif_input_DNA
GSM1035424	prolif_SUMO1_rep1_ChIPSeq
GSM1035425	prolif_SUMO1_rep2_ChIPSeq
GSM1035426	prolif_SUMO2_ChIPSeq
GSM1035427	prolif_Ubc9_ChIPSeq
GSM1035428	prolif_PolII_ChIPSeq
GSM1035429	prolif_H3K4me3_ChIPSeq
GSM1035430	prolif_H3K27me3_ChIPSeq
GSM1035431	prolif_H3K9me3_ChIPSeq
GSM1035432	ras_input_DNA
GSM1035433	ras_SUMO1_rep1_ChIPSeq
GSM1035434	ras_SUMO1_rep2_ChIPSeq
GSM1035435	ras_SUMO2_ChIPSeq
GSM1035436	ras_Ubc9_ChIPSeq
GSM1035437	ras_PolII_ChIPSeq
GSM1035438	ras_H3K4me3_ChIPSeq
GSM1035439	ras_H3K27me3_ChIPSeq
GSM1035440	ras_H3K9me3_ChIPSeq
GSM1035441	prolif_PIASy_ChIPSeq
GSM1035442	ras_PIASy_ChIPSeq
GSM1035443	Proliferatives_rep1_RNAseq
GSM1035444	Proliferatives_rep2_RNAseq
GSM1035445	Proliferatives_rep3_RNAseq
GSM1035446	Ras_rep1_RNAseq
GSM1035447	Ras_rep2_RNAseq
GSM1035448	Ras_rep3_RNAseq

RNA-seqデータは6サンプル分あることがわかる。これを見た段階で「Proliferatives vs. Ras」という2群間比較データであることがわかる。今ここで見ているのはNCBI GEOのサイトを眺めているが、日米欧の三極で提供しているファイル形式が異なる点に注意。

FASTQ形式ファイルの利用が基本

様々なファイル形式…

- 情報量：SRA−full ＞ SRA−lite ＞ FASTQ ＞ FASTA
  - SRA-full：塩基配列、クオリティ情報、Intensity情報など画像以外の全て
  - SRA-lite：SRA-fullからIntensity情報を除いて軽量化したもの
  - FASTQ：塩基配列とクオリティ情報のみからなるもの
  - FASTA：塩基配列のみからなるもの
  - ファイルサイズ (SRA-full : SRA-lite : FASTQ : FASTA)
    - 6 : 3 : 2 : 1
    - 例：SRA-fullはFASTQの約3倍

http://rgm22.nig.ac.jp/mediawiki−ogareport/index.php/RAW_DATA_archiving/sharing_at_DDBJ

FASTA形式とFASTQ形式

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
  - 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

ファイル形式の違い

- European Nucleotide Archive (ENA; 欧)
  - FASTQ形式(gzip圧縮)
- NCBI Sequence Read Archive (SRA; 米)
  - SRA形式
- DDBJ Sequence Read Archive (DRA; 日)
  - FASTQ形式(bzip2圧縮)
  - SRA-Lite形式

トランスクリプトーム(RNA-seq)データの場合は、NCBI GEO、EMBL-EBI ArrayExpress、DDBJ SRAのいずれかになる。どこでもいいので自分が取得したいデータセットの実験デザインのメタデータ(全体像)情報を把握しておく。門田的には、ArrayExpressが一番理解しやすい。後述するQuasRパッケージを用いてマッピングをする際の入力データはFASTQ形式ファイル。MacintoshはダメだがWindowsだとgzip圧縮ファイルも受け付けてくれるので、事実上ArrayExpress (つまりEBI)から取得するのが一番手っ取り早い。

NGSデータ取得

SRAdbパッケージを利用すれば、ファイルを一つ一つダウンロードする手間が省けます。但し、GSEから始まるIDは受け付けないので、SRPから始まるID情報を予め入手しておく必要がある。**ダウンロード済みなので絶対にやらないで!!**

- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2014/11/12)
- イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25)
- イントロ | NGS | 配列取得 | FASTQ or SRA | 公共DBから (last modified 2015/02/23) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRA | SRAdb(Zhu_2013) (last modified 2014/06/26)
- イントロ | NGS | 配列取得 | シミュレーションデータ | について (last modified 2015/01/18)
- イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成から (last modified 2015/01/18)

イントロ | NGS | 配列取得 | FASTQ or SRA | SRAdb(Zhu_2013)

SRAdbパッケージを用いてRNA-seq配列を取得するやり方を示します。SRA形式ファイルの場合はNCBIからダウンロードしているようですが、FASTQ形式ファイルの場合はEBIからダウンロードしているようです(2014年6月23日、孫建強氏提供情報)。ここではFASTQファイルをダウンロードするやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. RNA-seqデータ("SRA000299":Marioni et al., Genome Res., 2008)の実験デザインの全体像やファイルサイズを眺める場合:

論文中の記述が
かったうえで下
います:腎臓3.(
(SRX000604)。

```
param <- "S

#必要なパッ
library(SRA

#前処理
#sqlfile <-
sqlfile <-
sra_con <-

#本番(実験デ
hoge <- sra
hoge
```

3. RNA-seqデータ("SRP017142":Neyret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"                    #取得したいSRA IDを指定

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

#前処理
#sqlfile <- "SRAmetadb.sqlite"          #最新でなくてもよく、手元に予めダウンロードして
sqlfile <- getSRAdbFile()               #最新のSRAmetadb SQLiteファイルをダウンロード
sra_con <- dbConnect(SQLite(), sqlfile)#おまじない

#前処理(実験デザインの全体像を表示)
```

NGSデータ取得

原著論文中の記述はGSE42213のみ。重要なのは全体像の理解です。これはChIP-seq(GSE42211)とRNA-seq(GSE42212)をまとめたID。

NCBI

Gene Expression Omnibus

HOME | SEARCH | SITE MAP | GEO Publications | FAQ | MIAME | Email GEO

NCBI > GEO > **Accession Display**

Not logged in | Login

Scope: Self | Format: HTML | Amount: Quick | GEO accession: GSE42213 | GO

Series GSE42213 Query DataSets for GSE42213

Status	Public on Jul 24, 2013
Title	SUMO is an Integral and Instructive Compone and Senescence
Organism	Homo sapiens
Experiment type	Genome binding/occupancy profiling by high th Expression profiling by high throughput sequen
Summary	This SuperSeries is composed of the SubSeries
Overall design	Refer to individual Series
Citation(s)	Neyret-Kahn H, Benhamed M, Ye T, Le Gras S governs coordinated repression of a transcripti growth and proliferation. *Genome Res* 2013 Oc PMID: 23893515
Submission date	Nov 09, 2012
Last update date	Feb 17, 2015
Contact name	Tao YE
Organization name	IGBMC (CNRS/INSERM/UDS)
Street address	1 rue Laurent Fries
City	Illkirch
ZIP/Postal code	67404
Country	France

Platforms (2)	GPL10999	Illumina Genome Analyzer IIx (Homo sapiens)
	GPL11154	Illumina HiSeq 2000 (Homo sapiens)
Samples (26)	GSM1035423	prolif_input_DNA
More...	GSM1035424	prolif_SUMO1_rep1_ChIPSeq
	GSM1035425	prolif_SUMO1_rep2_ChIPSeq

This SuperSeries is composed of the following SubSeries:

GSE42211 Genome wide occupancy of SUMO machinery in proliferative and Ras-induced senescent human primary fibroblasts

GSE42212 Quantitative analysis of proliferative and Ras-induced senescent human primary fibroblasts transcriptomes

Relations

BioProject PRJNA179295

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE42213_RAW.tar	2.8 Gb	(http)(custom)	TAR (of BED, WIG)

NGSデータ取得

RNA-seqデータのみをまとめたID (GSE42212)のページを眺めると、SRP017142というSRAdbパッケージで入力として利用可能なIDを取得できる。

Scope: Self | Format: HTML | Amount: Quick | GEO accession: GSE42212 | GO

Series GSE42212 — Query DataSets for GSE42212

Status	Public on Jul 24, 2013
Title	Quantitative analysis of proliferative and Ras-induced senescent human primary fibroblasts transcriptomes
Organism	Homo sapiens
Experiment type	Expression profiling by high throughpu
Summary	The goals of this study is to analyse tr and to compare them with ChIPseq proteins of the SUMO machinery
Overall design	mRNA profiling of proliferative versus fibroblasts 5 days post-infection
Contributor(s)	Neyret-Kahn H, Benhamed M, Ye T, Le Dasso M, Seeler J, Davidson I, Dejean
Citation(s)	Neyret-Kahn H, Benhamed M, Ye T, Le governs coordinated repression of a tra growth and proliferation. *Genome Res* PMID: 23893515
Submission date	Nov 09, 2012
Last update date	Dec 22, 2014
Contact name	Tao YE
Organization name	IGBMC (CNRS/INSERM/UDS)
Street address	1 rue Laurent Fries
City	Illkirch
ZIP/Postal code	67404
Country	France

Platforms (1) GPL10999 Illumina Genome Analyzer IIx (Homo sapiens)

Samples (6) GSM1035443 Proliferatives_rep1_RNAseq
More... GSM1035444 Proliferatives_rep2_RNAseq
GSM1035445 Proliferatives_rep3_RNAseq

This SubSeries is part of SuperSeries:
GSE42213 SUMO is an Integral and Instructive Component of Chromatin in Cell Growth and Senescence

Relations

BioProject	PRJNA179305
SRA	SRP017142

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE42212_RNASeq_RPKM.txt.gz	11.4 Kb	(ftp)(http)	TXT
SRP/SRP017/SRP017142		(ftp)	SRA Study

Raw data provided as supplementary file

Processed data is available on Series record

NGSデータ取得

実習環境では、「デスクトップ – hoge – mapping_SRP017142」フォルダ中に**NGSデータは既に存在**する。ここではフォルダ中に何もない状態でNGSデータをダウンロードする手順を示す。

- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2014/11/12)
- イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25)
- イントロ | NGS | 配列取得 | FASTQ or SRA | 公共DBから (last modified 2015/02/23) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRA | SRAdb(Zhu_2013) (last modified 2014/06/26)
- イントロ | NGS | 配列取得 | シミュレーションデータ | について (last modified 2015/01/18)
- イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成から (last modified 2015/01/18)

イントロ | NGS | 配列取得 | FASTQ or SRA | SRAdb(Zhu_2013)

SRAdbパッケージを用いてRNA-seq配列を取得するやり方を示します。SRA形式ファイルの場合はNCBIからダ

3. RNA-seqデータ("SRP017142":Neyret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"                    #取得したいSRA IDを指定

#必要なパッケージをロード
library(SRAdb)

#前処理
#sqlfile <- "SRA
sqlfile <- getSR
sra_con <- dbCon

#前処理(実験デザイ
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_srp017142"
> list.files()
character(0)
>
```

NGSデータ取得

全選択はCTRLとALTキーを押しながらコードの枠内で左クリック。コピペして数分後の状態。SRAmetadb.sqlite.gzというファイルのダウンロードで結構時間がかかる。これはNCBI SRA中の全メタデータ情報を含んでいる。そのおかげで、SRP017142を入力として与えるだけで関連する全サンプルのFASTQファイルの情報を自動的に取り出すことができる。

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFAS をダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見 情報からSRP017142にたどり着いています。計6ファイル、合計6GB程度の容量のファイルがダウ ます。東大の有線LANで一時間弱程度かかります。早く オプションを'ftp'から'fasp'に変更すると時間短縮可能です

```
param <- "SRP017142"                          #取得
#必要なパッケージをロード
library(SRAdb)                                #パ
#前処理
#sqlfile <- "SRAmetadb.sqlite"                #最新
sqlfile <- getSRAdbFile()                     #最新
sra_con <- dbConnect(SQLite(), sqlfile)#お
#前処理(実験デザインの全体像を表示)
hoge <- sraConvert(param, sra_con=sra_con)#
hoge                                          #hog
apply(hoge, 2, unique)                        #hog
getFASTQinfo(in_acc=hoge$run)                 #hog
#本番(FASTQファイルのダウンロード)
getFASTQfile(hoge$run, srcType='ftp')  #hog
```

R Console

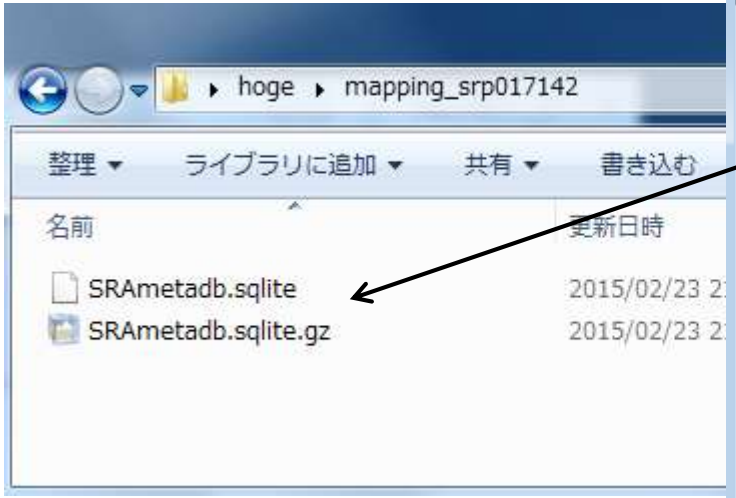
```
> library(SRAdb)
 要求されたパッケージ RSQLite
 要求されたパッケージ DBI をロード中です
 要求されたパッケージ graph をロード中です

 次のパッケージを付け加えます: 'graph'

The
 要
 要求されたパッケージ bitops をロード中です
Setting options('download.file.method.GEOquery'='aut$
>
> #前処理
> #sqlfile <- "SRAmetadb.sqlite"              #最新でなく$
> sqlfile <- getSRAdbFile()                   #最新のSRAm$
 URL 'http://gbnci.abcc.ncifcrf.gov/backup/SRAmetadb$
Content type 'application/x-gzip' length 776702957 b$
 開かれた URL
```

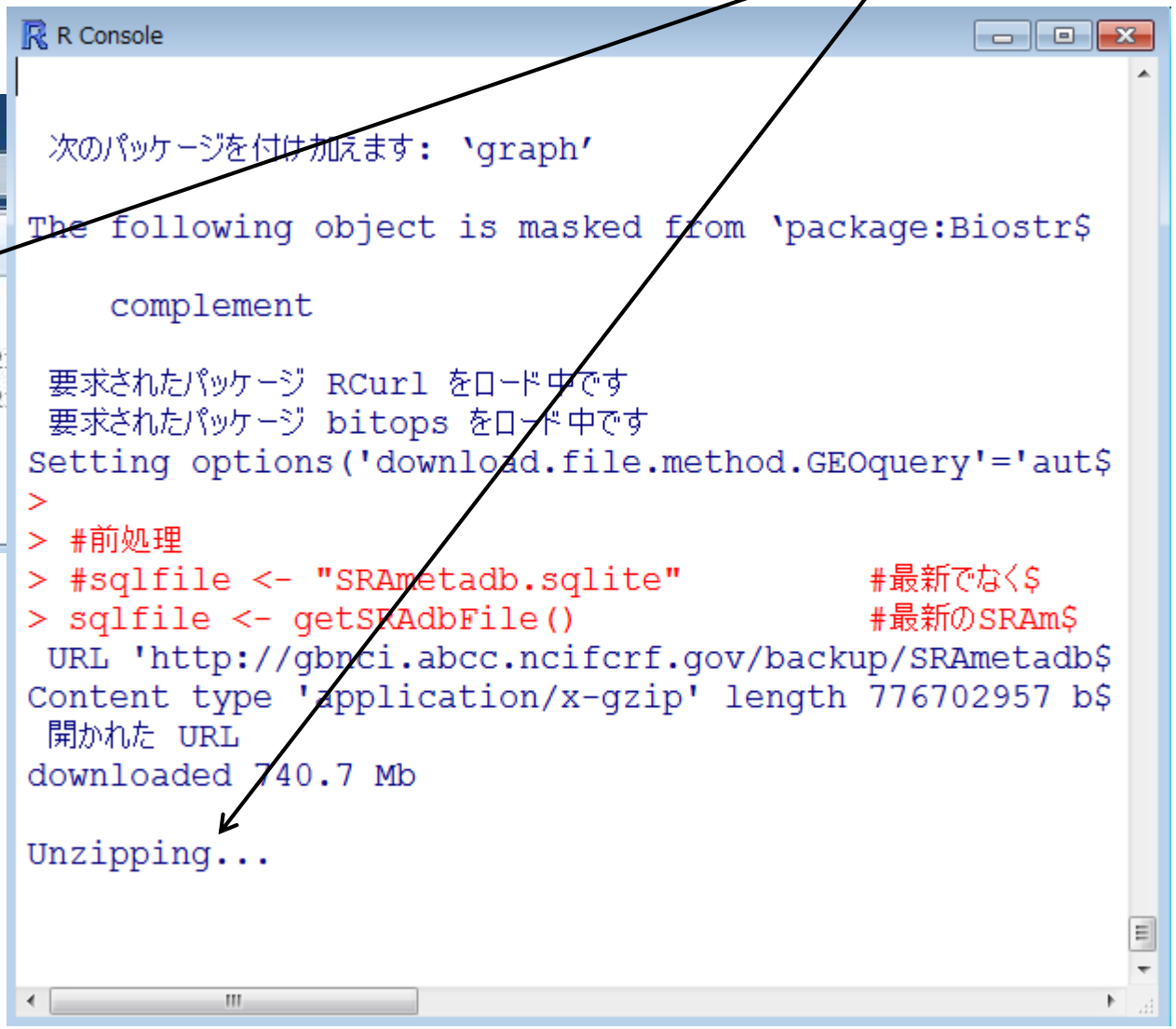
15% downloaded

URL: http://gbnci.abcc.ncifcrf.gov/backup/SRAmetadb.sqlite.gz

NGSデータ取得

SRAmetadb.sqlite.gzというファイルのダウンロードが終了し、解凍(unzip)しているところ。フォルダ上でも解凍されていることがわかる。





これは2015.02.23に行って失敗した例

NGSデータ取得

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮

```
param <- "SRP017142"

#必要なパッケージをロード
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb.sqlite"
sqlfile <- getSRAdbFile()
sra_con <- dbConnect(SQLite(), sqlfi

#前処理(実験デザインの全体像を表示)
hoge <- sraConvert(param, sra_con=sr
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=hoge$run)

#本番(FASTQファイルのダウンロード)
getFASTQfile(hoge$run, srcType='ftp'
```

R Console

```
> hoge2                                    #hoge2の中$
NULL
>
> #本番(FASTQファイルのダウンロード)
> getFASTQfile(hoge$run, srcType='ftp')   #「hoge$run$
Files are saved to:
'C:/Users/kadota/Desktop/hoge/mapping_srp017142'


1
2
3
4
```

R Console

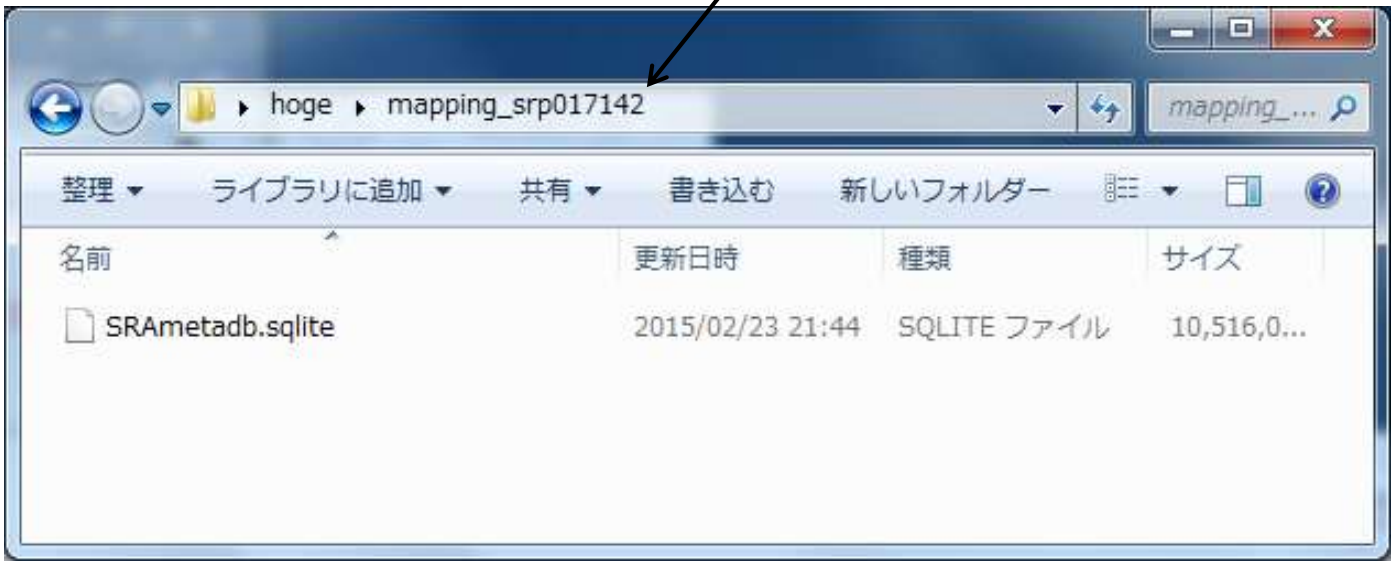
```
639                                                    $
658                                                    $
664                                                    $
670 <p class=legal>Copyright &copy; EMBL-EBI 2$
675                                                    $
683                                                    $
687                                                    $
689                                                    $
690                                                    $
691                                                    $
>
```

NGSデータ取得

正しくダウンロードできていれば*.fastq.gzというファイルが存在するはずであるが…ない。フォルダを眺めてもないので、失敗確定。

R Console

```
687                                                    $
689                                                    $
690                                                    $
691                                                    $
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_srp017142"
> list.files()
[1] "SRAmetadb.sqlite"
>
```



トラブルシューティング

2015.01.27にうまくいった別のPCや、新規購入したPCで気を取り直して再実行

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"                    #取得したいSRA IDを指定

#必要なパッケージをロード
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb.sqlite"
sqlfile <- getSRAdbFile()
sra_con <- dbConnect(SQLite(), sq

#前処理(実験デザインの全体像を表示)
hoge <- sraConvert(param, sra_con
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=hoge$run)

#本番(FASTQファイルのダウンロード)
getFASTQfile(hoge$run, srcType='f
```

R Console

```
 配布条件の詳細に関しては、'license()' あるいは 'licence()' と$

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

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

> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
character(0)
>
```

トラブルシューティング

以前は見られなかったgetFASTQinfo関数実行結果がちゃんと見られていて、明らかに挙動が違う!

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"

#必要なパッケージをロー
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb
sqlfile <- getSRAdbFil
sra_con <- dbConnect(S

#前処理(実験デザインの全
hoge <- sraConvert(par
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=ho

#本番(FASTQファイルのダ
getFASTQfile(hoge$run,
```

R Console

```
> getFASTQinfo(in_acc=hoge$run)          #hoge$runで指定したSRRから始ま$
        run submission     study    sample experiment fastq_ID
1 SRR616151  SRA061444 SRP017142 SRS375081  SRX204181  1212951
2 SRR616152  SRA061444 SRP017142 SRS375082  SRX204182  1212952
3 SRR616153  SRA061444 SRP017142 SRS375083  SRX204183  1212953
4 SRR616154  SRA061444 SRP017142 SRS375084  SRX204184  1212954
5 SRR616155  SRA061444 SRP017142 SRS375085  SRX204185  1212955
6 SRR616156  SRA061444 SRP017142 SRS375086  SRX204186  1212956
                                                                ftp
1 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616151/SRR616151.fastq.gz
2 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616152/SRR616152.fastq.gz
3 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616153/SRR616153.fastq.gz
4 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616154/SRR616154.fastq.gz
5 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616155/SRR616155.fastq.gz
6 ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616156/SRR616156.fastq.gz
                                 md5      bytes
1 79f7663a958458f8fe77e3707f2602d5 1694201684
2 1fd9743365db4c76381a0c5cab018dcd 1657696979
3 b11cd4c2b9e96ff8da2b6e7294461e0c 1452944898
4 8e0a7d2d8b5a096385b07ae76a4dafe8 1025583201
```

ダウンロードができている!

トラブルシューティング

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"

#必要なパッケージをロー
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb
sqlfile <- getSRAdbFil
sra_con <- dbConnect(S

#前処理(実験デザインの全
hoge <- sraConvert(par
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=ho

#本番(FASTQファイルのダ
getFASTQfile(hoge$run,
```

R Console

```
2 2014-01-15 05:59:49 2015-02-21 19:29:02
3 2014-01-15 05:52:10 2015-02-21 19:29:02
4 2014-01-15 05:31:20 2015-02-21 19:29:02
5 2014-01-15 05:53:12 2015-02-21 19:29:02
6 2014-01-15 05:48:28 2015-02-21 19:29:02
>
> #本番(FAST
> getFASTQf
Files are sa
'C:/Users/ka

 URL 'ftp://
ftp data connection made, file length 1694201684 bytes
 開かれた URL
downloaded 1615.7 Mb

 URL 'ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616152/SRR616152.fas$
ftp data connection made, file length 1657696979 bytes
 開かれた URL
```

45% downloaded

URL: ... sra.ebi.ac.uk/vol1/fastq/SRR616/SRR616152/SRR616152.fastq.gz

トラブルシューティング

ダウンロード終了後の状態。
list.files()実行結果からもわかるように、確かに目的のgzip圧縮FASTQファイルが6個存在する。

3. RNA-seqデータ("SRP017142":Neyret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファイルをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計7ファイル、合計6GB程度の容量のファイルがダウンロードされます。東大の有線LANオプションを'ftp'から'fa

```
param <- "SRP017

#必要なパッケージ
library(SRAdb)

#前処理
#sqlfile <- "SRA
sqlfile <- getSR
sra_con <- dbCon

#前処理(実験デザイ
hoge <- sraConve
hoge
apply(hoge, 2, u
getFASTQinfo(in_

#本番(FASTQファイ
getFASTQfile(hog
```

R Console

```
                                md5      bytes           audit_time
1 79f7663a958458f8fe77e3707f2602d5 1694201684 2014-01-15 06:01:01
2 1fd9743365db4c76381a0c5cab018dcd 1657696979 2014-01-15 05:59:49
3 b11cd4c2b9e96ff8da2b6e7294461e0c 1452944898 2014-01-15 05:52:10
4 8e0a7d2d8b5a096385b07ae76a4dafe8 1025583201 2014-01-15 05:31:20
5 3c3fefcef86d12cdd8769ebb5395e41b 1482959798 2014-01-15 05:53:12
6 2a15030e622a5b939ccd172262dcb2f4 1372463724 2014-01-15 05:48:28
        sradb_updated
1 2015-02-21 19:29:02
2 2015-02-21 19:29:02
3 2015-02-21 19:29:02
4 2015-02-21 19:29:02
5 2015-02-21 19:29:02
6 2015-02-21 19:29:02
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
[1] "SRAmetadb.sqlite"   "SRR616151.fastq.gz" "SRR616152.fastq.gz"
[4] "SRR616153.fastq.gz" "SRR616154.fastq.gz" "SRR616155.fastq.gz"
[7] "SRR616156.fastq.gz"
>
```

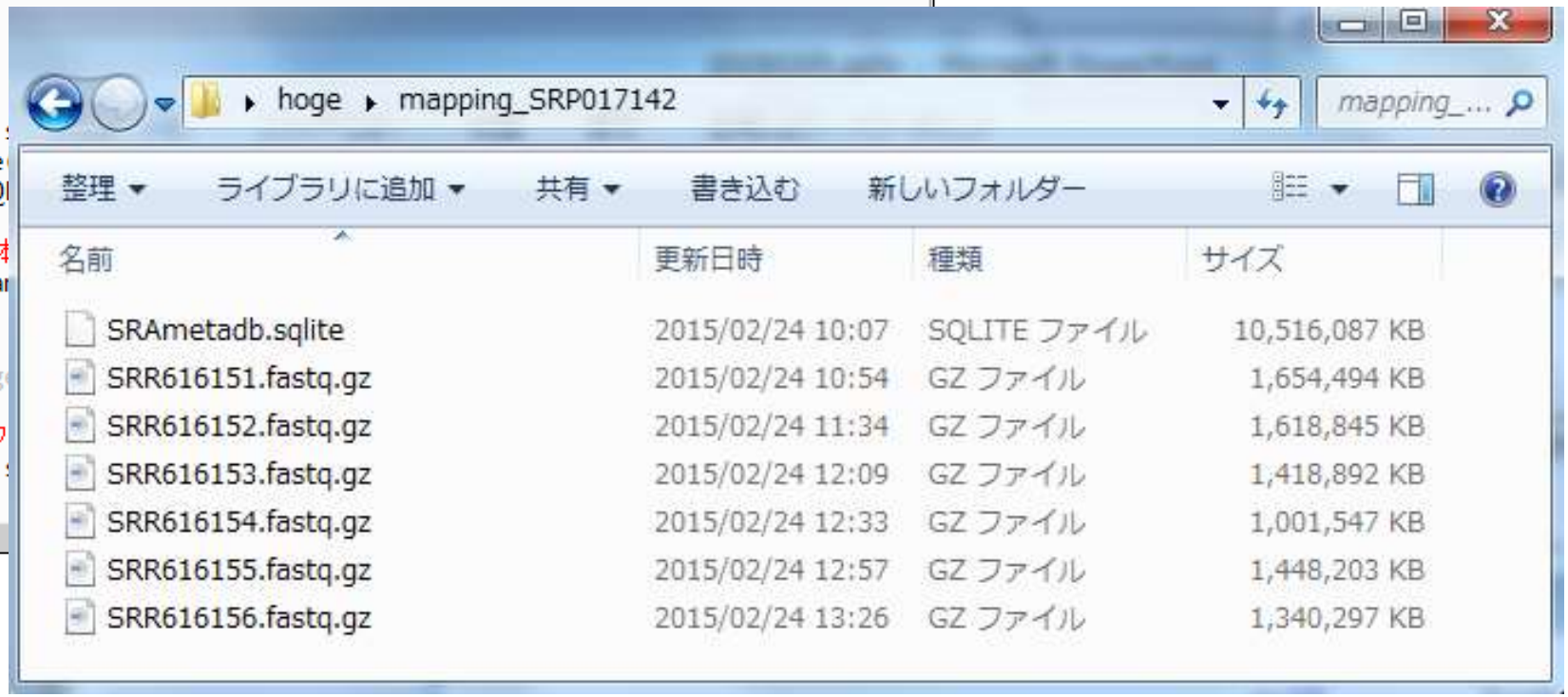
トラブルシューティング

gzip圧縮状態でも6個のFASTQファイルだけで8GB以上になっていることがわかる。ここまでで、**マップする側のRNA-seqデータ取得が完了。**

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済みのFASTQファをダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められていることを見出し、その情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイルがダウンロードされます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のgetFASTQfile関数のオプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"                    #取得したいSRA IDを指定

#必要なパッケージをロード
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb.
sqlfile <- getSRAdbFile
sra_con <- dbConnect(SQ

#前処理(実験デザインの全体
hoge <- sraConvert(para
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=hog

#本番(FASTQファイルのダウ
getFASTQfile(hoge$run,
```



Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

QC

- イントロ | ファイル形式の変換 | qseq --> FASTA (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/11/06)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 前処理 | クオリティチェック | PHREDスコアに変換 (last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 2014/03

クオリティーコントロール(Quality Control)、クオリティーチェック(Quality Check)、おそらくどちらの略と解釈してもよいのでしょう。

前処理 | クオリティチェック | について

Quality Control (QC)を実行する様々な方法をリストアップします。Krakenなどアダプター配列除去などが行えるものも含みます。

R用:

- qrqc: 原著論文なし
- PIQA: Martinez-Alcantara et al., Bioinformatics, 2009
- ShortRead: Morgan et al., Bioinformatics, 2009
- girafe: Toedling et al., Bioinformatics, 2010
- QuasR: Gaidatzis et al., Bioinformatics, 2014

R以外:

- FastQC: 原著論文なし
- FASTX-ToolKit: 原著論文なし
- SolexaQA: Cox et al., BMC Bioinformatics, 2010
- Quake: Kelley et al., Genome Biol., 2010
- NGSQC: Dai et al., BMC Genomics, 2010
- Cutadapt: Martin, M., EMBnet.journal, 2011
- PRINSEQ: Schmieder and Edwards, Bioinformatics, 2011
- ECHO: Kao et al., Genome Res., 2011
- Btrim: Kong Y., Genomics, 2011
- Hammer: Medvedev et al., Bioinformatics, 2011
- ConDeTri: Smeds et al., PLoS One, 2011
- BIGpre: Zhang et al., Genomics Proteomics Bioinformatics, 2011
- NGS QC Toolkit: Patel et al., PLoS One, 2012
- RobiNA: Lohse et al., Nucleic Acids Res., 2012
- SEQuel: Ronen et al., Bioinformatics, 2012
- AdapterRemoval: Lindgreen S., BMC Res Notes, 2012
- Slim-Filter: Golovko et al., BMC Bioinformatics, 2012
- HTQC: Yang et al., BMC Bioinformatics, 2013
- QC-Chain: Zhou et al., PLoS One, 2013
- Kraken: Davis et al., Methods, 2013
- AlienTrimmer: Criscuolo and Brisse, Genomics, 2013
- NextClip: Leggett et al., Bioinformatics, 2014
- QTrim (Roche/454などの long read用): Shrestha et al., BMC Bioinformatics, 2014
- Trimmomatic: Bolger et al., Bioinformatics, 2014
- Skewer: Jiang et al., BMC Bioinformatics, 2014

Review:

- Paszkiewicz et al., Front Genet., 2014

QC

- 前処理 | クオリティチェック | について

前処理 | クオリティチェック | について

Quality Control (QC)を実行する様々な方法をリストアップします。Krakenなどアダプター配列
去などが行えるものも含みます。

R用:

- qrqc: 原著論文なし
- PIQA: Martinez-Alcantara et al., Bioinformatics, 2009
- ShortRead: Morgan et al., Bioinformatics, 2009
- girafe: Toedling et al., Bioinformatics, 2010
- QuasR: Gaidatzis et al., Bioinformatics, 2014

R以外:

- FastQC: 原著論文なし
- FASTX-ToolKit: 原著論文なし
- SolexaQA: Cox et al., BMC Bioinformatics, 2010
- Quake: Kelley et al., Genome Biol., 2010
- NGSQC: Dai et al., BMC Genomics, 2010
- Cutadapt: Martin, M., EMBnet.journal, 2011
- PRINSEQ: Schmieder and Edwards, Bioinformatics, 2011
- ECHO: Kao et al., Genome Res., 2011
- Btrim: Kong Y., Genomics, 2011
- Hammer: Medvedev et al., Bioinformatics, 2011
- ConDeTri: Smeds et al., PLoS One, 2011
- BIGpre: Zhang et al., Genomics Proteomics Bioinformatics, 2011
- NGS QC Toolkit: Patel et al., PLoS One, 2012
- RobiNA: Lohse et al., Nucleic Acids Res., 2012
- SEQuel: Ronen et al., Bioinformatics, 2012
- AdapterRemoval: Lindgreen S., BMC Res Notes, 2012
- Slim-Filter: Golovko et al., BMC Bioinformatics, 2012
- HTQC: Yang et al., BMC Bioinformatics, 2013
- QC-Chain: Zhou et al., PLoS One, 2013
- Kraken: Davis et al., Methods, 2013
- AlienTrimmer: Criscuolo and Brisse, Genomics, 2013
- NextClip: Leggett et al., Bioinformatics, 2014
- QTrim (Roche/454などの long read用): Shrestha et al., BMC Bioinformatics, 2014
- Trimmomatic: Bolger et al., Bioinformatics, 2014
- Skewer: Jiang et al., BMC Bioinformatics, 2014

FASTQ形式ファイルを入力として全体像を眺める作業。FastQCが有名だが、Rパッケージもいくつかある。2014年6月18日のアグリバイオ大学院講義資料中にQCについての記載あり。

実験デザインや使用する機器にもよるが様々な前処理が行われるようです

QC

- 作業内容
  - フィルタリング（filtering）
    - クオリティ値の低い塩基やリードの除去
    - rRNAやtRNAの除去
  - トリミング（trimming）
    - 最初の35塩基のみ利用など
  - 重複除去（de-duplication）
  - コンタミ（contamination）
  - バーコード配列（barcoding）
  - アダプター配列除去（adapter removal）
  - …

前処理 | クオリティチェック | について

Quality Control (QC)を実行する様々な方法をリストアップします。Krakenなどアダプター配列除去などが行えるものも含みます。

R用:

- qrqc: 原著論文なし
- PIQA: Martinez-Alcantara et al., Bioinformatics, 2009
- ShortRead: Morgan et al., Bioinformatics, 2009
- girafe: Toedling et al., Bioinformatics, 2010
- QuasR: Gaidatzis et al., Bioinformatics, 2014

R以外:

- FastQC: 原著論文なし
- FASTX-ToolKit: 原著論文なし
- SolexaQA: Cox et al., BMC Bioinformatics, 2010
- Quake: Kelley et al., Genome Biol., 2010
- NGSQC: Dai et al., BMC Genomics, 2010
- Cutadapt: Martin, M., EMBnet.journal, 2011
- PRINSEQ: Schmieder and Edwards, Bioinformatics, 2011
- ECHO: Kao et al., Genome Res., 2011
- Btrim: Kong Y., Genomics, 2011
- Hammer: Medvedev et al., Bioinformatics, 2011
- ConDeTri: Smeds et al., PLoS One, 2011
- BIGpre: Zhang et al., Genomics Proteomics Bioinformatics, 2011
- NGS QC Toolkit: Patel et al., PLoS One, 2012
- RobiNA: Lohse et al., Nucleic Acids Res., 2012
- SEQuel: Ronen et al., Bioinformatics, 2012
- AdapterRemoval: Lindgreen S., BMC Res Notes, 2012
- Slim-Filter: Golovko et al., BMC Bioinformatics, 2012
- HTQC: Yang et al., BMC Bioinformatics, 2013
- QC-Chain: Zhou et al., PLoS One, 2013
- Kraken: Davis et al., Methods, 2013
- AlienTrimmer: Criscuolo and Brisse, Genomics, 2013
- NextClip: Leggett et al., Bioinformatics, 2014
- QTrim (Roche/454などの long read用): Shrestha et al., BMC Bioinformatics, 2014
- Trimmomatic: Bolger et al., Bioinformatics, 2014
- Skewer: Jiang et al., BMC Bioinformatics, 2014

Review:

- Paszkiewicz et al., Front Genet., 2014

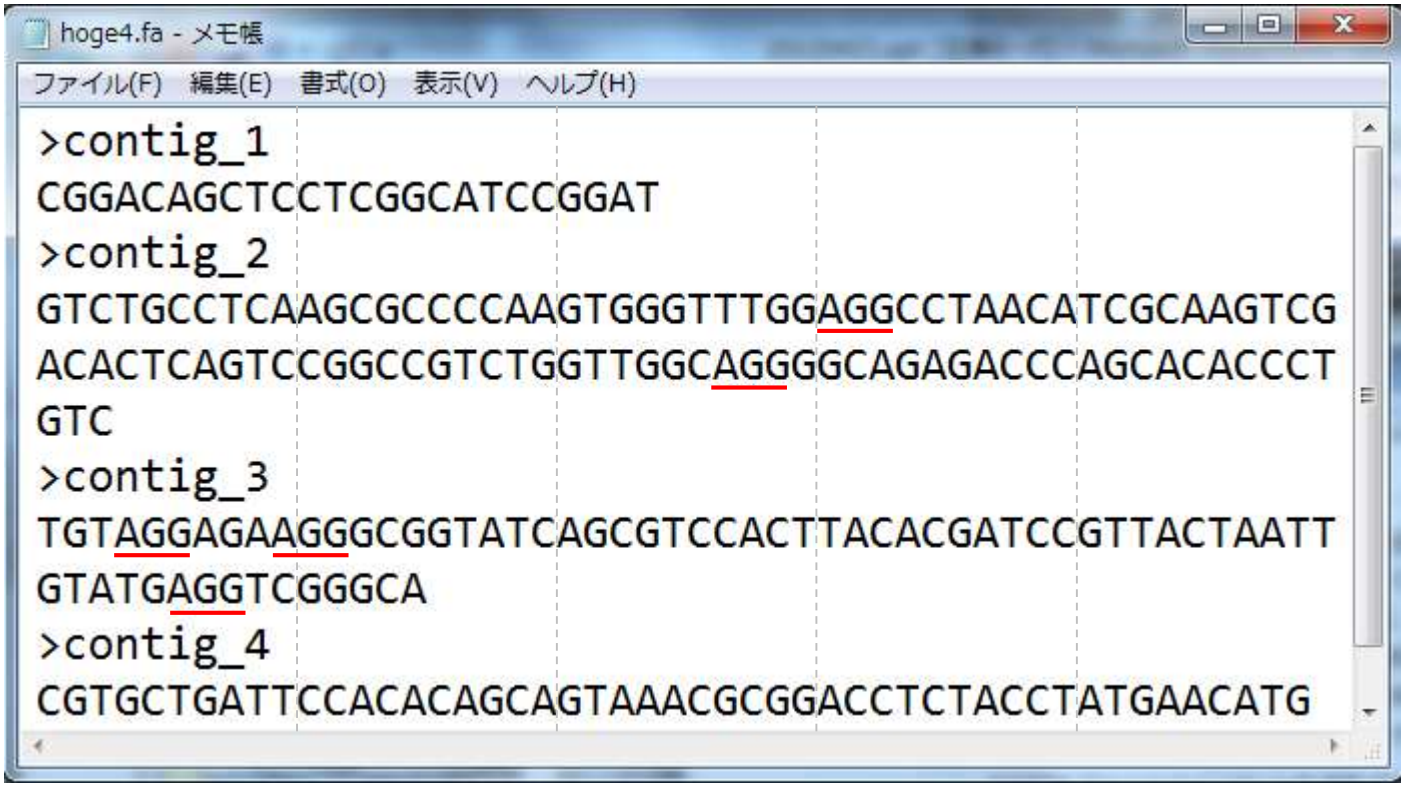
Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

マッピング ＝ 大量高速文字列検索

- マップされる側のリファレンス配列：hoge4.fa
- マップする側のRNA-seqデータ（リードと呼ばれる）："AGG"

マッピングプログラムの出力：（どのリードが）リファレンス配列上のどの位置から転写されたものかという座標情報



出力ファイル

	start	end
contig_2	31	33
contig_2	77	79
contig_3	4	6
contig_3	10	12
contig_3	56	58

マッピング

- 前処理 | トリミング | 指定した末端塩基数だけ除去 (last modified 2013/06/15)
- 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 2014/07/08)
- アセンブル | トランスクリプトーム(転写物)用 (last modified 2014/07/08)
- マッピング | について (last modified 2015/01/16)
- マッピング | basic aligner (last modified 2014/08/08)
- マッピング | splice-aware aligner (last modified 2014/07/09)
- マッピング | Bisulfite sequencing用 (last modified 2014/07/0
- マッピング | (ESTレベルの長さの)contig (last modified 2014
- 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
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力ファイル形式について (last modified 2013/11

Rパッケージとして提供されているものは圧倒的に少ないですが、QuasRパッケージ(Gaidatzis et al., 2014)のおかげでWindows OS上でもマッピングを気軽に利用できるようになりました。これは内部的にBowtie (Langmead et al., 2009)プログラムを利用しています。

マッピング | について

リファレンス配列にマッピングを行うプログラム達です。basic aligner (unspliced aligner)はsplice-aware aligner (spliced aligner)内部で使われていたりします。

R用:

- Rsubread(Windows版なし): Liao et al., Nucleic Acids Res., 2013
- QuasR(Windows版あり): Gaidatzis et al., Bioinformatics, 2014
- HTSeqGenie(Windows版なし): 原著論文はまだみたいです

R以外(basic aligner; unspliced aligner):

- SSAHA2: Ning et al., Genome Res., 2001
- RMAP: Smith et al., BMC Bioinformatics, 2008
- MAQ: Li et al., Genome Res., 2008
- PASS: Campagna et al., Bioinformatics, 2009
- MOM: Eaves and Gao, Bioinformatics, 2009
- Bowtie: Langmead et al., Genome Biol., 2009
- BWA: Li and Durbin, Bioinformatics, 2009(BWA-shortの論文)
- SHRiMP: Rumble et al., PLoS Comput. Biol., 2009

マッピング

オプションは、デフォルトである程度よきに計らってくれるが…実際の挙動を完全に把握できる状況で様々なオプションを試したい

- QuasRパッケージは内部的にBowtieを利用
  - マッピング時に多くのオプションを指定可能
  - “-v”：許容するミスマッチ数を指定するオプション。”-v 0”は、リードがリファレンスに完全一致するもののみレポート。”-v 2”は、2塩基ミスマッチまで許容してマップされうる場所を探索。
  - “-m”：出力するリード条件を指定するオプション。”-m 1”は、複数個所にマップされるリードを除外して、1か所にのみマップされたリードをレポート。”-m 3”は、合計3か所にマップされるリードまでをレポート。
  - “--best --strata”：最も少ないミスマッチ数でマップされるもののみ出力する、という意思表示。これをつけずに”-v 2 -m 1”などと指定すると、たとえ完全一致(ミスマッチ数0)で1か所にのみマップされるリードがあったとしても、どこか別の場所で1塩基ミスマッチでマップされる個所があれば、マップされうる場所が2か所ということを意味し、そのリードは出力されなくなる。それを防ぐのが主な目的
  - …

マッピング

chr3とchr5の違いは、2番目と7番目の塩基のみ。主に”-m”オプションの違いの把握が可能。

- マップされる側のリファレンス配列：ref_genome.fa

ref_genome.fa - メモ帳

ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

コピペで作成しています

マッピング

- マップされる側のリファレンス配列：ref_genome.fa

- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (last
- 書籍
- 書籍
- 書籍

サンプルデータ NEW

1. Illumina/36bp/single-end/human (SRA000299) data (Marioni et al., Genome Res., 2008)
「Kidney 7 samples vs Liver 7 samples」のRNA-seqの遺伝子発現行列データ(SupplementaryTable2.txt)です。サンプルは二つの濃度(1.5 pM and 3 pM)でシーケンスされており、「3 pMのものが5 samples vs. 5

18. ランダムな塩基配列から生成したリファレンスゲノム配列データ(ref_genome.fa)。48, 160, 100, 123, 100 bpの配列長をもつ、計5つの塩基配列を生成しています。description行は"contig"という記述を基本としています。塩基の存在比はAが28%, Cが22%, Gが26%, Tが24%にしています。set.seed関数を利用し、chr3の配列と同じものをchr5としてコピーして作成したのち、2番目と7番目の塩基置換を行っています。そのため、実際に指定するのは最初の4つ分の配列長のみです。

```
out_f <- "ref_genome.fa"                    #出力ファイル名を指定してout_fに格納
param_len_ref <- c(48, 160, 100, 123)       #配列長を指定
narabi <- c("A","C","G","T")                #以下の数値指定時にACGTの並びを間違えないよう
param_composition <- c(28, 22, 26, 24)      #(A,C,G,Tの並びで)各塩基の存在比率を指定
param_desc <- "chr"                         #FASTA形式ファイルのdescription行に記述する
param4 <- 3                                 #コピーを作成したい配列番号を指定
param5 <- c(2, 7)                           #コピー先配列の塩基置換したい位置を指定

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

#塩基置換関数の作成
enkichikan <- function(fa, n) {             #関数名や引数の作成
```

マッピング

- マップする側のRNA-seqデータ：sample_RNAseq1.fa

許容するミスマッチ数による違いや、マップされるべき場所が完全に把握できるように、リードのdescription行に記述されている

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

sample_RNAseq1.fa - メモ帳

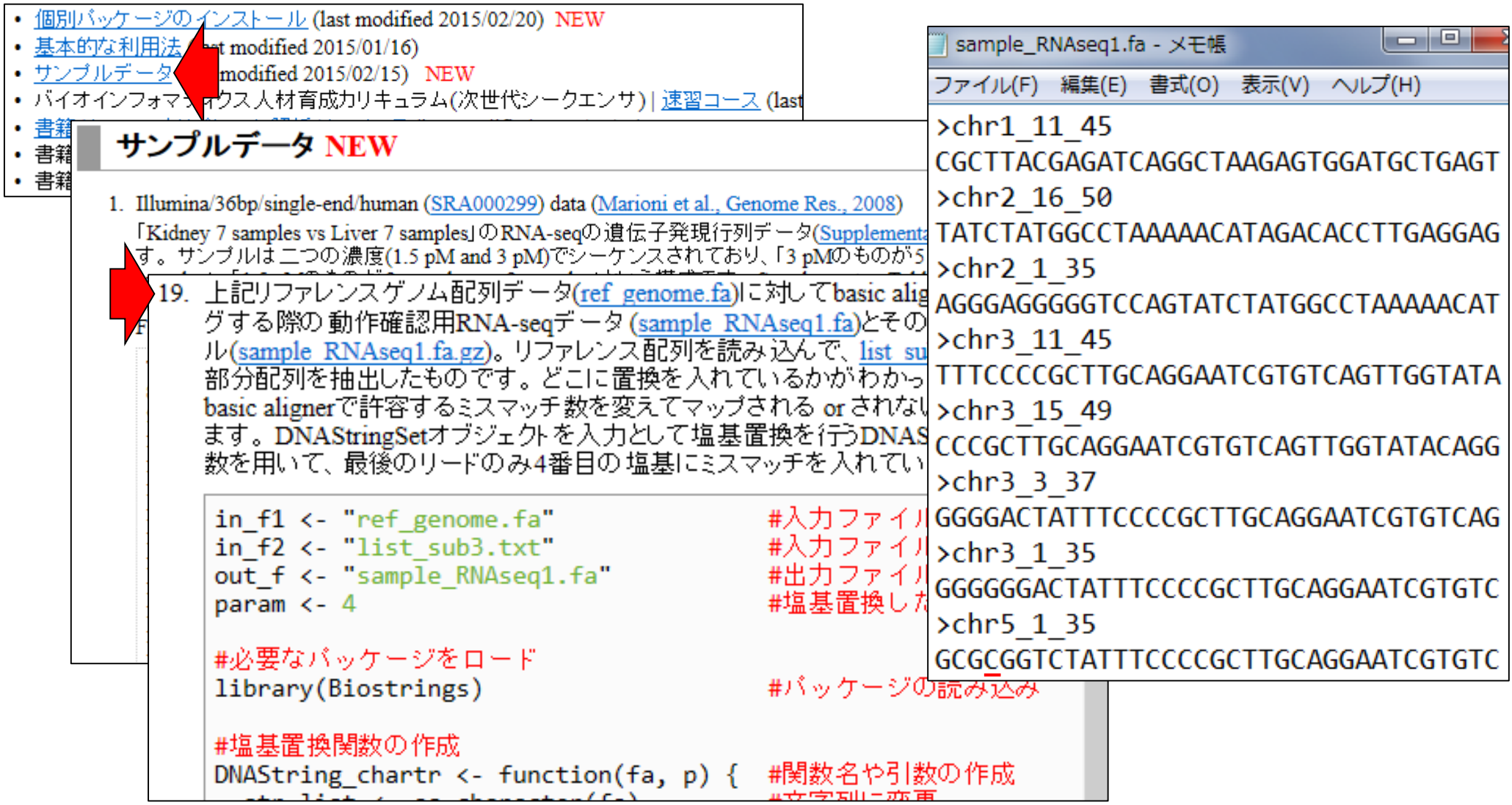
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

コピペで作成しています

マッピング

- マップする側のRNA-seqデータ：sample_RNAseq1.fa



やってみましょう

マッピング

- 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)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力フ
- マップ後 | 出力フ

マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) NEW

QuasRパッケージを用いてsingle-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の一つである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(uniquely 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ペ。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping_single_genome1.txt):

オプションを"-m 1 --best --strata -v 0"とした例です。sample_RNAseq1.faでマップされないのは計3リードです。2リード("chr3_11_45"と"chr3_15_49")はchr5にもマップされるので、"-m 1"オプションで落とされます。1リード("chr5_1_35")は該当箇所と完全一致ではない(4番目の塩基にミスマッチをいれている)ので落とされます。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.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)
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
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
```

やってみましょう

マッピング

- 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)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力フ
- マップ後 | 出力フ

	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

マッピング | single-end | ゲノム | basic aligner(応用) | Qu

QuasRパッケージを用いてsingle-end RNA-seqデータのリファレンスゲノム配列へのマ
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(uniquely 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

複数のRNA-seqサンプルを実行できるようにリストファイルとして与える

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへの

オプションを"-m 1 --best --strata -v 0"とした例です。sample_RNAseq1.faでマップされないのは計3リードです。2リード("chr3_11_45"と"chr3_15_49")はchr5にもマップされるので、"-m 1"オプションで落とされます。1リード("chr5_1_35")は該当箇所と完全一致ではない(4番目の塩基にミスマッチをいれている)ので落とされます。

```
in_f1 <- "mapping_single_genome1.txt"   #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.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)
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
library(GenomicAlignments)              #パッケージの読

#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
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
```

許容するミスマッチ数は0個(”-v 0”)、1か所にマップされるリードのみ出力(”-m 1”)

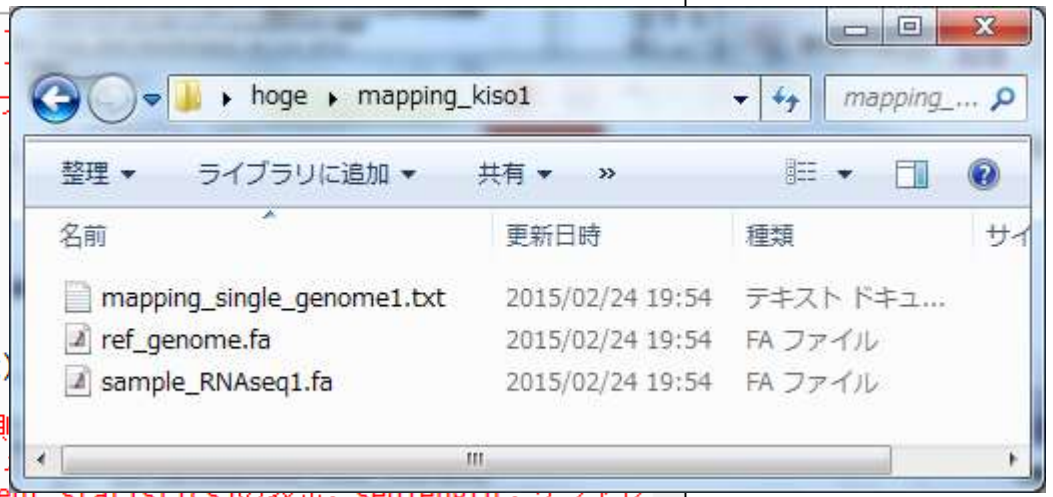
マッピング

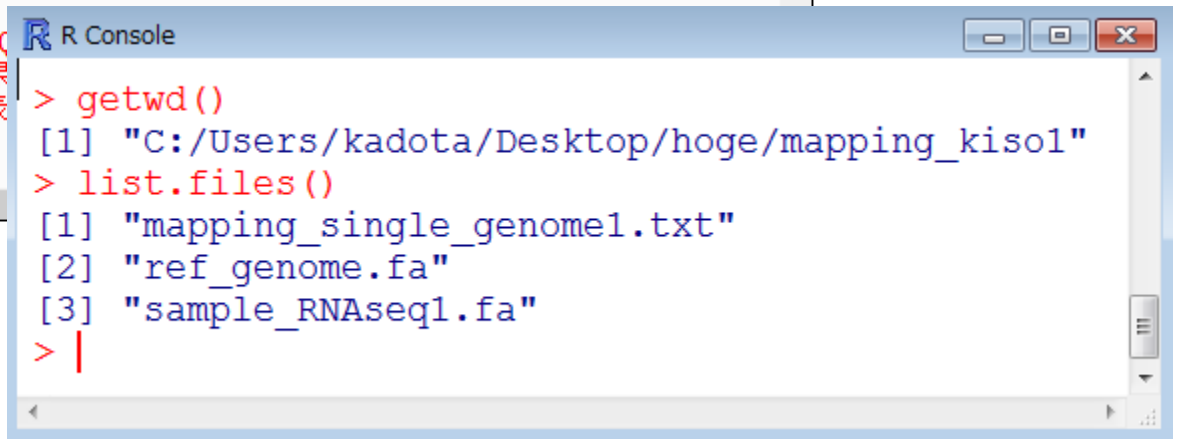
「デスクトップ – hoge – mapping_kiso1」フォルダに入力ファイルは揃っています。他のoutフォルダなどは基本無視でいいです(予備の結果ファイルを格納しています)。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場

オプションを"-m 1 --best --strata -v 0"とした例です。sample_RNAseq1.faでマップされないのは計3リ
("chr3_11_45"と"chr3_15_49")はchr5にもマップされるので、"-m 1"オプションで落とされます。1リー
一致ではない(4番目の塩基にミスマッチをいれている)ので落とされます。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定し
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定し
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオ

#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)
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
out                                    #マッピングに用いたパラ
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示。seqlength:リファレ

#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])#Q
qQCReport(out, pdfFilename=out_f)      #QCレポート結果
out_f                                  #ファイル名を表

#ファイルに保存(BED形式ファイル)
```





マッピング

出力ファイルとして実際に取り扱うのはBAM形式ファイルです

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping_single_genome1.txt):

オプションを"-m 1 --best --strata -v 0"とした例です。sample_RNAseq1.faでマップされないのは計3リードです。2リード("chr3_11_45"と"chr3_15_49")はchr5にもマップされるので、"-m 1"オプションで落とされます。1リード("chr5_1_35")は該当箇所と完全一致ではない(4番目の塩基にミスマッチをいれている)ので落とされます。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.fa"
param_mapping <- "-m 1 --best --st

#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, alignm
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
out_f <- sub(".bam", "_QC.pdf", ou
qQCReport(out, pdfFilename=out_f)
out_f

#ファイルに保存(BED形式ファイル)
```

R Console

```
+    out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名$
+    out_f                                    #ファイル名を表示して$
+    write.table(tmp, out_f, sep="\t", append=F, quote=F, row.na$
+ }
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
>
```

実用上はBAM形式、視覚上はBED形式

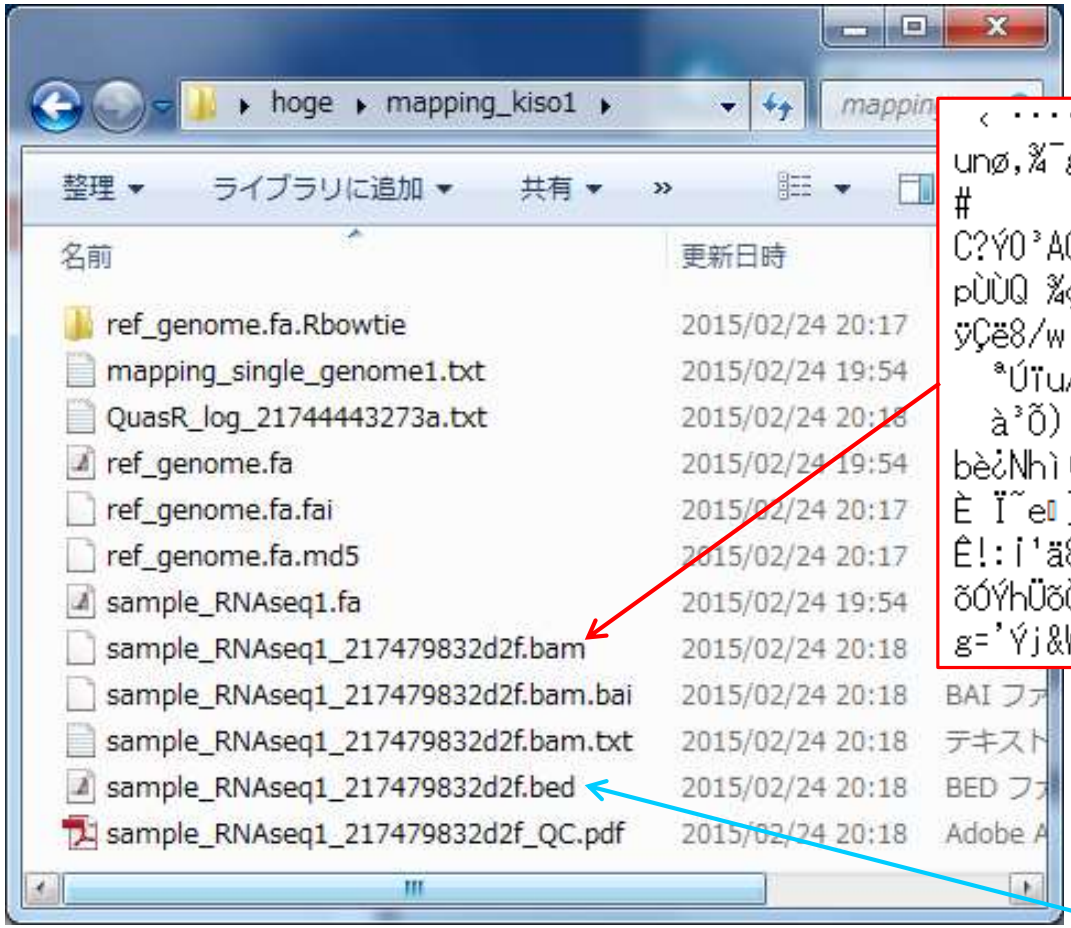
マッピング

- ゲノム上のどの位置にどのリードがマッピングされたか（トランスクリプトームの場合どの転写物配列上のどの位置にどのリードがマッピングされたか）を表す出力ファイル形式は複数あります。
  - SAM (Sequence Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078-2079, 2009)
  - **BAM** (Binary Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078-2079, 2009)
  - **BED** (Browser Extensible Data) format
    - BEDtools (Quinlan et al., *Bioinformatics*, **26**: 841-842, 2010)

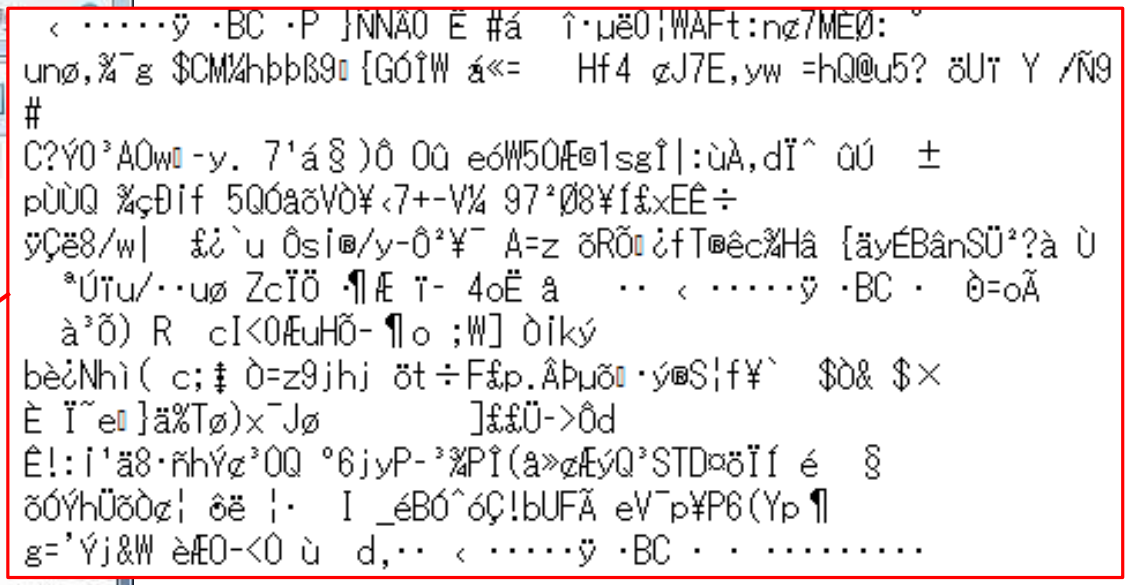
  ...

BEDの最小限の情報は、リードIDを含まない

マッピング



BAM形式ファイル



BED形式ファイル

chr1	11	45
chr2	1	35
chr2	16	50
chr3	1	35
chr3	3	37

使用オプションと結果の解釈

マップされなかったのは、
赤枠の計8リード中3リード

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGA
>chr2
AGGGAGGGGGTCCAGTATC
ACGCAGGTAGGCTGAGGAT
CTCGGGTATGGTTAGTCTT
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

TGGG
GAGGAG
GGGCTG
GGTTCA

chr1	11	45	
chr2	1	35	
chr2	16	50	
chr3	1	35	
chr3	3	37	

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

使用オプションと結果の解釈

完全一致でも複数個所にマップされるために落とされたのは2リード

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

使用オプションと結果の解釈

1か所にのみマップされるがミスマッチのため落とされたのは1リード

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

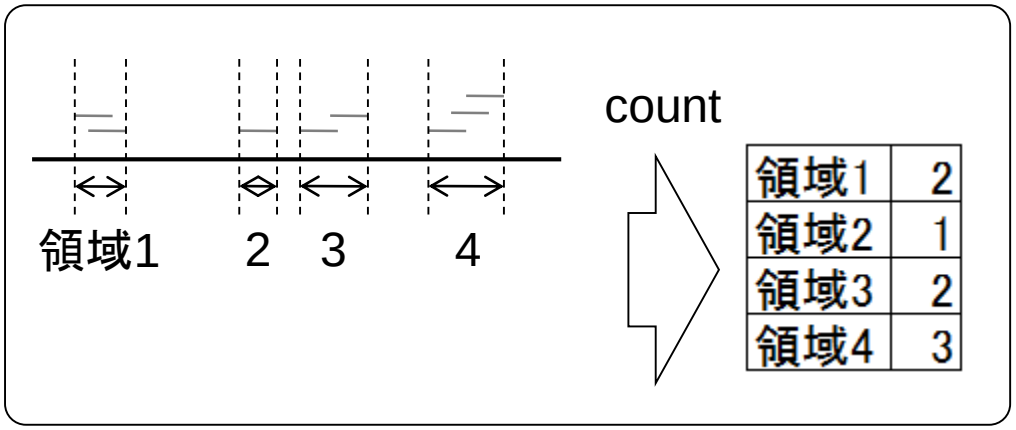
Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

カウント情報取得

Rでのアノテーション情報利用は、TranscriptDbが基本。

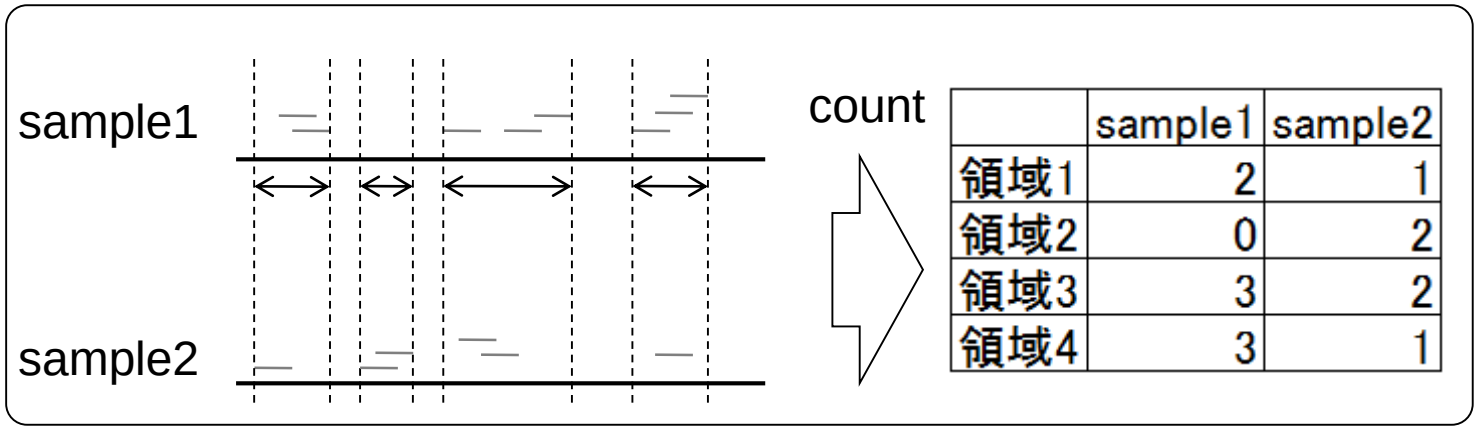
- アノテーション情報を利用する場合
  - UCSC known Genes, Ensembl Genesなど様々なテーブル名を指定可能
  - gene, exon, promoter, junctionなど様々なレベルを指定可能
- アノテーション情報がない場合
  - マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
  - BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



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



	sample1	sample2
領域1	2	1
領域2	0	2
領域3	3	2
領域4	3	1

カウント情報取得1

「デスクトップ − hoge − mapping_kiso1」フォルダに入力ファイルは揃っています。他のoutフォルダなどは基本無視でいいです(予備の結果ファイルを格納しています)。

- マップ後 | 出力ファイルの読み込み | Bowtie形式 (last modified 2013/06/18)
- マップ後 | 出力ファイルの読み込み | SOAP形式 (last modified 2013/06/19)
- 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 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- 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 | 配列長とカウント数の関係 (last modified 2014/07/03)
- 正規化 | について

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014)

QuasRパッケージを用いたsingle-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。アノテーション情報がない場合を想定しているので、GenomicAlignmentsパッケージを利用して、マップされたリードの和集合領域(union range)を得たのち、領域ごとにマップされたリード数をカウントしています。
「ファイル」−「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping_single_genome1.txt):

オプションを"-m 1 --best --strata -v 0"とした例です。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.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)
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

#前処理(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
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
```

カウント情報取得1

「デスクトップ − hoge − mapping_kiso1」フォルダに入力ファイルは揃っています。以前のマッピング結果が残っていても問題ありません。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの

オプションを"-m 1 --best --strata -v 0"とした例です。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.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)
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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間を
out <- qAlign(in_f1, in_f2, alignmentParameter=pa
time_e <- proc.time()                  #計算時間を
time_e - time_s                        #計算時間を
out                                    #マッピング
alignmentStats(out)                    #マッピング

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル名
tmpsname <- out@alignments[,2]         #サンプル名
for(i in 1:length(tmpfname)){          #サンプル数
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式フ
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
>
```

カウント情報取得1

出力ファイルは何も指定していませんが、*_range.txtという名前のファイルが作成されます。これは、.bamという名前のファイルを内部的に入力として読み込み、その文字列中の.bamを_range.txtに置換したものを出力ファイル名として自動作成しているからそうなります。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの
(mapping_single_genome1.txt):

オプションを"-m 1 --best --strata -v 0"とした例です。

```
for(i in 1:length(tmpfname)){          #サンプル数(ファイル数)分だけループを
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式フ
  } else{
    k <- c(k, readGAlignments(tmpfname[i]))#BAM形
  }
}
m <- reduce(granges(k))                #GRangesオ

#本番(カウント情報取得)
tmp <- as.data.frame(m)                #出力ファイ
for(i in 1:length(tmpfname)){          #サンプル数
  tmpcount <- summarizeOverlaps(m, tmpfname[i])#G
  count <- assays(tmpcount)$counts     #Summarize
  colnames(count) <- tmpsname[i]       #行列count
  tmp <- cbind(tmp, count)             #保存したい
}

#ファイルに保存
out_f <- sub(".bam", "_range.txt", tmpfname[i])#
write.table(tmp, out_f, sep="\t", append=F, quote
```

R Console

```
> #ファイルに保存
> out_f <- sub(".bam", "_range.txt", tmpfname[i])#$
> write.table(tmp, out_f, sep="\t", append=F, quote$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
[13] "sample_RNAseq1_217479832d2f_range.txt"
>
```

カウント情報取得1

.bedファイルと*_range.txtファイルを見比べると理解が深まるでしょう。*_range.txtファイルの一番右側の列がカウント情報です。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピン

オプションを"-m 1 --best --strata -v 0"とした例です。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを

#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

#前処理(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
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。
out                                    #マッピングに用いたパラメータや入
```

	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

*.bed

chr1	11	45
chr2	1	35
chr2	16	50
chr3	1	35
chr3	3	37

*_range.txt

seqnames	start	end	width	strand	namae
chr1	11	45	35	+	1
chr2	1	50	50	+	2
chr3	1	37	37	+	2

カウント情報取得2

「デスクトップ − hoge − mapping_kiso2」フォルダに入力ファイルは揃っています。他のoutフォルダなどは基本無視でいいです(予備の結果ファイルを格納しています)。

- 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 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- 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
- 正規化 |

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_201

QuasRパッケージを用いたsingle-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
連の流れを示します。アノテーション情報がない場合を想定しているので、GenomicAlignments パッケージを利用し
合領域(union
「ファイル」−「

1.サンプルデータ

オプションを"-m

```
in_f1 <- "ma
in_f2 <- "re
param_mappir

#必要なパッケ
library(Quas
library(Genc

#前処理(マッ
time_s <- pr
out <- qAlig
time_e <- pr
time_e - tim
out
```

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq2.fa)をref_genome.faにマッピングする場合(mapping_single_genome4.txt):

全部のマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。一般的なカウントデータ行列の形式(2列目以降がカウント情報)にし、配列長情報と別々のファイルにして保存するやり方です。

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seq
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納(リファレ
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_genelength.txt"       #出力ファイル名を指定してout_f2に格納
param_mapping <- "-m 1 --best --strata -v 1"#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

#前処理(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qAl
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
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示
```

カウント情報取得2

「デスクトップ – hoge – mapping_kiso2」フォルダに入力ファイルは揃っています。他のoutフォルダなどは基本無視でいいです(予備の結果ファイルを格納しています)。

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq2.fa)をref_genome.faにマッピングする場合(mapping_single_genome4.txt):

全部のマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。一般的なカウタ行列の形式(2列目以降がカウント情報)にし、配列長情報と別々のファイルにして保存するやり方です。

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファイル名を指定してin_f1に格納(RNA-se
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納(リファ
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_genelength.txt"       #出力ファイル名を指定してout_f2に格納
param_mapping <- "-m 1 --best --strata -v 1"#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
library(GenomicAlignments)             #パッケー

#前処理(マッピング)
time_s <- proc.time()                  #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=p
time_e <- proc.time()                  #計算時間
time_e - time_s                        #計算時間
out                                    #マッピン
alignmentStats(out)                    #マッピン

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル
tmpsname <- out@alignments[,2]         #サンプル
for(i in 1:length(tmpfname)){          #サンプル
```

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$
'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$
'q()' と入力すれば R を終了します。

> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso2"
> list.files()
[1] "mapping_single_genome4.txt"
[2] "ref_genome.fa"
[3] "sample_RNAseq1.fa"
[4] "sample_RNAseq2.fa"
>
```

カウント情報取得2

出力ファイル群のうち、主に取り扱うのはカウントデータを含むファイル(hoge4_count.txt)のほうです

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq2.fa)をref_genome.faにマッピングする場合(mapping_single_genome4.txt):

全部のマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。一般的なカウントデータ行列の形式(2列目以降がカウント情報)にし、配列長情報と別々のファイルにして保存するやり方です。

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファ
in_f2 <- "ref_genome.fa"               #入力ファ
out_f1 <- "hoge4_count.txt"            #出力ファ
out_f2 <- "hoge4_genelength.txt"       #出力ファ
param_mapping <- "-m 1 --best --strata -v 1"#マ

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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=p
time_e <- proc.time()                  #計算時間
time_e - time_s                        #計算時間
out                                    #マッピン
alignmentStats(out)                    #マッピン

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル
tmpsname <- out@alignments[,2]         #サンプル
for(i in 1:length(tmpfname)){          #サンプル
```

R Console

```
> tmp <- cbind(tmp, h$width)                    #和集合$
> write.table(tmp, out_f2, sep="\t", append=F, q$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso2"
> list.files()
 [1] "hoge4_count.txt"
 [2] "hoge4_genelength.txt"
 [3] "mapping_single_genome4.txt"
 [4] "QuasR_log_201814983c7.txt"
 [5] "ref_genome.fa"
 [6] "ref_genome.fa.fai"
 [7] "ref_genome.fa.md5"
 [8] "ref_genome.fa.Rbowtie"
 [9] "sample_RNAseq1.fa"
[10] "sample_RNAseq1_2018acaf46.bam"
[11] "sample_RNAseq1_2018acaf46.bam.bai"
[12] "sample_RNAseq1_2018acaf46.bam.txt"
[13] "sample_RNAseq2.fa"
[14] "sample_RNAseq2_201843d33cbe.bam"
[15] "sample_RNAseq2_201843d33cbe.bam.bai"
[16] "sample_RNAseq2_201843d33cbe.bam.txt"
>
```

カウント情報取得2

リストファイル中で指定したサンプル名がカウントデータ行列の列名となります

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq ref_genome.faにマッピングする場合(mapping_single_genome4.txt):

全部のマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。一タ行列の形式(2列目以降がカウント情報)にし、配列長情報と別々のファイルにして保存

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファイル名を指定してin_f
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f
out_f1 <- "hoge4_count.txt"            #出力ファイル名を指定してout_
out_f2 <- "hoge4_genelength.txt"       #出力ファイル名を指定してout_
param_mapping <- "-m 1 --best --strata -v 1"#マッピング時のオプショ

#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

#前処理(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
```

FileName	SampleName
sample_RNAseq1.fa	sample1
sample_RNAseq2.fa	sample2

tmp	sample1	sample2
chr1_11_45_35_+	1	0
chr2_1_60_60_+	2	1
chr3_1_37_37_+	2	0
chr4_6_65_60_+	0	1
chr5_1_35_35_+	1	0

Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

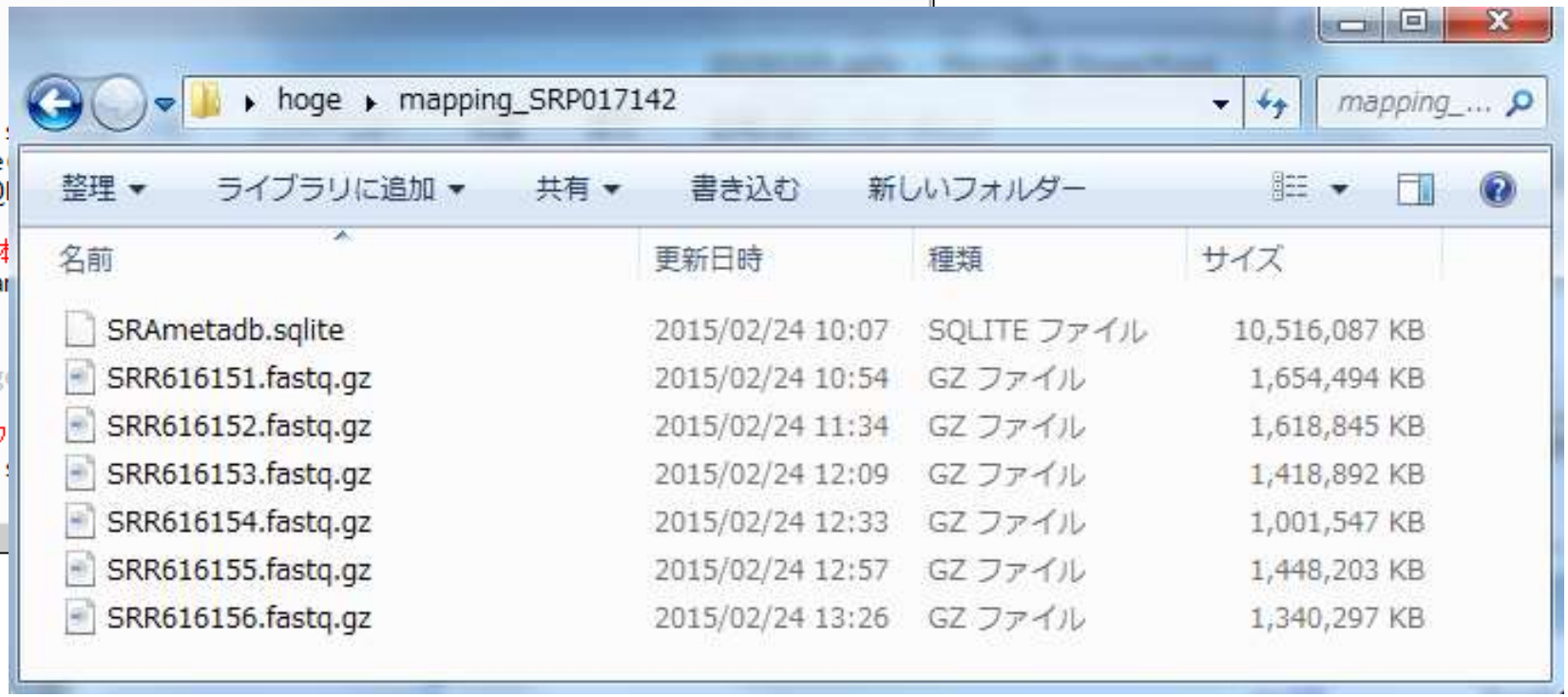
実データ解析

「デスクトップ – hoge – mapping_SRP017142」にダウンロードしておいたリアルRNA-seqデータ(SRP017142)のマッピングおよびカウントデータの取得を行う。ここではQCのステップを(入れなくてもそこそこの情報が得られたので)飛ばすが、通常はQCを行う。

3. RNA-seqデータ("SRP017142":Nevret-Kahn et al., Genome Res., 2013)のgzip圧縮済み をダウンロードする場合:

論文中の記述からGSE42213を頼りに、RNA-seqデータがGSE42212として収められている 情報からSRP017142にたどり着いています。計6ファイル、合計6Gb程度の容量のファイル ます。東大の有線LANで一時間弱程度かかります。早く終わらせたい場合は、最後のget オプションを'ftp'から'fasp'に変更すると時間短縮可能です。

```
param <- "SRP017142"                    #取得したいSRA IDを指定

#必要なパッケージをロード
library(SRAdb)

#前処理
#sqlfile <- "SRAmetadb.
sqlfile <- getSRAdbFile
sra_con <- dbConnect(SQ

#前処理(実験デザインの全体
hoge <- sraConvert(para
hoge
apply(hoge, 2, unique)
getFASTQinfo(in_acc=hog

#本番(FASTQファイルのダウ
getFASTQfile(hoge$run,
```



参考

マッピング（実データ）

マッピングの基本形はこちら。ヒトゲノムのhg19にマップ。BAM形式、BED形式ファイル、およびQCレポートファイルが出力として得られるが、基本的にはBAMファイルが得られれば十分なので、最低限黒枠部分のコピペだけでもよい。

- 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)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マップ後 | について (last modified 2013/06/19)
- マップ後
- マップ後

マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) NEW

QuasRパッケージを用いてsingle-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
Bowtie
指定した
示しません
マッピングす
無視で大丈
「ファイル」

1.サンプル
(mapping_s

オプション
("chr3_11_
箇所と完全

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリに保存したうえで、下記を実行します。BSgenomeパッケージで利用可能なBSgenome.Hsapiens.UCSC.hg19へマッピングしています。名前から推測できるように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
param_mapping <- "-m 1 --best --strata -v 2"#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
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
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)
```

参考

マッピング（実データ）

全部で6つのFASTQファイルのマッピングを行うべく、リストファイルを作業ディレクトリ中に保存。これは、複製あり2群間比較用ヒトRNA-seqデータ(3 Ras vs. 3 Proliferative)。

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリに作成したうえで、下記を実行します。BSgenomeパッケージで利用可能なBSgenome.Hsapiens.UCSC.hg19へマッピングします。名前から推測できるように"UCSC"の"hg19"にマッピングしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"          #
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #
param_mapping <- "-m 1 --best --strata -

#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, alignmentPar
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
[1] "SRAmetadb.sqlite"          "srp017142_samplename.txt"
[3] "SRR616151.fastq.gz"        "SRR616152.fastq.gz"
[5] "SRR616153.fastq.gz"        "SRR616154.fastq.gz"
[7] "SRR616155.fastq.gz"        "SRR616156.fastq.gz"
>
```

FileName	SampleName	
SRR616151.fastq.gz	Pro_rep1	G1群
SRR616152.fastq.gz	Pro_rep2	
SRR616153.fastq.gz	Pro_rep3	
SRR616154.fastq.gz	Ras_rep1	G2群
SRR616155.fastq.gz	Ras_rep2	
SRR616156.fastq.gz	Ras_rep3	

参考

マッピング（実データ）

約10時間かけてマッピングを行ったあとの作業ディレクトリの中身。.bamや.bedのマッピング結果ファイル、そしてQCレポートファイル(*_QC.pdf)が作成されていることが分かる。

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングす

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業デ
たうえで、下記を実行します。BSgenomeパッケージで利用可能なBSgenome.Hsapiens.UCSC.hg19へマッ
す。名前から推測できるように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部
的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。10時間程度かかります。

```
in_f1 <- "srp017142_s
in_f2 <- "BSgenome.Hs
param_mapping <- "-m

#必要なパッケージをロー
library(QuasR)
library(GenomicAlignm

#本番(マッピング)
time_s <- proc.time()
out <- qAlign(in_f1
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
 [1] "QuasR_log_110810e6284f.txt"     "SRAmetadb.sqlite"
 [3] "srp017142_samplename.txt"       "SRR616151.fastq.gz"
 [5] "SRR616151_1108753997.bam"       "SRR616151_1108753997.bam.bai"
 [7] "SRR616151_1108753997.bam.txt"   "SRR616151_1108753997.bed"
 [9] "SRR616151_1108753997_QC.pdf"    "SRR616152.fastq.gz"
[11] "SRR616152_11083b4d1925.bam"     "SRR616152_11083b4d1925.bam.bai"
[13] "SRR616152_11083b4d1925.bam.txt" "SRR616152_11083b4d1925.bed"
[15] "SRR616153.fastq.gz"             "SRR616153_11083aef23ed.bam"
[17] "SRR616153_11083aef23ed.bam.bai" "SRR616153_11083aef23ed.bam.txt"
[19] "SRR616153_11083aef23ed.bed"     "SRR616154.fastq.gz"
[21] "SRR616154_110816c28b2.bam"      "SRR616154_110816c28b2.bam.bai"
[23] "SRR616154_110816c28b2.bam.txt"  "SRR616154_110816c28b2.bed"
[25] "SRR616155.fastq.gz"             "SRR616155_110821c0377c.bam"
[27] "SRR616155_110821c0377c.bam.bai" "SRR616155_110821c0377c.bam.txt"
[29] "SRR616155_110821c0377c.bed"     "SRR616156.fastq.gz"
[31] "SRR616156_1108169b360a.bam"     "SRR616156_1108169b360a.bam.bai"
[33] "SRR616156_1108169b360a.bam.txt" "SRR616156_1108169b360a.bed"
>
```

カウント情報取得(実データ)

この項目では、マッピングからカウントデータ取得までの一通りの作業を一気に行う手順が示されている。

- 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 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- マップ後 | カウント情報取得 | リファレンスゲノムリピート | BEDファイルから (last modified 2014/06/21)
- マップ後 | 配
- 正規化 | に
- 正規化 | 基

マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) NEW

QuasRパッケージを用いたsingle-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。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTranscriptDbオブジェクトをネットワーク経由で取得するのを基本としつつ、TxDbパッケージを読み込むやり方も示しています。マッピングのやり方やオプション
「ファ

1.サンプル

mapping
2列目に
--strata
て、UC

```
in_f1
in_f2
out_f
param
param
param
param

#必要
libra
libra
```

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリに保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。名前から推測できるように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。ここでの目的はUCSC known genesのgene-levelのカウントデータを得ることです。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"    #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンスゲ
out_f <- "hoge6.txt"                   #出力ファイル名を指定してout_fに格納
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーショ
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプションを指定
param_reportlevel <- "gene"            #カウントデータ取得時のレベルを指定："gene", "exon

#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示 seq
```

参考

カウント情報取得(実データ)

基本的には、以下のR Console画面に示すようなリストファイル(srp017142_samplename.txt)とFASTQファイルのみが存在する状況下でマッピングからスタートしてもよいが…。

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピ

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。るように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。ここでの目的はUCSC known genesのgene-levelのカウントデータを得ることです。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンス配
out_f <- "hoge6.txt"                    #出力ファイル名を指定してout_fに格納
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーショ
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプションを指定
param_reportlevel <- "gene"             #カウントデータ取得時のレベルを指定："gene", "exo

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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignm
alignmentStats(out)

#前処理(TranscriptDbオブジェクト作成
library(param_txdb, character.only
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
[1] "SRAmetadb.sqlite"          "srp017142_samplename.txt"
[3] "SRR616151.fastq.gz"        "SRR616152.fastq.gz"
[5] "SRR616153.fastq.gz"        "SRR616154.fastq.gz"
[7] "SRR616155.fastq.gz"        "SRR616156.fastq.gz"
>
```

参考

カウント情報取得(実データ)

マッピング時のオプションが同じ場合には、当然同じ内容のbamやbedファイルが出力されるだけなので無駄。できれば避けたい

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッ

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリに保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。名前から推測できるように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。ここでの目的はUCSC known genesのgene-levelのカウントデータを得ることです。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"    #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンス配
out_f <- "hoge6.txt"                   #出力ファイル名を指定してout_fに格納
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーショ
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプションを指定
param_reportlevel <- "gene"            #カウントデータ取得時のレベルを指定："gene", "exon

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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignm
alignmentStats(out)

#前処理(TranscriptDbオブジェクト作成
library(param_txdb, character.only
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
[1] "SRAmetadb.sqlite"          "srp017142_samplename.txt"
[3] "SRR616151.fastq.gz"        "SRR616152.fastq.gz"
[5] "SRR616153.fastq.gz"        "SRR616154.fastq.gz"
[7] "SRR616155.fastq.gz"        "SRR616156.fastq.gz"
>
```

参考

カウント情報取得(実データ)

以前のマッピング結果ファイルを含む以下のような状態からスタートしてもよい。QuasRパッケージは、おそらくこのlogファイルを眺めて同じオプションでマップした結果があるかどうかを判定し、なければ新たにマッピング、あれば既存のbamファイルを利用してその後の解析を行う。

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマ

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.t
保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています
るように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを
ここではマッピング時の
gene-levelのカウントデー

```
in_f1 <- "srp0171
in_f2 <- "BSgenom
out_f <- "hoge6.t
param_txdb <- "Tx
param_mapping <-
param_reportlevel

#必要なパッケージを
library(QuasR)
library(GenomicAl

#前処理(マッピング)
out <- qAlign(in_
alignmentStats(ou

#前処理(Transcript
library(param_txd
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_S
> list.files()
 [1] "QuasR_log_110810e6284f.txt"      "SRAmetadb.sqlite"
 [3] "srp017142_samplename.txt"        "SRR616151.fastq.gz"
 [5] "SRR616151_1108753997.bam"        "SRR616151_1108753997.bam.bai"
 [7] "SRR616151_1108753997.bam.txt"    "SRR616151_1108753997.bed"
 [9] "SRR616151_1108753997_QC.pdf"     "SRR616152.fastq.gz"
[11] "SRR616152_11083b4d1925.bam"      "SRR616152_11083b4d1925.bam.bai"
[13] "SRR616152_11083b4d1925.bam.txt" "SRR616152_11083b4d1925.bed"
[15] "SRR616153.fastq.gz"              "SRR616153_11083aef23ed.bam"
[17] "SRR616153_11083aef23ed.bam.bai" "SRR616153_11083aef23ed.bam.txt"
[19] "SRR616153_11083aef23ed.bed"      "SRR616154.fastq.gz"
[21] "SRR616154_110816c28b2.bam"       "SRR616154_110816c28b2.bam.bai"
[23] "SRR616154_110816c28b2.bam.txt"   "SRR616154_110816c28b2.bed"
[25] "SRR616155.fastq.gz"              "SRR616155_110821c0377c.bam"
[27] "SRR616155_110821c0377c.bam.bai" "SRR616155_110821c0377c.bam.txt"
[29] "SRR616155_110821c0377c.bed"      "SRR616156.fastq.gz"
[31] "SRR616156_1108169b360a.bam"      "SRR616156_1108169b360a.bam.bai"
[33] "SRR616156_1108169b360a.bam.txt" "SRR616156_1108169b360a.bed"
>
```

参考

カウント情報取得(実データ)

マッピング結果を含む状態で左記のコードを実行した途中経過。改めてマッピングを行う必要性はないと判断して次の作業(alignmentStats関数の実行)に移行している。マッピング結果がない場合には、qAlign関数実行のところで10時間ほどかけてマッピングを行う。

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピ

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。るように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部ここではマッピング時の
gene-levelのカウントデー

```
in_f1 <- "srp0171
in_f2 <- "BSgenom
out_f <- "hoge6.t
param_txdb <- "Tx
param_mapping <-
param_reportlevel

#必要なパッケージを
library(QuasR)
library(GenomicAl

#前処理(マッピング)
out <- qAlign(in_
alignmentStats(ou

#前処理(Transcript
library(param_txd
```

R Console

```
[29] "SRR616155_110821c0377c.bed"      "SRR6161
[31] "SRR616156_1108169b360a.bam"      "SRR6161
[33] "SRR616156_1108169b360a.bam.txt" "SRR6161
> in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1$
> in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2$
> out_f <- "hoge6.txt"                    #出力ファイル名を指定してout_f$
> param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(T$
> param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプション$
> param_reportlevel <- "gene"             #カウントデータ取得時のレベルを$
>
> #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
>
> #前処理(マッピング)
> out <- qAlign(in_f1, in_f2,alignmentParameter=param_mapping)#マッピン$
all necessary alignment files found
> alignmentStats(out)                     #マッピング結果(alignment stati$
                  seqlength    mapped unmapped
Pro_rep1:genome 3137161264 23333855  8917379
Pro_rep2:genome 3137161264 23512357  8606211
```

参考

カウント情報取得(実データ)

アノテーション情報を含むtxdbオブジェクトの作成やqCount関数を用いたカウント情報取得のところで十数分程度かかるが、マッピング所要時間に比べたら誤差範囲

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッ

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.tx
保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。
るように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、
ここではマッピング時の
gene-levelのカウントデー

```
in_f1 <- "srp0171
in_f2 <- "BSgenom
out_f <- "hoge6.t
param_txdb <- "Tx
param_mapping <-
param_reportlevel

#必要なパッケージを
library(QuasR)
library(GenomicAl

#前処理(マッピング)
out <- qAlign(in_
alignmentStats(ou

#前処理(Transcript
library(param_txdb
```

R Console

```
    species

The following object is masked from 'package:GenomeInfoDb':

    species

> tmp <- ls(paste("package", param_txdb, sep=":"))#指定したパッケージで$
> txdb <- eval(parse(text=tmp))                  #文字列tmpをRオブジェクトとして$
>
> #本番(カウントデータ取得)
> count <- qCount(out, txdb, reportLevel=param_reportlevel)#カウントデー$
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> data <- count[,-1]                             #カウント情報をdataに格納
>
> #ファイルに保存
> tmp <- cbind(rownames(data), data)             #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmp$
>
```

参考

カウント情報取得(実データ)

目的の遺伝子レベルのカウントデータファイル(hoge6.txt)が作成されていることがわかる。

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリに保存したうえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。名前から推測できるように"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここではマッピング時のオプションを"-m 1 --best --strata -v 2"にしています。ここでの目的はUCSC known genesのgene-levelのカウントデータを得る

```
in_f1 <- "srp017142_samp
in_f2 <- "BSgenome.Hsapi
out_f <- "hoge6.txt"
param_txdb <- "TxDb.Hsap
param_mapping <- "-m 1 -
param_reportlevel <- "ge

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

#前処理(マッピング)
out <- qAlign(in_f1, in_
alignmentStats(out)

#前処理(TranscriptDbオブ
library(param_txdb, char
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_SRP017142"
> list.files()
 [1] "hoge6.txt"                          "QuasR_log_110810e6284f.txt"
 [3] "SRAmetadb.sqlite"                   "srp017142_samplename.txt"
 [5] "SRR616151.fastq.gz"                 "SRR616151_1108753997.bam"
 [7] "SRR616151_1108753997.bam.bai"       "SRR616151_1108753997.bam.txt"
 [9] "SRR616151_1108753997.bed"           "SRR616151_1108753997_QC.pdf"
[11] "SRR616152.fastq.gz"                 "SRR616152_11083b4d1925.bam"
[13] "SRR616152_11083b4d1925.bam.bai"     "SRR616152_11083b4d1925.bam.txt"
[15] "SRR616152_11083b4d1925.bed"         "SRR616153.fastq.gz"
[17] "SRR616153_11083aef23ed.bam"         "SRR616153_11083aef23ed.bam.bai"
[19] "SRR616153_11083aef23ed.bam.txt"     "SRR616153_11083aef23ed.bed"
[21] "SRR616154.fastq.gz"                 "SRR616154_110816c28b2.bam"
[23] "SRR616154_110816c28b2.bam.bai"      "SRR616154_110816c28b2.bam.txt"
[25] "SRR616154_110816c28b2.bed"          "SRR616155.fastq.gz"
[27] "SRR616155_110821c0377c.bam"         "SRR616155_110821c0377c.bam.bai"
[29] "SRR616155_110821c0377c.bam.txt"     "SRR616155_110821c0377c.bed"
[31] "SRR616156.fastq.gz"                 "SRR616156_1108169b360a.bam"
[33] "SRR616156_1108169b360a.bam.bai"     "SRR616156_1108169b360a.bam.txt"
[35] "SRR616156_1108169b360a.bed"
>
```

6.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

マップしたいFASTQファイルリストおよびそのサンプル名を記述したsrp017142_samplename.txtを作業ディレクトリ
うえで、下記を実行します。BSgenome.Hsapiens.UCSC.hg19へマッピングしています。名前から推測できるよう
に"UCSC"の"hg19"にマップしているのと同じです。basic alignerの一つであるBowtieを内部的に用いており、ここ
ング時のオプションを"-m 1 --best --strata -v 2"にしています。ここでの目的はUCSC known genesのgene-levelのカ
タを得ることです。10時間程度かかります。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンス
out_f <- "hoge6.txt"                    #出力ファイル名を指定してout_fに格納
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーシ
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプションを指定
param_reportlevel <- "gene"             #カウントデータ取得時のレベルを指定："gene", "exo

#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関
alignmentStats(out)                     #マッピング結果(alignment statistics)の表示。se

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)   #指定したパッケージの読み込み
tmp <- ls(paste("package", param_txdb, sep=":"))#指定したパッケージで利用可能なオブジェ
txdb <- eval(parse(text=tmp))           #文字列tmpをRオブジェクトとしてtxdbに格納

#本番(カウントデータ取得)
count <- qCount(out, txdb, reportLevel=par
data <- count[,-1]                      #カ

#ファイルに保存
tmp <- cbind(rownames(data), data)      #保
write.table(tmp, out_f, sep="\t", append=F
```

出力ファイル(hoge6.txt)をエクセルなどで眺めなくても、出力ファイルの中身に相当するdataオブジェクトをR画面上で表示させることで全貌を把握可能。行数と列数を返すdim関数実行結果を眺めることで、カウントデータは、23,459行×6列からなることがわかる。また、最初の6行分を表示するhead関数実行結果を眺めることで、黒枠で示すUCSC known gene IDごとに妥当なカウント数を得られていることがわかる。

R Console

```
> dim(data)
[1] 23459     6
> head(data)
          Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1              407      319      312      246      273      338
10               0        0        0        1        1        0
100            178      183      182       53       91      203
1000          1451     1606     1390      630      915      599
10000         1997     2151     1604      576     1080      475
100008586        0        0        0        0        0        0
>
```

• マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) 参考

カウント情報取得(実データ)

gene-levelとexon-levelのカウントデータを一度に得ることもできます。

- 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 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- マップ後 | カウント情報取得 | トランスクリプト | BEDファイルから (last modified 2014/06/21)
- マップ後 | 配
- 正規化 | に
- 正規化 | 基

マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) NEW

QuasRパッケージを用いたsingle-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。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTranscriptDbオブジェクトをネットワ
ションの詳
「ファイル」「

1.サンプルデ

mapping_sin
2列目に「任
--strata -v 2"
て、UCSC G

```
in_f1 <-
in_f2 <-
out_f <-
param_map
param_txd
param_txd
param_rep

#必要なパ
library(Q
library(G
```

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

6と基本的に同じです。UCSC known genesのgene-levelとexon-levelの両方のカウントデータを同時に得るやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge7_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデータ)
out_f2 <- "hoge7_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデータ)
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーションパ
param_mapping <- "-m 1 --best --strata -v 2"#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実
alignmentStats(out)                     #マッピング結果(alignment statistics)の表示。seqleng

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)  #指定したパッケージの読み込み
tmp <- ls(paste("package", param_txdb, sep=":"))#指定したパッケージで利用可能なオブジェクト名
txdb <- eval(parse(text=tmp))           #文字列tmpをRオブジェクトとしてtxdbに格納

#本番(カウントデータ取得と保存; gene-level)
count <- qCount(out, txdb, reportLevel="gene")#カウントデータ行列を取得してcountに格納
```

参考

カウント情報取得(実データ)

- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 応用編 | TCC(Sun_2013) (last modified 2015/01/
- 解析 | 発現変動 | 5群間 | 対応なし | 複製あり | TCC(Sun_2013) (last modified 2014/08/22)推奨
- 解析 | 発現変動 | 時系列 | について (last modified 2014/12/19)
- 解析 | 発現変動 | 時系列 | Bayesian model-based clustering(Nascimento_2012) (last modified 201
- 解析 | 発現変動 | 時系列 | maSigPro(Nueda_2014) (last modified 2014/07/18)
- 解析 | 発現変動 | エクソン | について (last modified 2015/01/29) NEW
- 解析 | 発現変動 | エクソン | DEXseq(Anders_2012) (last modified 2014/06/23)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について (last modified 2014/12/22)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA(Wang_2014)(last modified 2014/04/
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GOseq(Young_2010) (last modified 2010/11/26
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | について (last modified 2014/12/22)
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | SeqGSEA(Wang_2014) (last modified 2014/04/04)
- 解析 | 菌叢解析 | について (last modified 2014/12/19)
- 解析 | 菌叢解析 | phyloseq(McMurdie_2013) (last modified 2014/05/29)
- 解析 | エクソーム解析 | について (last modified 2014/07/06)
- 解析 | ChIP-seq | について (last modified 2015/02/18) NEW
- 解析 | ChIP-seq | DiffBind(Ross-Innes_2012) (last modified 2014/02/04)
- 解析 | ChIP-seq | ChIPseqR(Humburg_2011) (last modified 2014/02/04)

exon-levelデータはexon-levelの発現変動解析や機能解析(SeqGSEAパッケージ)で入力データとして利用されます。

• マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis 2014)　参考

カウント情報取得（実データ）

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

6と基本的に同じです。UCSC known genesのgene-levelとexon-levelの両方のカウントデータを同時に得るやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge7_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデータ)
out_f2 <- "hoge7_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデータ)
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーションパッケージ)
param_mapping <- "-m 1 --best --strata -v 2"#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

#前処理(マッピング)
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
alignmentStats(out)                     #マッピング結果(alignment statistics)の表示。seqlength：リファ

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)  #指定したパッケージの読み込み
tmp <- ls(paste("package", param_txdb, sep=":"))#指定したパッケージで利用可能なオブジェクト名を取得した
txdb <- eval(parse(text=tmp))           #文字列tmpをRオブジェクトとしてtxdbに格納

#本番(カウントデータ取得と保存; gene-level)
count <- qCount(out, txdb, reportLevel="gene")#カウントデータ行列を取得してcountに格納
data <- count[,-1]                      #カウント情報をdataに格納
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに格納
write.table(tmp, out_f1, 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
data_gene <- data                       #dataをdata_geneに格納

#本番(カウントデータ取得と保存; exon-level)
count <- qCount(out, txdb, reportLevel="exon")#カウントデータ行列を取得してcountに格納
data <- count[,-1]                      #カウント情報をdataに格納
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに格納
write.table(tmp, out_f2, 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
data_exon <- data                       #dataをdata_exonに格納
```

gene-levelとexon-levelのカウントデータを一度に得ると決めている場合は、このようにコードの中に直接書き込んだほうがスッキリ

カウント情報取得（実データ）

30分弱かかる。exon-levelカウントデータ取得時には、ちゃんと「extracting **exon** regions …」となっていることがわかる。

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

6と基本的に同じです。UCSC known genesのgene-levelとexon-levelの両方のカウントデータを同時に得るやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge7_count_gene.txt"        #出力ファイル名を指定し
out_f2 <- "hoge7_count_exon.txt"        #出力ファイル名を指定し
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオ

#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mappin
alignmentStats(out)                     #マッピング結果(alignm

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)  #指定したパッケージの読
tmp <- ls(paste("package", param_txdb, sep=":"))#指定したパッ
txdb <- eval(parse(text=tmp))           #文字列tmpをRオブジェ

#本番(カウントデータ取得と保存; gene-level)
count <- qCount(out, txdb, reportLevel="gene")#カウントデータ
data <- count[,-1]                      #カウント情報をdataに格
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.na
data_gene <- data                       #dataをdata_geneに格納

#本番(カウントデータ取得と保存; exon-level)
count <- qCount(out, txdb, reportLevel="exon")#カウントデータ
data <- count[,-1]                      #カウント情報をdataに格
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.na
data_exon <- data                       #dataをdata_exonに格納
```

R Console

```
> #前処理(TranscriptDbオブジェクト作成)
> library(param_txdb, character.only=T)   #指定したパッケージ
> tmp <- ls(paste("package", param_txdb, sep=":"))#指
> txdb <- eval(parse(text=tmp))           #文字列tmpをRオ
>
> #本番(カウントデータ取得と保存; gene-level)
> count <- qCount(out, txdb, reportLevel="gene")#カウン
extracting gene regions from TxDb...done
counting alignments...done
collapsing counts by query name...done
> data <- count[,-1]                      #カウント情報をda
> tmp <- cbind(rownames(data), data)      #保存したい情報を
> write.table(tmp, out_f1, sep="\t", append=F, quote=
> data_gene <- data                       #dataをdata_
>
> #本番(カウントデータ取得と保存; exon-level)
> count <- qCount(out, txdb, reportLevel="exon")#カウン
extracting exon regions from TxDb...done
counting alignments...done
> data <- count[,-1]                      #カウント情報をda
> tmp <- cbind(rownames(data), data)      #保存したい情報を
> write.table(tmp, out_f2, sep="\t", append=F, quote=
> data_exon <- data                       #dataをdata_
>
```

参考

カウント情報取得（実データ）

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

6と基本的に同じです。UCSC known genesのgene-levelとexon-levelの両方のカウントデータを同時に得るやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge7_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデータ)
out_f2 <- "hoge7_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデータ)
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーションパ
param_mapping <- "-m 1 --best --strata -v 2"#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)                          #
library(GenomicAlignments)              #

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentPar
alignmentStats(out)                     #

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)   #
tmp <- ls(paste("package", param_txdb, s
txdb <- eval(parse(text=tmp))           #

#本番(カウントデータ取得と保存; gene-level)
count <- qCount(out, txdb, reportLevel="
data <- count[,-1]                      #
tmp <- cbind(rownames(data), data)      #
write.table(tmp, out_f1, sep="\t", appen
data_gene <- data                       #

#本番(カウントデータ取得と保存; exon-level)
count <- qCount(out, txdb, reportLevel="
data <- count[,-1]                      #
tmp <- cbind(rownames(data), data)      #
write.table(tmp, out_f2, sep="\t", appen
data_exon <- data                       #
```

R Console

```
> dim(data_gene)
[1] 23459     6
> head(data_gene)
          Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1              407      319      312      246      273      338
10               0        0        0        1        1        0
100            178      183      182       53       91      203
1000          1451     1606     1390      630      915      599
10000         1997     2151     1604      576     1080      475
100008586        0        0        0        0        0        0
> dim(data_exon)
[1] 289969      6
> head(data_exon)
        Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1              0        0        0        0        0        0
10             0        0        0        0        0        0
100          331      274      424      232      265      120
1000           0        0        0        1        0        0
10000         78       80       57       30       47       31
100000         0        0        0        0        0        0
>
```

出力ファイルを眺めて確認するのでもよい。hoge6.txtとhoge7_count_gene.txtは手順を間違えてなければ同じはず。exon-levelカウントデータ行列は289,969行とgene-levelデータの10倍以上の行数になっていることがわかる。

参考

カウント情報取得（実データ）

7.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

6と基本的に同じです。UCSC known genesのgene-levelとexon-levelの両方のカウントデータを同時に得るやり方

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqフ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレン
out_f1 <- "hoge7_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデ
out_f2 <- "hoge7_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデ
param_txdb <- "TxDb.Hsapiens.UCSC.hg19.knownGene"#パッケージ名を指定(TxDB系のアノテーションパッケージ)
param_mapping <- "-m 1 --best --strata -v 2"#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)
library(GenomicAlignments)

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentP
alignmentStats(out)

#前処理(TranscriptDbオブジェクト作成)
library(param_txdb, character.only=T)
tmp <- ls(paste("package", param_txdb,
txdb <- eval(parse(text=tmp))

#本番(カウントデータ取得と保存; gene-leve
count <- qCount(out, txdb, reportLevel
data <- count[,-1]
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f1, sep="\t", app
data_gene <- data

#本番(カウントデータ取得と保存; exon-leve
count <- qCount(out, txdb, reportLevel
data <- count[,-1]
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f2, sep="\t", app
data_exon <- data
```

行列countは、count[行, 列]で特定の行や列へのアクセスを可能とする。この場合コンマの右側に数値が入っているので、列に対して処理を行っている。count[, −1]は、1列目の情報を削除せよ、という意味。

R Console

```
> dim(count)
[1] 289969      7
> head(count)
       width Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1        354        0        0        0        0        0        0
10       192        0        0        0        0        0        0
100      225      331      274      424      232      265      120
1000      35        0        0        0        1        0        0
10000    156       78       80       57       30       47       31
100000   133        0        0        0        0        0        0
> dim(data)
[1] 289969      6
> head(data)
       Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1             0        0        0        0        0        0
10            0        0        0        0        0        0
100         331      274      424      232      265      120
1000          0        0        0        1        0        0
10000        78       80       57       30       47       31
100000        0        0        0        0        0        0
>
```

カウント情報取得(実データ)

TxDbパッケージで提供されていない、Ensembl gene IDでのカウント情報を得ることもできます

- 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 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- マップ後 | カウント情報取得 | リファレンスゲノム | BEDファイルから (last modified 2014/06/21)
- マップ後 | 配
- 正規化 | に
- 正規化 | 基

マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) NEW

QuasRパッケージを用いたsingle-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。アノテーション情報は、GenomicFeaturesパッケージ中の関数を利用してTranscriptDbオブジェクトをネットワ
ションの詳
「ファイル」「ラ

1.サンプルデー

mapping_sin
2列目に「任意
--strata -v 2"
て、UCSC G

```
in_f1 <-
in_f2 <-
out_f <-
param_map
param_txd
param_txd
param_rep

#必要なパ
library(Q
library(G
```

9.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

4と8を組み合わせたようなやり方です。数値だけのUCSC gene IDではなくEnsembl Gene IDで取り扱うやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge9_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデータ)
out_f2 <- "hoge9_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデータ)
out_f3 <- "hoge9_QC.pdf"                #出力ファイル名を指定してout_f3に格納(QCファイル)
param_txdb1 <- "hg19"                   #TranscriptDbオブジェクト作成用のリファレンスゲノムを
param_txdb2 <- "ensGene"                #TranscriptDbオブジェクト作成用のtable名を指定(「sup
param_mapping <- "-m 1 --best --strata -v 2"#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実
alignmentStats(out)                     #マッピング結果(alignment statistics)の表示。seqleng
qQCReport(out, pdfFilename=out_f3)      #QCレポート結果をファイルに保存

#前処理(TranscriptDbオブジェクト作成)
txdb <- makeTranscriptDbFromUCSC(genome=param_txdb1, tablename=param_txdb2)#TranscriptDb
txdb                                    #確認してるだけです
```

• マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis 2014)

9.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

4と8を組み合わせたようなやり方です。数値だけのUCSC gene IDではなくEnsembl Gene IDで取り扱うやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンス配
out_f1 <- "hoge9_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデー
out_f2 <- "hoge9_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデー
out_f3 <- "hoge9_QC.pdf"                #出力ファイル名を指定してout_f3に格納(QCファイル)
param_txdb1 <- "hg19"                   #TranscriptDbオブジェクト作成用のリファレンスゲノ
param_txdb2 <- "ensGene"                #TranscriptDbオブジェクト作成用のtable名を指定(「
param_mapping <- "-m 1 --best --strata -v 2"#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

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実
alignmentStats(out)                     #マッピング結果(alignment statistics)の表示。seqleng
qQCReport(out, pdfFilename=out_f3)      #QCレポート結果をファイルに保存

#前処理(TranscriptDbオブジェクト作成)
txdb <- makeTranscriptDbFromUCSC(genome=param_txdb1, tablename=param_txdb2)#TranscriptDb
txdb                                    #確認してるだけです

#本番(カウントデータ取得と保存; gene-level)
count <- qCount(out, txdb, reportLevel="gene")#カウントデータ行列を取得してcountに格納
data <- count[,-1]                      #カウント情報をdataに格納
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファ
data_gene <- data                       #dataをdata_geneに格納

#本番(カウントデータ取得と保存; exon-level)
count <- qCount(out, txdb, reportLevel="exon")#カウントデータ行列を取得してcountに格納
data <- count[,-1]                      #カウント情報をdataに格納
tmp <- cbind(rownames(data), data)      #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファ
data_exon <- data                       #dataをdata_exonに格納
```

GenomicFeaturesパッケージで提供されているmakeTranscriptDbFromUCSC関数を利用しています。ネットワーク経由で取得するので、最新のアノテーション情報を得ることができます。txdbオブジェクトを得れば、以降は同じ手順でできます。

カウント情報取得(完全データ名)

9.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

4と8を組み合わせたようなやり方です。数値だけのUCSC gene IDではなくEnsembl Gene IDで取り扱うやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #入力ファイル名を指定してin_f2に格納(リファレンス配
out_f1 <- "hoge9_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデー
out_f2 <- "hoge9_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデー
out_f3 <- "hoge9_QC.pdf"                #出力ファイル名を指定してout_f3に格納(QCファイル)
param_txdb1 <- "hg19"                   #TranscriptDbオブジェクト作成用のリファレンスゲノ
param_txdb2 <- "ensGene"                #TranscriptDbオブジェクト作成用のtable名を指定(「
param_mapping <- "-m 1 --best --strata -v 2"#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)
library(GenomicFeatures)

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alignmentP
alignmentStats(out)
qQCReport(out, pdfFilename=out_f3)

#前処理(TranscriptDbオブジェクト作成)
txdb <- makeTranscriptDbFromUCSC(genom
txdb

#本番(カウントデータ取得と保存; gene-lev
count <- qCount(out, txdb, reportLevel
data <- count[,-1]
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f1, sep="\t", app
data_gene <- data

#本番(カウントデータ取得と保存; exon-lev
count <- qCount(out, txdb, reportLevel
data <- count[,-1]
tmp <- cbind(rownames(data), data)
write.table(tmp, out_f2, sep="\t", app
data_exon <- data
```

作成されたtxdbオブジェクトの中身。Creation timeやGenomicFeatures, RSQLiteのバージョンなどを書きとめておいたほうがよいだろう。遺伝子数情報はここからはわからないが、exon数は584,914個であることがわかる。

R Console

```
> txdb                                  #確認してるだけです
TxDb object:
| Db type: TxDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Organism: Homo sapiens
| UCSC Table: ensGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Ensembl gene ID
| Full dataset: yes
| miRBase build ID: NA
| transcript_nrow: 204940
| exon_nrow: 584914
| cds_nrow: 280379
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2015-02-25 21:55:51 +0900 (Wed, 25 Feb 2015)
| GenomicFeatures version at creation time: 1.18.3
| RSQLite version at creation time: 1.0.0
| DBSCHEMAVERSION: 1.0
>
```

カウント情報取得(実データ)-2

9.srp017142_samplename.txt中のFASTQファイル群をBSgenome.Hsapiens.UCSC.hg19にマッピングする場合:

4と8を組み合わせたようなやり方です。数値だけのUCSC gene IDではなくEnsembl Gene IDで取り扱うやり方です。

```
in_f1 <- "srp017142_samplename.txt"     #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "BSgenome.Hsapiens.UCSC.hg19" #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)
out_f1 <- "hoge9_count_gene.txt"        #出力ファイル名を指定してout_f1に格納(カウントデータ)
out_f2 <- "hoge9_count_exon.txt"        #出力ファイル名を指定してout_f2に格納(カウントデータ)
out_f3 <- "hoge9_QC.pdf"                #出力ファイル名を指定してout_f3に格納(QCファイル)
param_txdb1 <- "hg19"                   #TranscriptDbオブジェクト作成用のリファレンスゲノムを
param_txdb2 <- "ensGene"                #TranscriptDbオブジェクト作成用のtable名を指定(「sup
param_mapping <- "-m 1 --best --strata -v 2"#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)
library(GenomicFeatures)

#前処理(マッピング)
out <- qAlign(in_f1, in_f2, alig
alignmentStats(out)
qQCReport(out, pdfFilename=out_f

#前処理(TranscriptDbオブジェクト作
txdb <- makeTranscriptDbFromUCSC
txdb

#本番(カウントデータ取得と保存; ge
count <- qCount(out, txdb, repor
data <- count[,-1]
tmp <- cbind(rownames(data), dat
write.table(tmp, out_f1, sep="\t
data_gene <- data

#本番(カウントデータ取得と保存; ex
count <- qCount(out, txdb, repor
data <- count[,-1]
tmp <- cbind(rownames(data), dat
write.table(tmp, out_f2, sep="\t
data_exon <- data
```

R Console

```
> dim(data_gene)
[1] 60234     6
> head(data_gene)
                 Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
ENSG00000000003      480      513      366      124      271      366
ENSG00000000005        0        0        0        1        0        0
ENSG00000000419      282      354      208      165      301      209
ENSG00000000457      201      254      183      166      296      148
ENSG00000000460      114      112      101       55       81       59
ENSG00000000938        0        0        0        2        2        1
> dim(data_exon)
[1] 584914      6
> head(data_exon)
       Pro_rep1 Pro_rep2 Pro_rep3 Ras_rep1 Ras_rep2 Ras_rep3
1             0        0        0        0        0        0
10            0        0        0        0        0        0
100           0        0        0        0        0        0
1000          0        0        0        0        0        0
10000         0        0        0        0        0        0
100000      417      385      529      646      794      314
>
```

gene-levelカウントデータの行名がちゃんとEnsembl gene IDになっていることがわかる。gene数は60,234個。exon数はtxdbオブジェクト中の数値と同じ584,914個であることがわかる。

カウント情報取得雑感

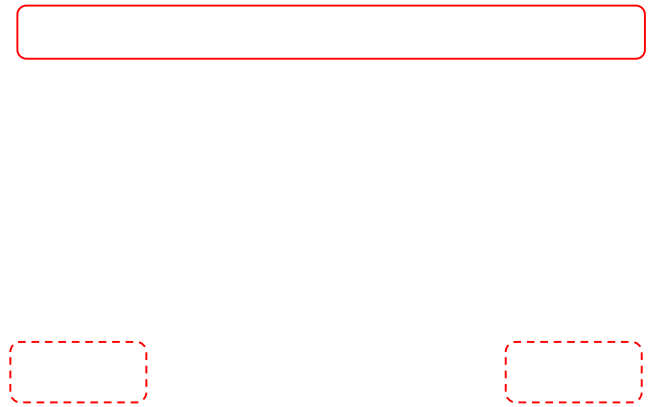
カウント情報を得る際に、どのexonの情報を用いるか?

- 全体的な流れとしてはgene-level → exon-level (or isoform-level)
  - 例：新規splice variantの発見（Twine et al., *PLoS One*, **6**: e16266, 2011）
- 遺伝子セット解析（Gene Ontology解析やパスウェイ解析など）のためのこれまでに蓄積されてきた知識は、遺伝子レベルの解像度
- 複数エクソン　→　遺伝子レベルの要約統計量
  - exon union method (Mortazavi et al., *Nat. Methods*, **5**: 621-628, 2008)
    - **全て**のisoforms間で用いられているexonの情報（**union**：和集合）を利用
  - exon intersection method (Bullard et al., *BMC Bioinformatics*, **11**: 94, 2010)
    - 複数isoforms間で**共通**して用いられているexonの情報のみ（**intersection**：積集合）を利用

カウント情報取得雑感

様々な思想があり、当然その後の解析結果に影響を及ぼします

- 算出された生リードカウント結果
  - exon union method（和集合）の場合：20リード
  - exon intersection method（積集合）の場合：11リード



「Garber et al., *Nat. Methods*, **8**: 469-477, 2011」のFig. 3c

カウント情報取得雑感

- マップ後 | 出力ファイルの読み込み | Bowtie形式 (last modified 2013/06/18)
- マップ後 | 出力ファイルの読み込み | SOAP形式 (last modified 2013/06/19)
- マップ後 | 出力ファイルの読み込み | htS Tools(Planet 2012) (last modified 2013/06/19)
- マップ後 | カウント情報取得 | について modified 2014/12/17)
- マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis 2014) (last modified 2015/02/26)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis 2014) (last modified 2014/06/22)
- 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 | 配列長とカウント数の関係 (last modified 2014/07/03)
- 正規化 | について (last modified
- 正規化 | 基礎 | RPK or CPK (配
- 正規化 | 基礎 | RPM or CPM (総

マップ後 | カウント情報取得 | について

BAM形式などのマッピング結果ファイルからカウントデータを取得するためのパッケージや関数もいくつかあります。htSeqToolsのislandCounts関数、Rsubread(Windows版はなし)のfeatureCounts関数、GenomicRangesのsummarizeOverlaps関数などです。QuasRは、内部的にGenomicRangesのsummarizeOverlaps関数を利用しています。

Rパッケージ:

- htSeqTools: Planet et al., Bioinformatics., 2012
- Rsubread: Liao et al., Nucleic Acids Res., 2013
- GenomicRanges: Lawrence et al., PLoS Comput. Biol., 2013
- QuasR: Gaidatzis et al., Bioinformatics, 2014

R以外:

- HTSeq: Anders et al., Bioinformatics, 2014
- Rcount: Schmid and Grossniklaus, Bioinformatics, 2014

R以外では、HTSeqがわりとよく使われているようです。Rcountは、カウント時の優先順位をつけたり、flanking regionを含めたりいろいろできるようです。

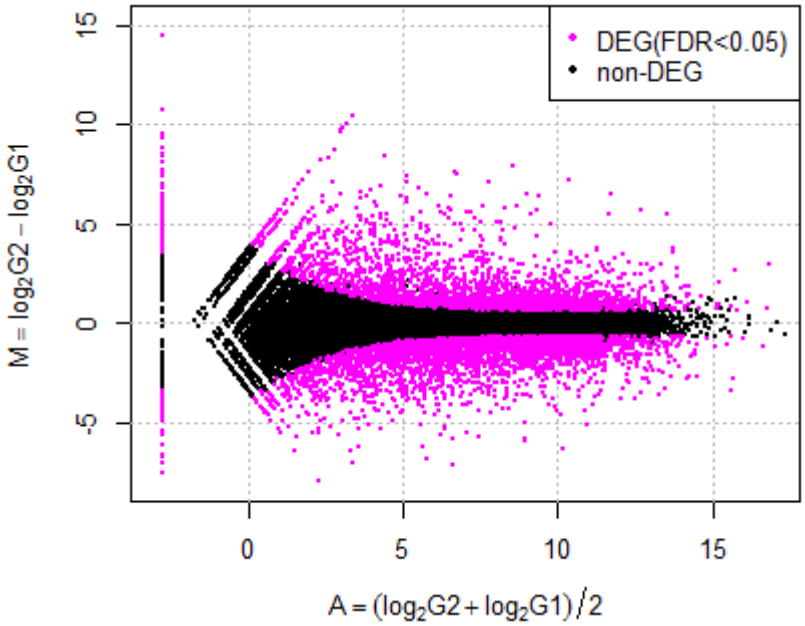
カウント情報取得以降の解析

カウントデータ取得以降のデータ解析のイメージ。これは複製あり2群間比較用ヒトRNA-seqデータ

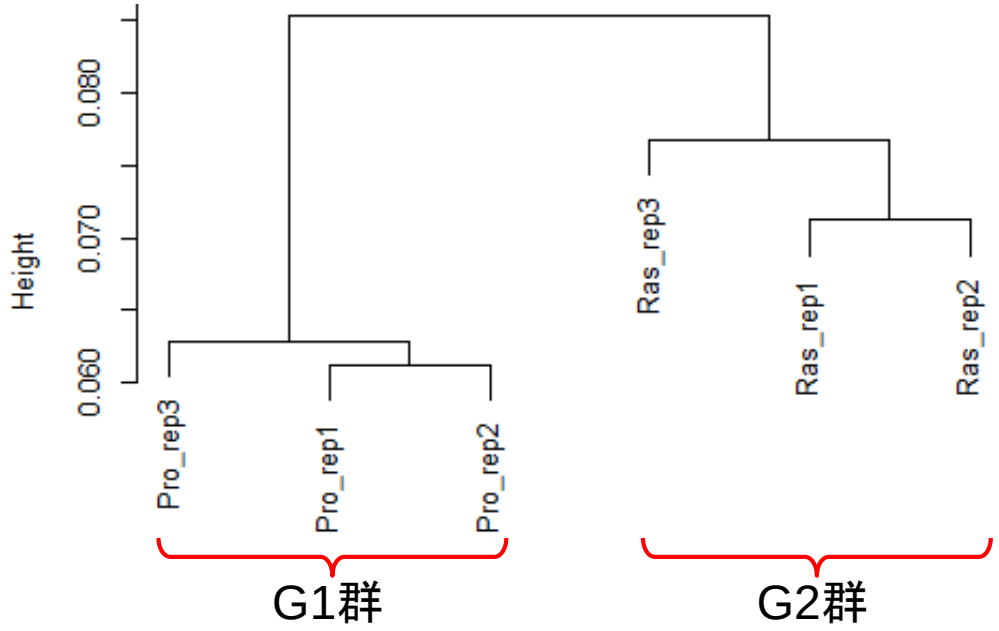
カウントデータ

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

発現変動遺伝子（DEG）同定



サンプル間クラスタリング



Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

参考

クラスタリング

ゼロカウントを含む低発現データのフィルタリングは重要です。入力ファイルは「デスクトップ − hoge − clustering」にあります。

- 解析 | 新規転写物同定(ゲノム配列を利用) (last modified 2014/07/24)
- 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 2014/06/30)
- 解析 | クラスタリング | サンプル間 | TCC(Sun 2013) (last modified 2015/02/26) NEW
- 解析 | クラスタリング | 遺伝子間 | ...
- 解析 | シミュレーション | ...
- 解析 | シミュレーション | ...
- 解析 | シミュレーション | ...

解析 | クラスタリング | サンプル間 | hclust

RNA-seqカウントデータのクラスタリング結果は、特にゼロカウント(0カウント; zero count)を多く含む場合に(もちろん距離の定義の仕方によっても変わってきますが) 低発現データのフィルタリングの閾値次第で結果が変わる傾向にあります。ここでは、上記閾値問題に悩まされることなく頑健なサンプル間クラスタリングを行うやり方を示します。内部的に行っていることは、1)全サンプルで0カウントとなる行(遺伝子)をフィルタリングした後、unique関数を用いて同一発現パターンのものを1つのパターンとしてまとめ、2)「1 - Spearman順位相関係数」でサンプル間距離を定義、3)Average-linkage clusteringの実行、です。順位相関係数を用いてサンプルベクトル間の類似度として定義するので、サンプル間正規化の問題に悩まされません。また、低発現遺伝子にありがちな同一発現パターンの遺伝子をまとめることで、(変動しやすい)同順位となる大量の遺伝子が集約されるため、結果的に「総カウント数がx個以下のものをフィルタリング...」という閾値問題をクリアしたことになります。

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

1. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1〜gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001〜gene_10000までがnon-DEGであることが既知です。

```
in_f <- "data_hypodata_3vs3.txt"
out_f <- "hoge1.png"
param_fig <- c(500, 400)

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

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)
data <- unique(data[obj,])
dim(data)

#本番
data.dist <- as.dist(1 - cor(data, method="spearman"))#サンプル間の距離を計算し、結果をdata.distに格納
```

#入力ファイル名を指定してin_fに格納

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/clustering"
> list.files()
[1] "data_hypodata_3vs3.txt"
[2] "hoge9_count_exon.txt"
[3] "hoge9_count_gene.txt"
>
```

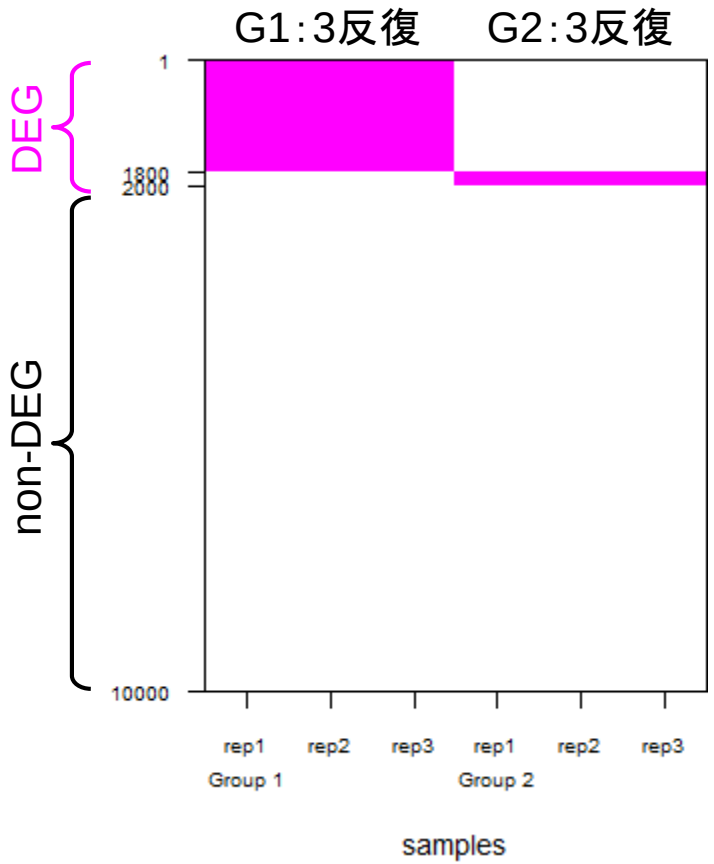
クラスタリング

DEGが多く存在するほど群間で明瞭なクラスターに分かれる傾向。→クラスタリング結果からDEGの有無をある程度把握可能です

- data_hypodata_3vs3.txt（2群間比較用）
  - 全部で10,000行×6列。最初の2,000行分が発現変動遺伝子（DEG）

DEG
G1で高発現
G2で高発現
non-DEG

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	36	56	144	2	1	0
gene_2	84	152	124	52	37	28
gene_3	592	840	800	151	257	200
gene_4	0	8	4	1	1	3
gene_5	32	32	0	1	1	0
…						
gene_1801	34	86	24	284	180	364
gene_1802	5	1	3	0	160	24
gene_1803	57	56	51	248	192	220
gene_1804	29	25	32	128	204	160
gene_1805	42	29	44	184	156	92
…						
gene_2001	4	8	9	13	12	4
gene_2002	88	139	40	22	44	21
gene_2003	933	667	462	889	396	443
gene_2004	48	37	14	36	57	71
gene_2005	290	338	553	319	210	504
…						
gene_9996	107	67	104	35	65	45
gene_9997	145	220	120	80	95	156
gene_9998	42	73	67	62	44	37
gene_9999	5	1	2	3	4	11
gene_10000	2	4	5	2	0	0



参考

クラスタリング

クラスタリング結果ファイルがちゃんと作成されている。

1. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG(最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。

```
in_f <- "data_hypodata_3vs3.txt"          #入力ファイル名を指
out_f <- "hoge1.png"                      #出力ファイル名を指
param_fig <- c(500, 400)                  #ファイル出力時の横

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="
dim(data)                                 #オブジェクトdataの

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)      #条件を満たすかどう
data <- unique(data[obj,])                #objがTRUEとなる行
dim(data)                                 #オブジェクトdataの

#本番
data.dist <- as.dist(1 - cor(data, method="spearman"))#サ
out <- hclust(data.dist, method="average")#階層的クラスタ
png(out_f, pointsize=13, width=param_fig[1], height=para
plot(out)                                 #樹形図（デンドログ
dev.off()                                 #おまじない
```

R Console

```
> #前処理(フィルタリング)
> obj <- as.logical(rowSums(data) > 0)    #条件$
> data <- unique(data[obj,])              #obj$
> dim(data)                               #オブ$
[1] 9391    6
>
> #本番
> data.dist <- as.dist(1 - cor(data, method="s$
> out <- hclust(data.dist, method="average")#$
> png(out_f, pointsize=13, width=param_fig[1],$
> plot(out)                                   #樹形$
> dev.off()                                   #おま$
null device
          1
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/clustering"
> list.files()
[1] "data_hypodata_3vs3.txt"
[2] "hoge1.png"
[3] "hoge9_count_exon.txt"
[4] "hoge9_count_gene.txt"
>
```

参考

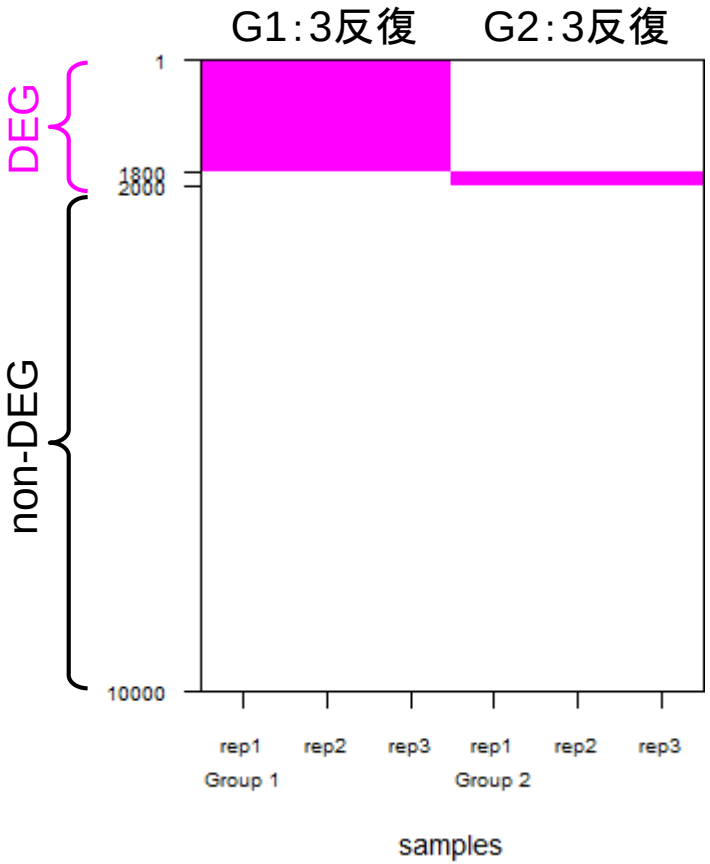
クラスタリング

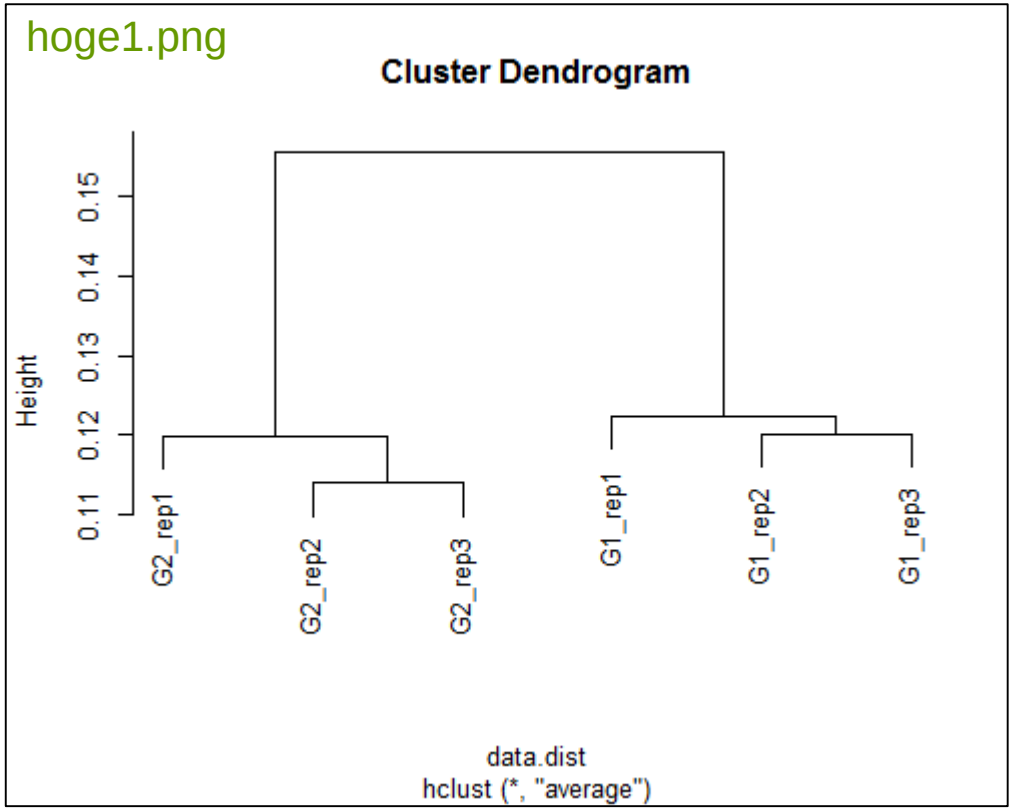
DEGが多く存在するほど群間で明瞭なクラスターに分かれる傾向、の意味がわかると思います

1. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG(最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。

```
in_f <- "data_hypodata_3vs3.txt"        #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.png"                    #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
```





参考

DEGが存在しないデータのステレオタイプなクラスタリング結果のイメージを示します。

クラスタリング

- 解析 | 発現量推定(トランスクリプトーム配列を利用) (last modified 2014/07/09)
- 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 2015/02/26) NEW
- 解析 | クラスタリング | 遺伝子間 | MBCluster.Seq (Si 2014) (last modified 2014/02/05)
- 解析 | シミュレ
- 解析 | シミュレ

解析 | クラスタリング | サンプル間 | hclust

RNA-seqカウントデータのクラスタリング結果は、特にゼロカウント(0カウント; zero count)を多く含む場合に(もちろん
によっても変
悩まされるこ
る行(遺伝子)を
位相関係数」でサ
度として定義する
まとめることで、(
グ...」という閾値問
「ファイル」-「ディレ

1. サンプルデータ

Biological replica
初の1800個がG

```
in_f <- "dat
out_f <- "ho
param_fig <-

#入力ファイル
data <- read
dim(data)
```

2. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。non-DEGデータのみでクラスタリングを行っています。

```
in_f <- "data_hypodata_3vs3.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.png"                      #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                  #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_nonDEG <- 2001:10000                #non-DEGの位置を指定

#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
dim(data)                                 #オブジェクトdataの行数と列数を表示

#前処理(サブセットの抽出)
data <- data[param_nonDEG,]               #指定した行のみ抽出した結果をdataに格納
dim(data)                                 #オブジェクトdataの行数と列数を表示

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)      #条件を満たすかどうかを判定した結果をobjに格納
data <- unique(data[obj,])                #objがTRUEとなる行のみ抽出し、ユニークパターンの
dim(data)                                 #オブジェクトdataの行数と列数を表示

#本番
```

参考

クラスタリング

non-DEGデータのみ(2001行目以降)でクラスタリングを行っていることがわかります。

2. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。no

```
in_f <- "data_hypodata_3vs3.txt"          #
out_f <- "hoge2.png"                      #
param_fig <- c(500, 400)                  #
param_nonDEG <- 2001:10000                #

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

#前処理(サブセットの抽出)
data <- data[param_nonDEG,]               #
dim(data)                                 #

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)      #
data <- unique(data[obj,])                #
dim(data)                                 #

#本番
data.dist <- as.dist(1 - cor(data, metho
out <- hclust(data.dist, method="average
png(out_f, pointsize=13, width=param_fig
```

R Console

```
> obj <- as.logical(rowSums(data) > 0)     #条件を満たすかど$
> data <- unique(data[obj,])               #objがTRUEとなる$
> dim(data)                                #オブジェクトdata$
[1] 7493    6
>
> #本番
> data.dist <- as.dist(1 - cor(data, method="spearman"))#$
> out <- hclust(data.dist, method="average")#階層的クラス$
> png(out_f, pointsize=13, width=param_fig[1], height=para$
> plot(out)                                #樹形図(デンドロ$
> dev.off()                                #おまじない
null device
          1
> head(data)
          G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_2001       4       8       9      13      12       4
gene_2002      88     139      40      22      44      21
gene_2003     933     667     462     889     396     443
gene_2004      48      37      14      36      57      71
gene_2005     290     338     553     319     210     504
gene_2006      18      25      16      27      20      51
>
```

参考

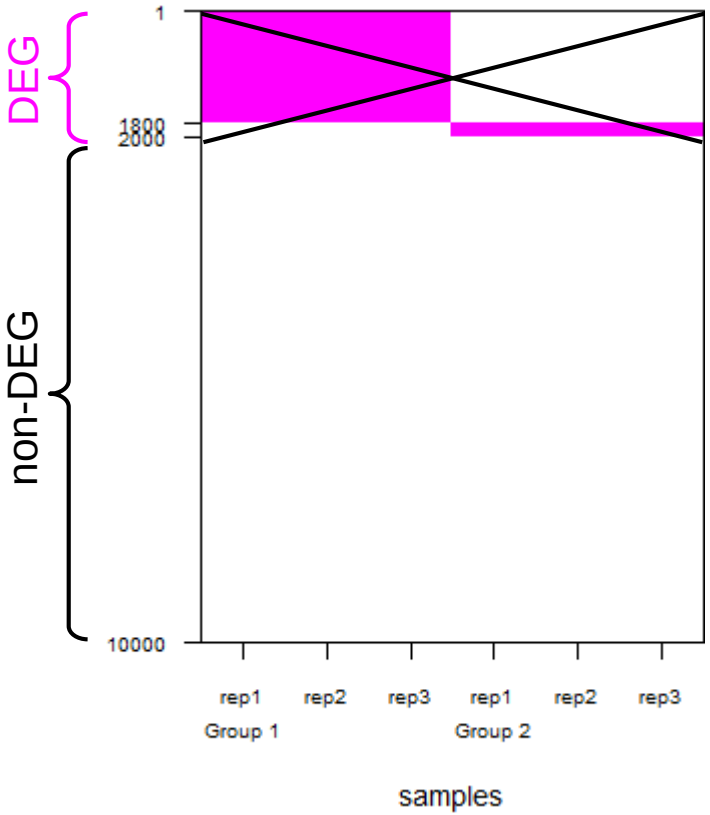
クラスタリング

DEGが存在しないデータのステレオタイプなクラスタリング結果のイメージです。G1サンプルとG2サンプルが入り混じっていればDEGはないか、あっても非常に少ない傾向となります。

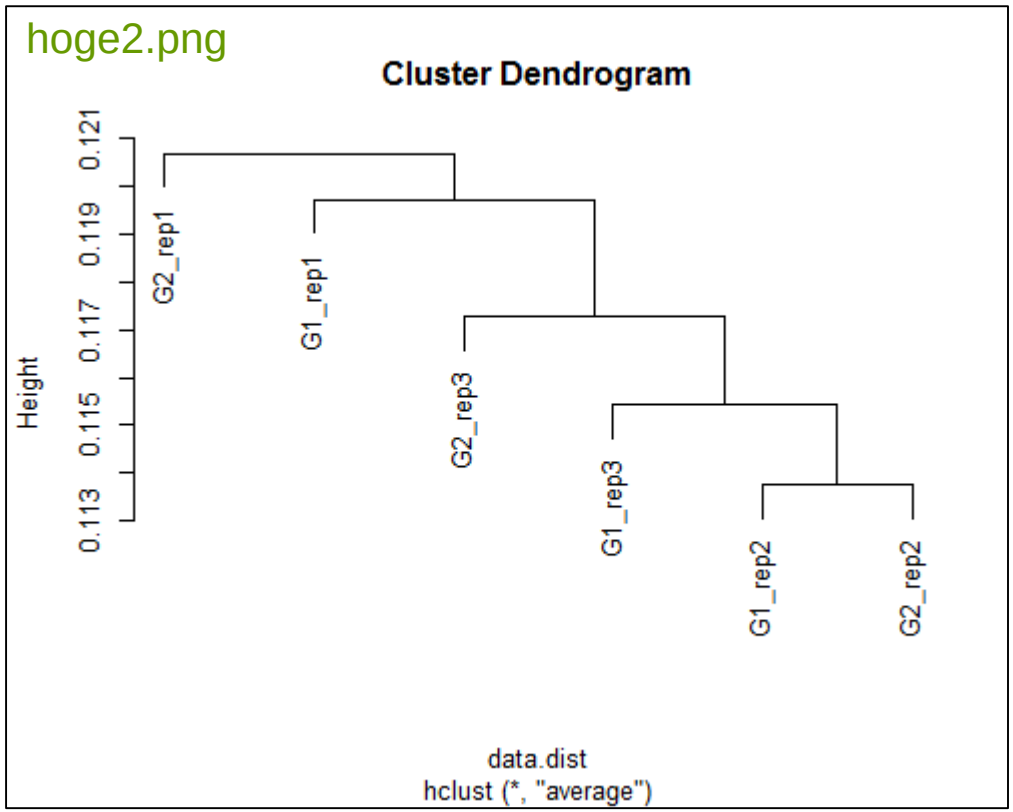
2. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。non-DEGデータのみでクラスタリングを行っています。

```
in_f <- "data_hypodata_3vs3.txt"        #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.png"                    #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_nonDEG <- 2001:10000              #non-DEGの位置を指定
```



hoge2.png



クラスタリング

リアルデータ(gene-levelカウントデータ)のクラスタリングを行う。R本の中にも解説あり(3.3.3節)。TCCパッケージでclusterSample関数を提供。入力ファイルは「デスクトップ – hoge – clustering」にもあります。

- 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 2015/02/26) NEW
- 解析 | クラスタリング | 遺伝子間 | MBCluster.Seq (Si…14) (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…結果を返します。

「ファイル」-「ディレ…

1. 59,857 genes×6 samp…

Neyret-Kahn et al., Ge…
RNA-seqカウントデー…
SRP017142(Neyret-K…

```
in_f <- "srp017…
out_f <- "hoge1…
param_fig <- c(…

#必要なパッケージ…
library(TCC)

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

5. 60,234 genes×6 samplesのリアルデータ(hoge9_count_gene.txt)の場合:

Neyret-Kahn et al., Genome Res., 2013のgene-levelの2群間比較用(3 proliferative samples vs. 3 Ras samples)ヒトRNA-seqカウントデータです。マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014)から得られます。

```
in_f <- "hoge9_count_gene.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.png"                    #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                #ファイル出力時の横幅と縦幅を指定(単位は

#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
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
            hclust.method="average", unique.pattern=TRUE)#クラスタリング実行結

#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
par(mar=c(0, 4, 1, 0))                  #下、左、上、右の順で余白（行）を指定
plot(out, sub="", xlab="", cex.lab=1.2,#樹形図(デンドログラム)の表示
  cex=1.3, main="", ylab="Height")      #樹形図(デンドログラム)の表示
dev.off()                               #おまじない
```

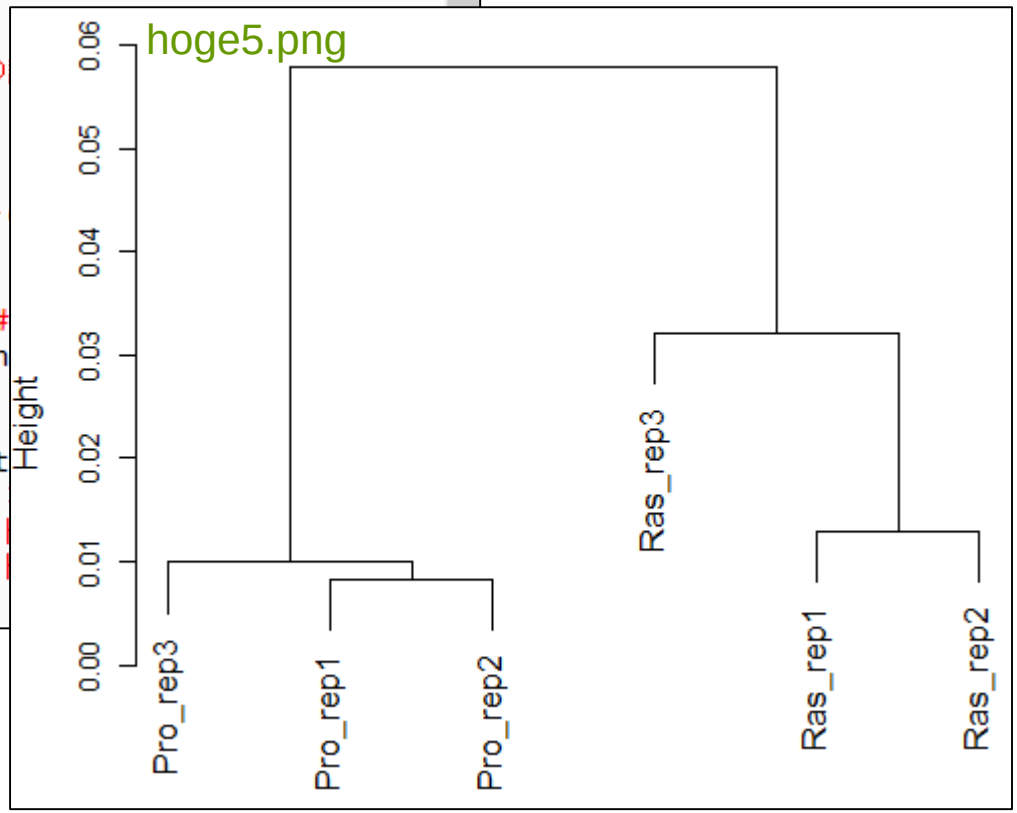
クラスタリング

5. 60,234 genes×6 samplesのリアルデータ(hoge9_count_gene.txt)の場合:

Neyret-Kahn et al., Genome Res., 2013のgene-levelの2群間比較用(3 proliferative samples vs. 3 Ra
samples)ヒトRNA-seqカウントデータです。マップ後 | カウント情報取得 | ゲノム | アノテーション有 |
QuasR(Gaidatzis 2014)から得られます。

```
in_f <- "hoge9_count_gene.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.png"                    #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                #ファイル出力時の横幅と縦幅を指定(単位は

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1,
dim(data)                               #オブジェクト

#本番
out <- clusterSample(data, dist.method="spearman",#
            hclust.method="average", unique.pattern

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height
par(mar=c(0, 4, 1, 0))                  #下、左、上、
plot(out, sub="", xlab="", cex.lab=1.2,#樹形図(デン
  cex=1.3, main="", ylab="Height")      #樹形図(デン
dev.off()                               #おまじない
```

Pro群(G1群)とRas群(G2群)できれいに分かれていることから、このデータセット中には多くの発現変動遺伝子(DEG)が存在することが予想されます。



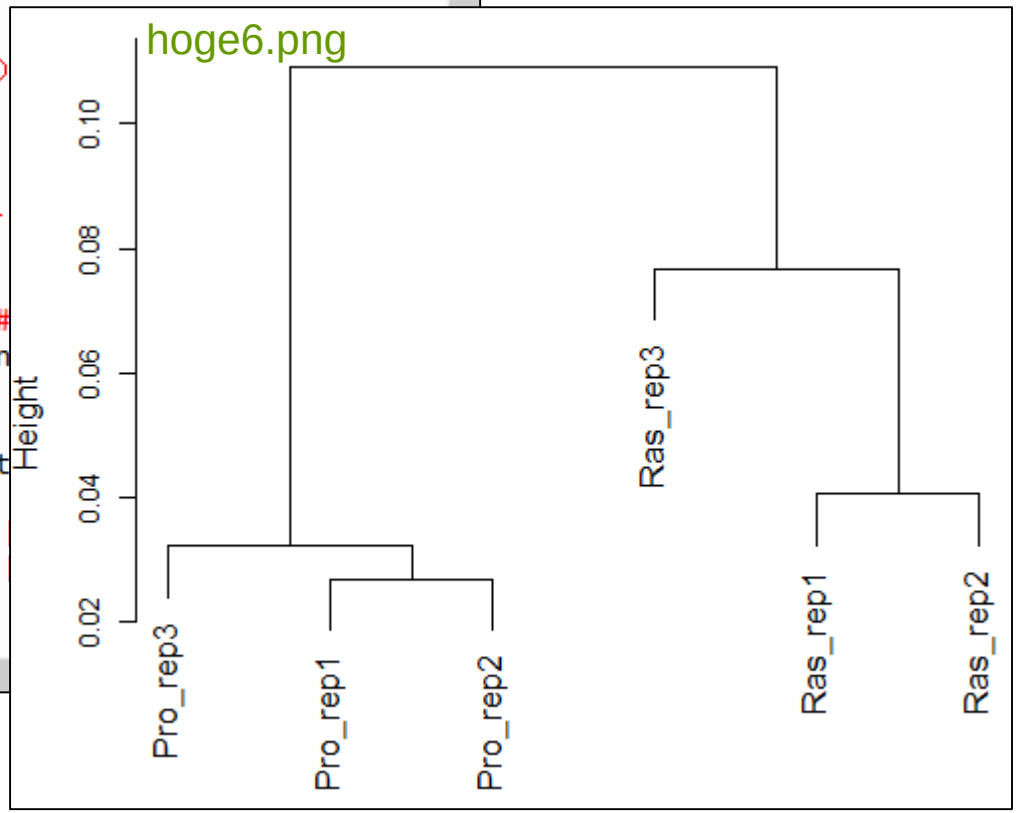
クラスタリング

exon-levelカウントデータでもクラスタリング結果の全体的な傾向は不変であることがわかる。

6. 584,914 exons×6 samplesのリアルデータ(hoge9_count_exon.txt)の場合:

Neyret-Kahn et al., Genome Res., 2013のexon-levelの2群間比較用(3 proliferative samples vs. 3 Ras samples)ヒトRNA-seqカウントデータです。マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014)から得られます。

```
in_f <- "hoge9_count_exon.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge6.png"                    #出力ファイル名を指定してout_fに格納
param_fig <- c(500, 400)                #ファイル出力時の横幅と縦幅を指定(単位は

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1,
dim(data)                               #オブジェクト

#本番
out <- clusterSample(data, dist.method="spearman",#
            hclust.method="average", unique.pattern

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height
par(mar=c(0, 4, 1, 0))                  #下、左、上、
plot(out, sub="", xlab="", cex.lab=1.2,#樹形図(デン
  cex=1.3, main="", ylab="Height")      #樹形図(デン
dev.off()                               #おまじない
```



Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

発現変動解析

2014年7月2日の大学院講義資料とほぼ同じです。入力はカウントデータ。出力は発現変動遺伝子(DEG)検出結果

- 解析 | シミュレーションカウントデータ | Biological rep. | 3群間 | 基礎編 | TCC(Sun_2013) (last mod
- 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 2015/2/02) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun_2013) (last modified 2014/03/03)推奨
- 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 | 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exact testで発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeR...という解析パイプラインに相当します。全てTCCパッケージ(Sun et al., BMC Bioinformatics, 2013)内で完結します。

「ファ...

1. サンプ...

Biologic...
でがDE...
あること...

```
in_f
out_f
out_f
param
param
param
param

#必要
libra
```

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ごろと若干古いだけ)です。パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn_2013)から得られます。

```
in_f <- "srp017142_count_bowtie.txt"    #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge9.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge9.png"                   #出力ファイル名を指定してout_f2に格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2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(430, 390)                #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

#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))#G1群を1、G2群を2としたベクトルdata.clを作成
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)
```

発現変動解析

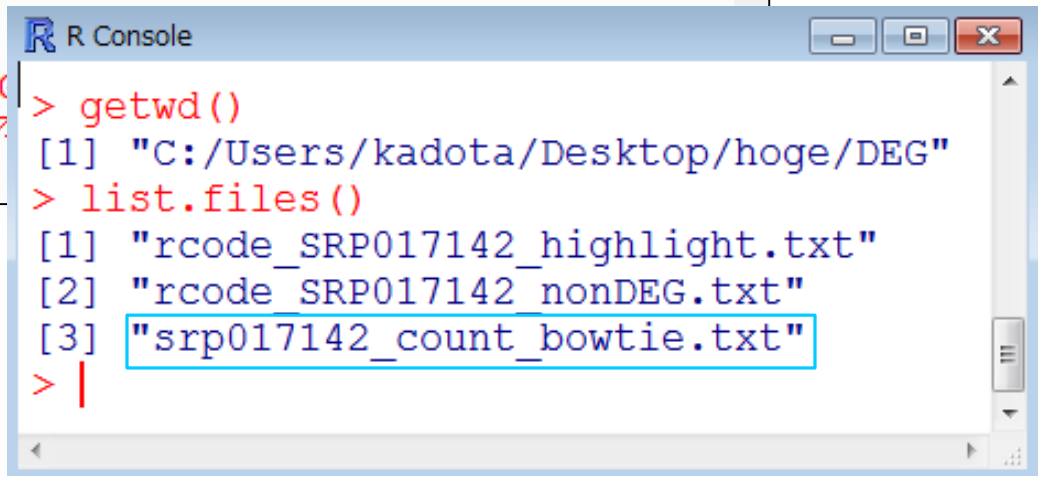
リアルデータ(gene-levelカウントデータ)の発現変動解析を行う。入力ファイルは「デスクトップ – hoge – DEG」にもあります。R本のp145-157に説明あり。

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ごろと若干古いだけ)です。パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2013)から得られます。

```
in_f <- "srp017142_count_bowtie.txt"      #入力
out_f1 <- "hoge9.txt"                     #出力
out_f2 <- "hoge9.png"                     #出力
param_G1 <- 3                             #G1群
param_G2 <- 3                             #G2群
param_FDR <- 0.05                         #DEG検
param_fig <- c(430, 390)                  #ファ

#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

#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))#C
tcc <- new("TCC", data, data.cl)          #TCCクラスオ

#本番(正規化)
```

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						



発現変動解析

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ご
す。パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2(

```
in_f <- "srp017142_count_bowtie.txt"    #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge9.txt"                   #出力ファイル名を指定してout_f1に格
out_f2 <- "hoge9.png"                   #出力ファイル名を指定してout_f2に格
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2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(430, 390)                #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

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

#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, par
tcc <- new("TCC", data, data.cl)        #T

#本番(正規化)
```

5%偽物を含むのを許容するとDEG数は5,669個。20%の偽物混入を許容すると8,110 DEGs。FDR閾値が30%の場合は9,151個がDEGと判定される。このデータセット中に存在する本物のDEGは9,151×0.7 = 6,405.7個程度だと判断できる。

R Console

```
> dev.off()                             #おまじない
null device
          1
> sum(tcc$stat$q.value < 0.05)          #FDR < 0.05$
[1] 5669
> sum(tcc$stat$q.value < 0.10)          #FDR < 0.10$
[1] 6680
> sum(tcc$stat$q.value < 0.20)          #FDR < 0.20$
[1] 8110
> sum(tcc$stat$q.value < 0.30)          #FDR < 0.30$
[1] 9151
>
```

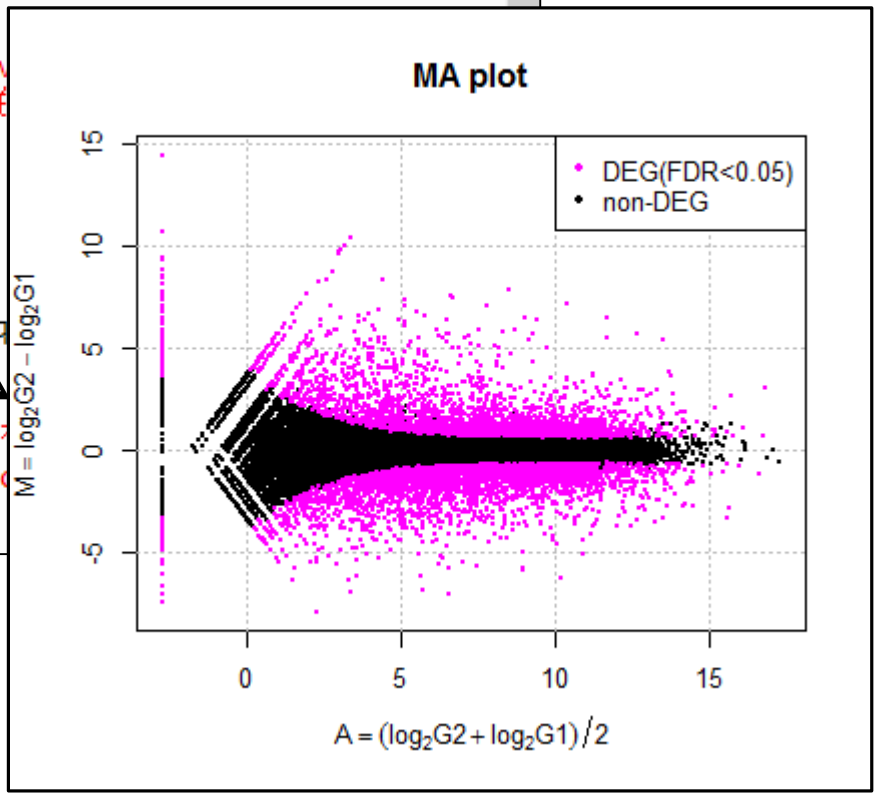
発現変動解析

これがいわゆる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。

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ごろと若干古いだけ)です。パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2013)から得られます。

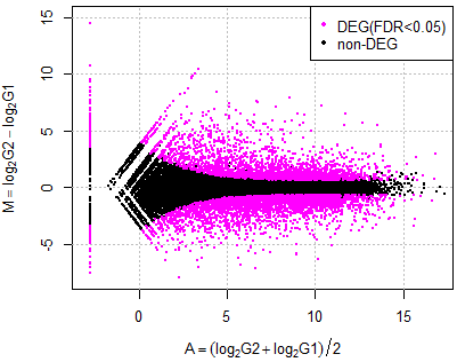
```
in_f <- "srp017142_count_bowtie.txt"    #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge9.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge9.png"                   #出力ファイル名を指定してout_f2に格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2群のサンプル数を指定
param_FDR <- 0.05                       #DEG検出時のfalse discov
param_fig <- c(430, 390)                #ファイル出力時の横幅と縦

#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", q

#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))#G1群を1、G2群
tcc <- new("TCC", data, data.cl)        #TCCクラスオブジェクトtc

#本番(正規化)
```

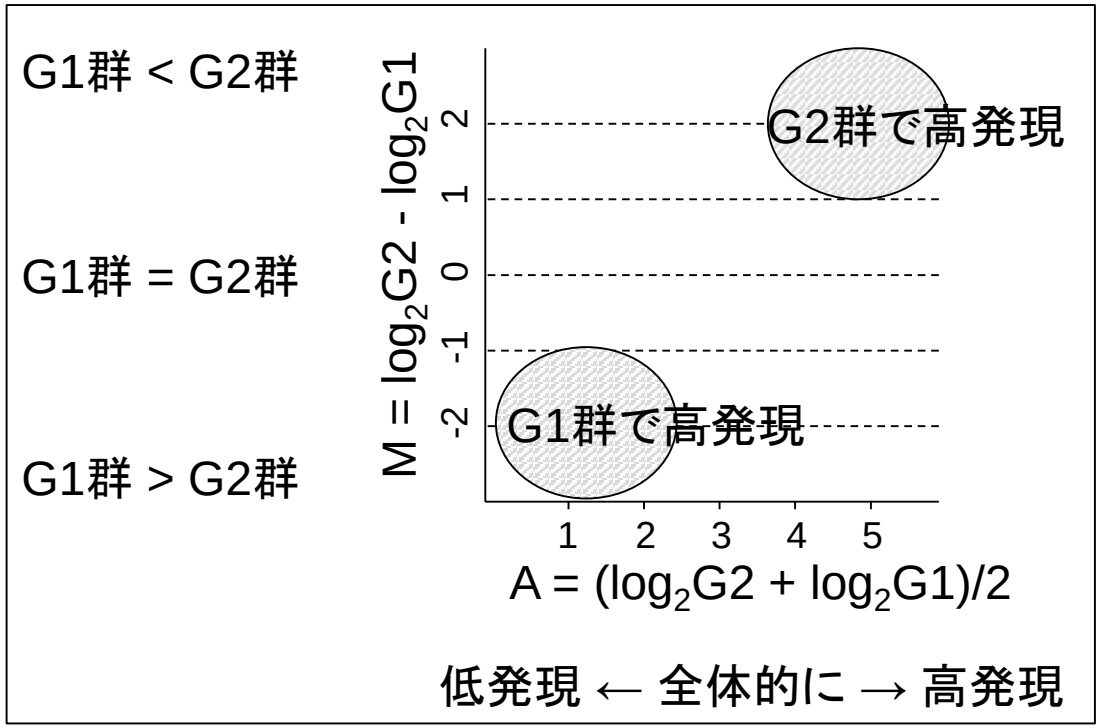


M-A plot

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。

- 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）





Dudoit et al., *Stat. Sinica*, **12**: 111-139, 2002

基本的にはこれが解析結果。

発現変動解析

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ごろと若干古いだけ)です。パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2013)から得られます。

```
in_f <- "srp017142_count_bowtie.txt"    #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge9.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge9.png"                   #出力ファイル名を指定してout_f2に格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2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(430, 390)                #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
```

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1
ENSG00000163431	3716.6	3244.5	4185.2	70.1	78.8	49.0	ENSG00000163431	8.95	−5.82	2.90E−80	3.48E−76	5	1
ENSG00000204291	1215.5	1236.8	1339.3	22.4	27.8	21.5	ENSG00000204291	7.44	−5.72	3.45E−78	3.44E−74	6	1
ENSG00000181634	107.0	158.6	66.5	12842.0	16014.1	19820.5	ENSG00000181634	10.39	7.20	1.04E−76	8.88E−73	7	1
ENSG00000178726	53.1	46.3	62.5	6006.1	5567.6	3166.0	ENSG00000178726	9.01	6.51	2.39E−74	1.79E−70	8	1
ENSG00000117600	576.9	518.9	602.6	7.0	5.6	2.4	ENSG00000117600	5.73	−6.83	1.14E−72	7.59E−69	9	1
ENSG00000158050	7.7	9.1	11.1	552.3	536.6	523.5	ENSG00000158050	6.14	5.85	1.16E−70	6.91E−67	10	1
ENSG00000124126	50.5	44.6	53.4	1819.4	1683.2	1413.9	ENSG00000124126	8.15	5.05	5.35E−69	2.91E−65	11	1

1位はRas群（G2群）で高発現のDEG

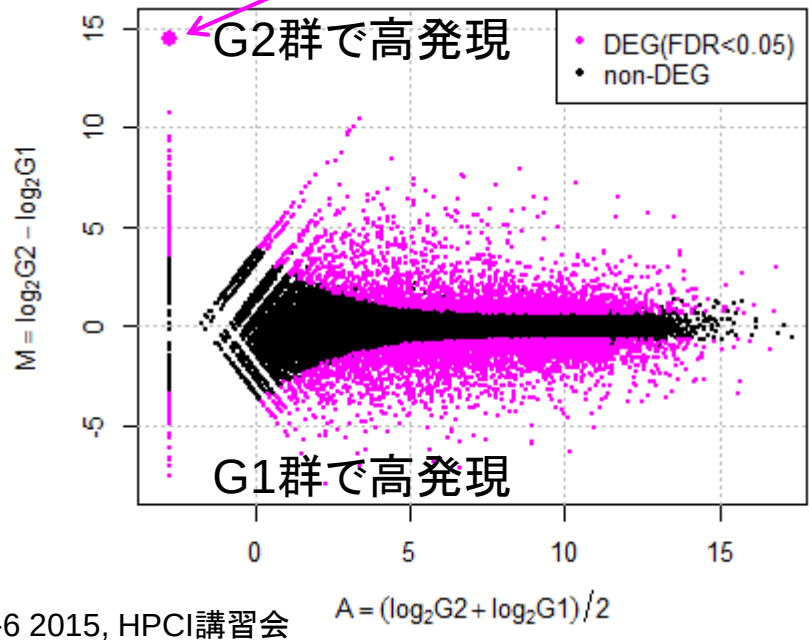
発現変動遺伝子検出結果

hoge9.txt

G1群　G2群

***p*-valueとその順位**

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1
ENSG00000163431	3716.6	3244.5	4185.2	70.1	78.8	49.0	ENSG00000163431	8.95	−5.82	2.90E−80	3.48E−76	5	1
ENSG00000204291	1215.5	1236.8	1339.3	22.4	27.8	21.5	ENSG00000204291	7.44	−5.72	3.45E−78	3.44E−74	6	1
ENSG00000181634	107.0	158.6	66.5	12842.0	16014.1	19820.5	ENSG00000181634	10.39	7.20	1.04E−76	8.88E−73	7	1
ENSG00000178726	53.1	46.3	62.5	6006.1	5567.6	3166.0	ENSG00000178726	9.01	6.51	2.39E−74	1.79E−70	8	1
ENSG00000117600	576.9	518.9	602.6	7.0	5.6	2.4	ENSG00000117600	5.73	−6.83	1.14E−72	7.59E−69	9	1
ENSG00000158050	7.7	9.1	11.1	552.3	536.6	523.5	ENSG00000158050	6.14	5.85	1.16E−70	6.91E−67	10	1
ENSG00000124126	50.5	44.6	53.4	1819.4	1683.2	1413.9	ENSG00000124126	8.15	5.05	5.35E−69	2.91E−65	11	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。

2位もRas群（G2群）で高発現のDEG

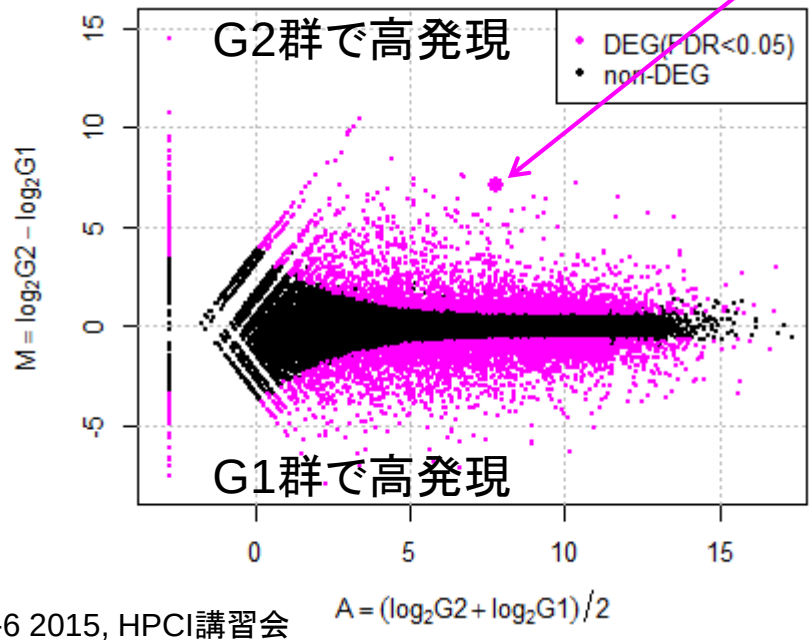
発現変動遺伝子検出結果

hoge9.txt

G1群　G2群

***p*-valueとその順位**

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1
ENSG00000163431	3716.6	3244.5	4185.2	70.1	78.8	49.0	ENSG00000163431	8.95	−5.82	2.90E−80	3.48E−76	5	1
ENSG00000204291	1215.5	1236.8	1339.3	22.4	27.8	21.5	ENSG00000204291	7.44	−5.72	3.45E−78	3.44E−74	6	1
ENSG00000181634	107.0	158.6	66.5	12842.0	16014.1	19820.5	ENSG00000181634	10.39	7.20	1.04E−76	8.88E−73	7	1
ENSG00000178726	53.1	46.3	62.5	6006.1	5567.6	3166.0	ENSG00000178726	9.01	6.51	2.39E−74	1.79E−70	8	1
ENSG00000117600	576.9	518.9	602.6	7.0	5.6	2.4	ENSG00000117600	5.73	−6.83	1.14E−72	7.59E−69	9	1
ENSG00000158050	7.7	9.1	11.1	552.3	536.6	523.5	ENSG00000158050	6.14	5.85	1.16E−70	6.91E−67	10	1
ENSG00000124126	50.5	44.6	53.4	1819.4	1683.2	1413.9	ENSG00000124126	8.15	5.05	5.35E−69	2.91E−65	11	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。

3,4位もRas群（G2群）で高発現のDEG

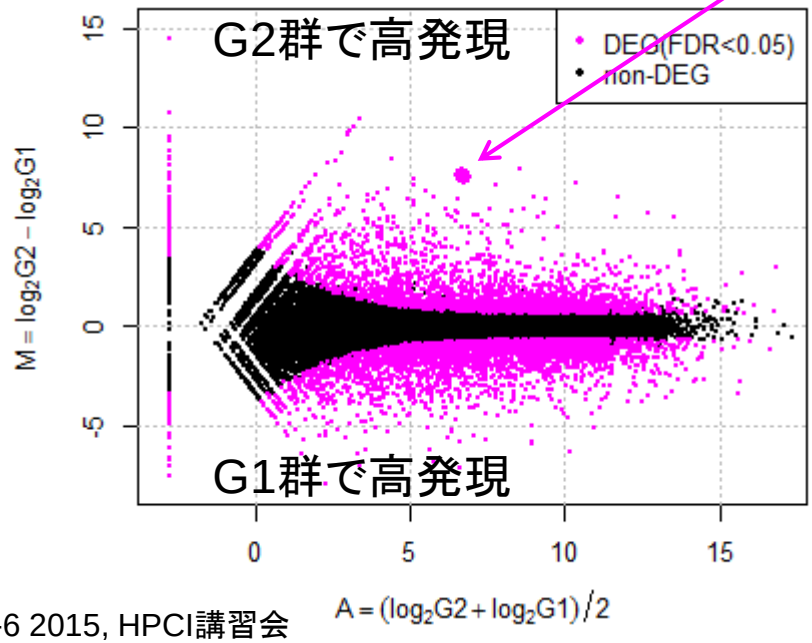
発現変動遺伝子検出結果

hoge9.txt

G1群　G2群

***p*-valueとその順位**

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1
ENSG00000163431	3716.6	3244.5	4185.2	70.1	78.8	49.0	ENSG00000163431	8.95	−5.82	2.90E−80	3.48E−76	5	1
ENSG00000204291	1215.5	1236.8	1339.3	22.4	27.8	21.5	ENSG00000204291	7.44	−5.72	3.45E−78	3.44E−74	6	1
ENSG00000181634	107.0	158.6	66.5	12842.0	16014.1	19820.5	ENSG00000181634	10.39	7.20	1.04E−76	8.88E−73	7	1
ENSG00000178726	53.1	46.3	62.5	6006.1	5567.6	3166.0	ENSG00000178726	9.01	6.51	2.39E−74	1.79E−70	8	1
ENSG00000117600	576.9	518.9	602.6	7.0	5.6	2.4	ENSG00000117600	5.73	−6.83	1.14E−72	7.59E−69	9	1
ENSG00000158050	7.7	9.1	11.1	552.3	536.6	523.5	ENSG00000158050	6.14	5.85	1.16E−70	6.91E−67	10	1
ENSG00000124126	50.5	44.6	53.4	1819.4	1683.2	1413.9	ENSG00000124126	8.15	5.05	5.35E−69	2.91E−65	11	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。

5位はPro群（G1群）で高発現のDEG

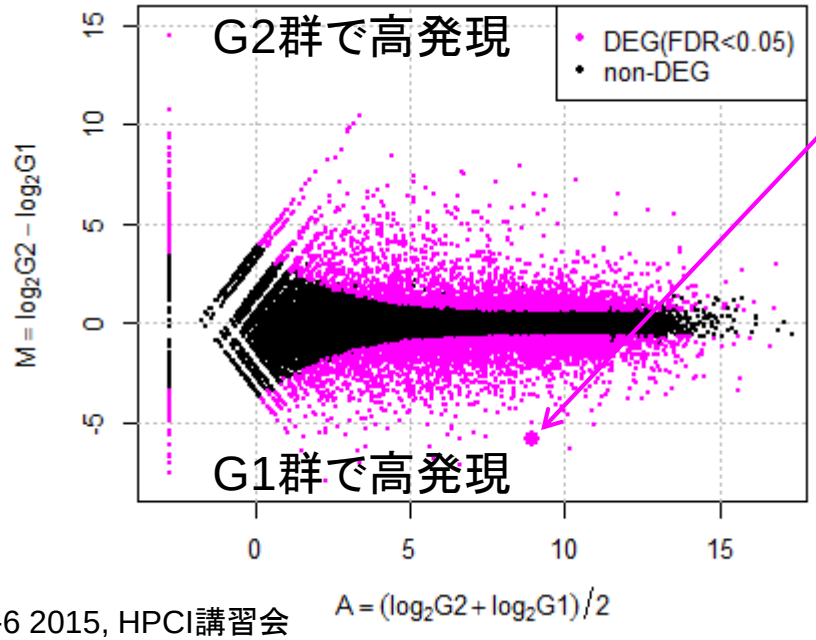
発現変動遺伝子検出結果

hoge9.txt

G1群　G2群

***p*-valueとその順位**

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1
ENSG00000163431	3716.6	3244.5	4185.2	70.1	78.8	49.0	ENSG00000163431	8.95	−5.82	2.90E−80	3.48E−76	5	1
ENSG00000204291	1215.5	1236.8	1339.3	22.4	27.8	21.5	ENSG00000204291	7.44	−5.72	3.45E−78	3.44E−74	6	1
ENSG00000181634	107.0	158.6	66.5	12842.0	16014.1	19820.5	ENSG00000181634	10.39	7.20	1.04E−76	8.88E−73	7	1
ENSG00000178726	53.1	46.3	62.5	6006.1	5567.6	3166.0	ENSG00000178726	9.01	6.51	2.39E−74	1.79E−70	8	1
ENSG00000117600	576.9	518.9	602.6	7.0	5.6	2.4	ENSG00000117600	5.73	−6.83	1.14E−72	7.59E−69	9	1
ENSG00000158050	7.7	9.1	11.1	552.3	536.6	523.5	ENSG00000158050	6.14	5.85	1.16E−70	6.91E−67	10	1
ENSG00000124126	50.5	44.6	53.4	1819.4	1683.2	1413.9	ENSG00000124126	8.15	5.05	5.35E−69	2.91E−65	11	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。

発現変動遺伝子検出結果

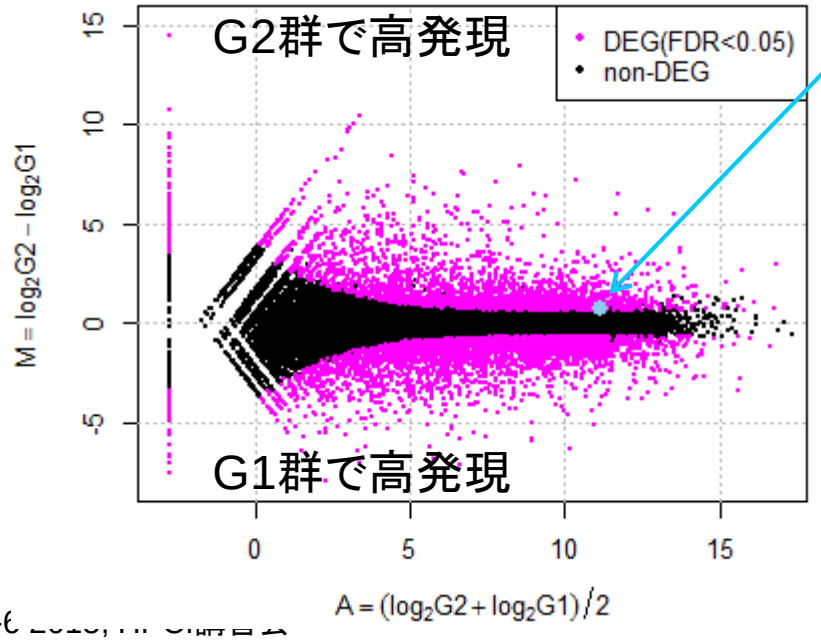
指定したFDR閾値（0.05）をギリギリ満たす5,669位の遺伝子

***p*-valueとその順位**

G1群　G2群

hoge9.txt

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000148848	286.8	327.2	262.0	486.4	475.5	419.5	ENSG00000148848	8.52	0.66	0.004726	0.049922	5666	1
ENSG00000186603	16.3	13.2	10.1	23.8	16.7	69.3	ENSG00000186603	4.46	1.47	0.004727	0.049927	5667	1
ENSG00000168556	161.8	142.9	146.1	218.7	257.7	236.6	ENSG00000168556	7.56	0.66	0.004729	0.049936	5668	1
ENSG00000189159	1794.1	1668.1	1774.6	2377.2	2307.9	4183.1	ENSG00000189159	11.15	0.76	0.004731	0.049954	5669	1
ENSG00000177096	621.4	575.0	600.6	322.4	317.0	468.5	ENSG00000177096	8.88	−0.70	0.004739	0.050031	5670	0
ENSG00000103148	1707.7	1452.5	1820.0	2347.8	2142.0	4082.7	ENSG00000103148	11.09	0.78	0.004746	0.050088	5671	0
ENSG00000156011	918.5	1103.8	882.8	605.5	685.9	271.3	ENSG00000156011	9.47	−0.89	0.00475	0.050127	5672	0
ENSG00000089818	472.5	544.5	478.7	685.4	845.3	815.1	ENSG00000089818	9.29	0.65	0.004751	0.050127	5673	0
ENSG00000160007	4551.2	4256.6	4650.7	3080.9	3115.1	1459.3	ENSG00000160007	11.72	−0.81	0.004752	0.05013	5674	0
ENSG00000105778	900.5	1027.8	984.6	1529.2	1904.7	1239.4	ENSG00000105778	10.26	0.68	0.004765	0.050255	5675	0
ENSG00000246451	46.2	91.7	73.6	46.3	33.4	29.9	ENSG00000246451	5.66	−0.95	0.004771	0.050317	5676	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。

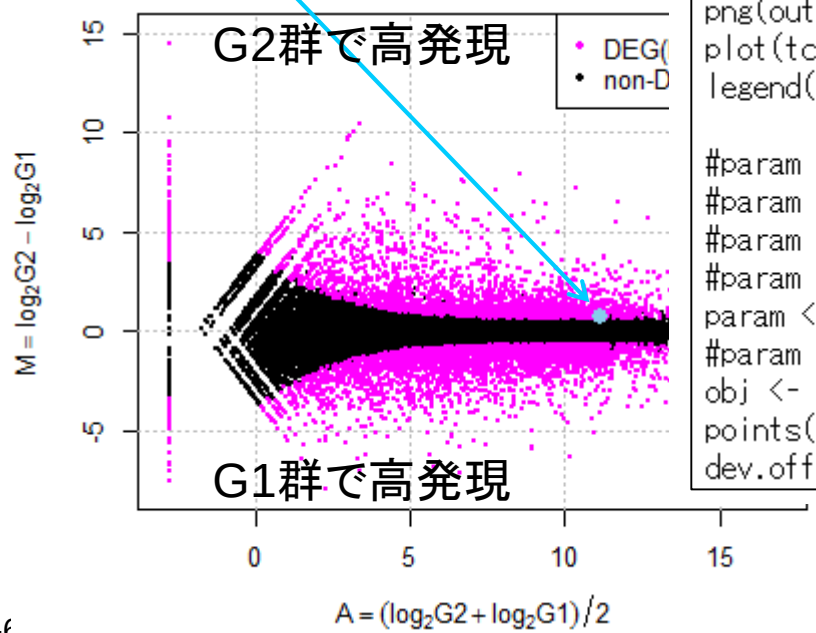
発現変動遺伝子検出結果

ハイライトさせたいGene IDの位置情報を論理値ベクトルobjとして取得後、points関数を用いてobjがTRUEとなる要素のみ、pch, cex, colオプションを駆使して追加で描画している。

hoge9.txt

G1群　G2群

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000148848	286.8	327.2	262.0	486.4	475.5	419.5	ENSG00000148848	8.52	0.66	0.004726	0.049922	5666	1
ENSG00000186603	16.3	13.2	10.1	23.8	16.7	69.3	ENSG00000186603	4.46	1.47	0.004727	0.049927	5667	1
ENSG00000168556	161.8	142.9	146.1	218.7	257.7	236.6	ENSG00000168556	7.56	0.66	0.004729	0.049936	5668	1
ENSG00000189159	1794.1	1668.1	1774.6	2377.2	2307.9	4183.1	ENSG00000189159	11.15	0.76	0.004731	0.049954	5669	1
ENSG00000177096	621.4	575.0	600.6	322.4	317.0	468.5	ENSG00000177096	8.88	−0.70	0.004739	0.050031	5670	0
ENSG00000103148	1707.7	1452.5	1820.0	2347.8	2142.0	4082.7	ENSG00000103148	11.09	0.78	0.004746	0.050088	5671	0
ENSG00000156011	918.5	1103.8	882.8	605.5	685.9	271.3	ENSG00000156011	9.47	−0.89	0.00475	0.050127	5672	0
ENSG00000089818	472.5	544.5	478.7	685.4	845.3	815.1	ENSG00000089818	9.29	0.65	0.004751	0.050127	5673	0
ENSG00000160007	4551.2	4256.6	4650.7	3080.9									
ENSG00000105778	900.5	1027.8	984.6	1529.2									
ENSG00000246451	46.2	91.7	73.6	46.3									



rcode_SRP017142_highlight.txt(の一部)

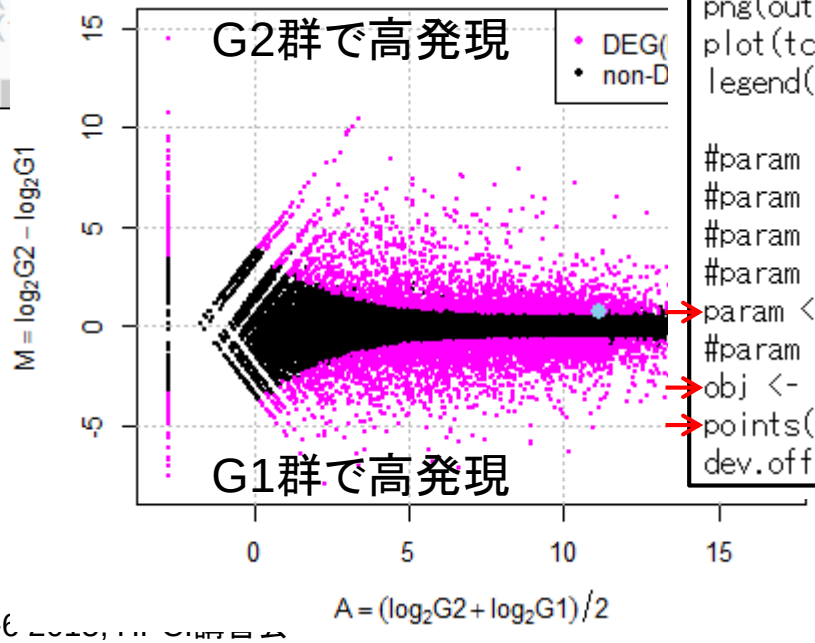
```
#ファイルに保存(M-A plot)↓
png(out_f8, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, FDR=param_FDR, xlim=c(-3, 17), ylim=c(-8, 15))#param_FDRで指定した閾
legend("topright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG"),#凡
       col=c("magenta", "black"), pch=20)#凡例を作成している↓
#param <- "ENSG00000240386"                          # 1位↓
#param <- "ENSG00000128564"                          # 2位↓
#param <- c("ENSG00000188064", "ENSG00000101188") # 3,4位↓
#param <- "ENSG00000163431"                          # 5位↓
param <- "ENSG00000189159"                          # 5669位 skyblue↓
#param <- c("ENSG00000166359", "ENSG00000146676") # 24461, 24462位 skyblue↓
obj <- is.element(tcc$gene_id, param)↓
points(result$a.value[obj], result$m.value[obj], pch=20, cex=2, col="skyblue")
dev.off()                                           #おまじない↓
```

9. 59,857 genes×6 samplesのリアルデータ(srp017142_count_bowtie.txt)の場合:

8.の入力ファイル(hoge9_count_gene.txt)と本質的に同じもの(アノテーション情報が2014年3月ごろと若干古いだけ)です。パイプライン|ゲノム|発現変動|2群間|対応なし|複製あり|SRP017142(Neyret-Kahn_2013)から得られます。

左側のテンプレートスクリプトとの違いは赤矢印部分のみ

```
#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値などの結果を抽出してを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$count), normalized, result)#正規化後のデータの右側にDEG検出結果を
tmp <- tmp[order(tmp$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の中身を指定したファ

#ファイルに保存(M-A plot)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメ
plot(tcc, FDR=param_FDR, xlim=c(-3, 17), ylim=c(-8, 15))#param_FDRで指定した閾値を満たすDEG
legend("topright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG"),#凡例を作成して
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                               #おまじ
sum(tcc$stat$q.value < 0.05)            #FDR <
sum(tcc$stat$q.value < 0.10)            #FDR <
```



rcode_SRP017142_highlight.txt(の一部)

```
#ファイルに保存(M-A plot)↓
png(out_f8, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, FDR=param_FDR, xlim=c(-3, 17), ylim=c(-8, 15))#param_FDRで指定した閾
legend("topright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG"),#凡
       col=c("magenta", "black"), pch=20)#凡例を作成している↓
#param <- "ENSG00000240386"                              # 1位↓
#param <- "ENSG00000128564"                              # 2位↓
#param <- c("ENSG00000188064", "ENSG00000101188")        # 3,4位↓
#param <- "ENSG00000163431"                              # 5位↓
param <- "ENSG00000189159"                               # 5669位 skyblue↓
#param <- c("ENSG00000166359", "ENSG00000146676")        # 24461, 24462位 skyblue↓
obj <- is.element(tcc$gene_id, param)↓
points(result$a.value[obj], result$m.value[obj], pch=20, cex=2, col="skyblue")
dev.off()                                                #おまじない↓
```

作図

M-A plotはまだ放置したままですが、「クラスタリング」と「ROC曲線」については、順を追ってプロット時のオプションをいろいろ変えた例題を用意しています。これらが参考になると思います

- 解析 | 制限酵素切断部位(RECS)地図 | REDseq(Zhu 201X) (last modified 2011/12
- 解析 | small RNA | segmentSeq(Hardcastle 2012) (last modified 2014/02/04)
- 作図 | について (last modified 2012/09/10)
- 作図 | M-A plot | 基本編 (last modified 2012/10/01)
- 作図 | M-A plot | ggplot2編 (last modified 2013/07/30)
- 作図 | クラスタリング | サンプル間 | TCC(Sun 2013) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 1. 感覚をつかむ (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 2. 色を自在に変える(col) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 3. 形や大きさを変える(cex, lwd, lty) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 4. 軸ラベルやタイトルを消す(ann, axes) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 5. 軸ラベルの表示角度を変える(las) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 6. 余白を変える(mar) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 7. 図の重ね書き(new) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 基礎編 | 8. 凡例を追加(legend) (last modified 2015/02/15) NEW
- 作図 | ROC曲線 | 応用編 (last modified 2015/02/07) NEW
- 作図 | SplicingGraphs (last modified 2013/08/07)
- パイプライン | について (last modified 2013/10/17)
- パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2013) (last modified 2015/01/21)
- パイプライン | ゲノム | 機能解析 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn 2013) (last modified 2015/01/27)
- パイプライン | ゲノム | 機能解析 | 2群間 | 対応なし | 複製あり | SRP011435(Huang 2012) (last modified 2015/01/21)
- パイプライン | ゲノム | small RNA | SRP016842(Nie 2013) (last modified 2014/06/21)
- リンク集 (last modified 2012/03/29)

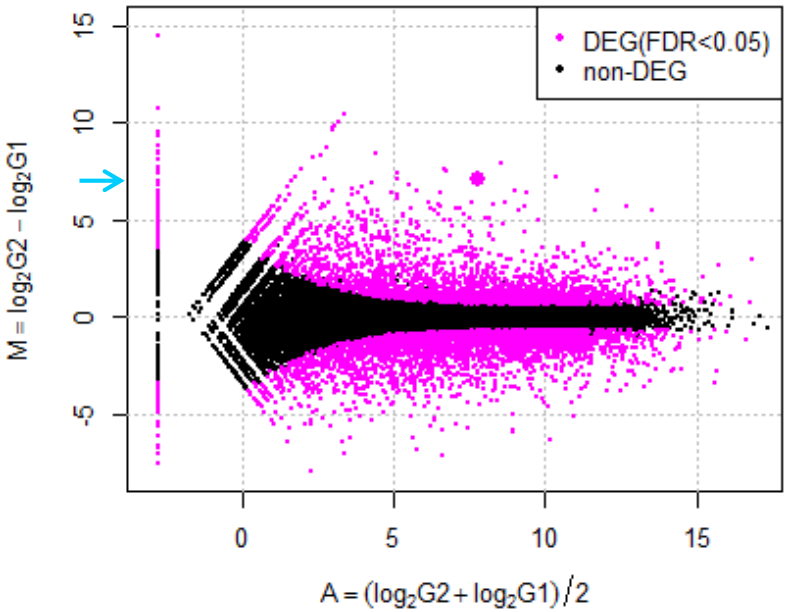
Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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**
  - 機能解析、遺伝子セット解析(SeqGSEA)

分布やモデルのイントロ

- TCCパッケージを用いたDEG同定

M-A plotのM値は倍率変化(log比)に相当($2^{7.11}$ 倍G2群で高発現)



G1群 G2群

59,857 genes

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

rownames(tcc$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	-2.78	14.48	1.75E-139	1.04E-134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E-107	1.21E-102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E-98	5.33E-94	3	1
ENSG00000101188	6.0	5.0						5	7.55	3.81E-97	5.70E-93	4	1
ENSG00000163431	3716.6	3244.5						5	-5.82	2.90E-80	3.48E-76	5	1
ENSG00000204291	1215.5	1236.8						4	-5.72	3.45E-78	3.44E-74	6	1
ENSG00000181634	107.0	158.6						9	7.20	1.04E-76	8.88E-73	7	1
ENSG00000178726	53.1	46.3						1	6.51	2.39E-74	1.79E-70	8	1
ENSG00000117600	576.9	518.9						3	-6.83	1.14E-72	7.59E-69	9	1
ENSG00000158050	7.7	9.1						4	5.85	1.16E-70	6.91E-67	10	1
ENSG00000124126	50.5	44.6						5	5.05	5.35E-69	2.91E-65	11	1

R Console

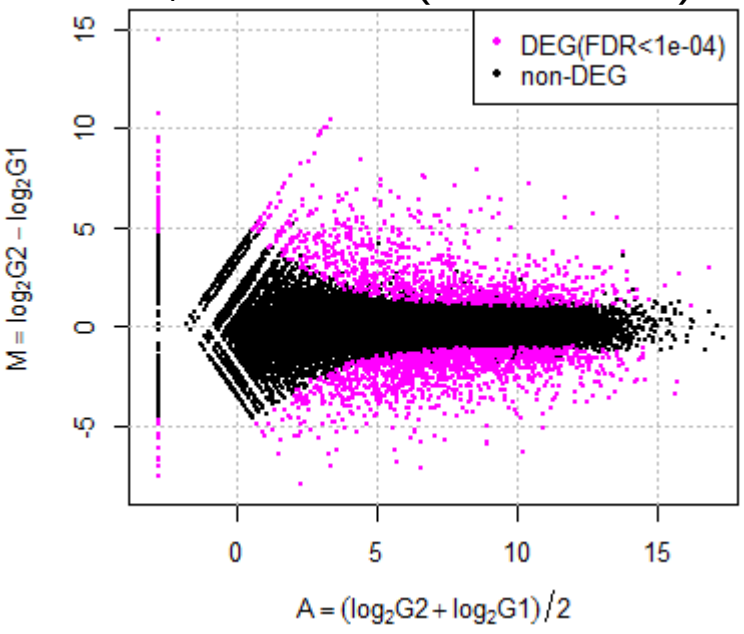
```
> (15.4 + 22.3 + 19.1)/3
[1] 18.93333
> (2856.6 + 2462.6 + 2555.3)/3
[1] 2624.833
> (log2(2624.833) + log2(18.93333))/2
[1] 7.800433
> log2(2624.833) - log2(18.93333)
[1] 7.115154
>
```

FDR閾値を緩めると得られるDEG数は増える傾向

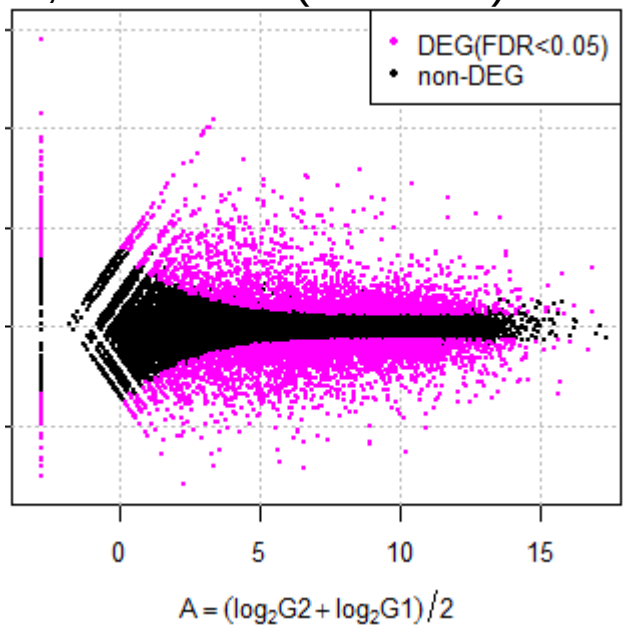
分布やモデルのイントロ

- TCCパッケージを用いたDEG同定

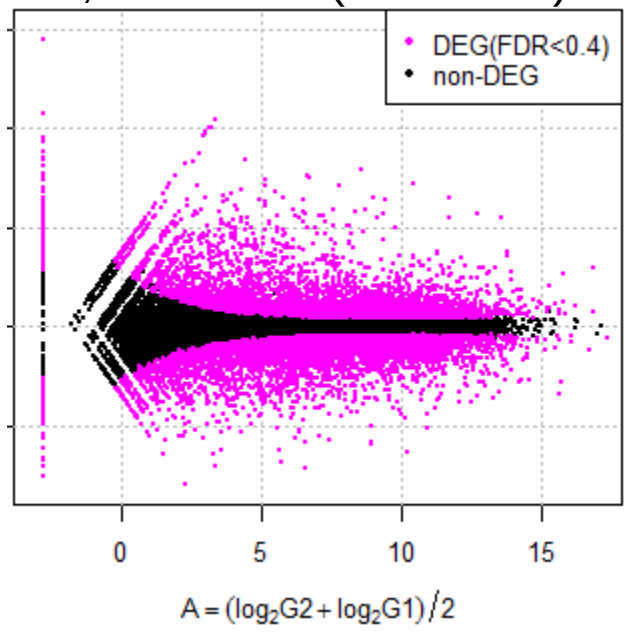
2,314 DEGs (FDR 0.01%)



5,669 DEGs (FDR 5%)



10,053 DEGs (FDR 40%)



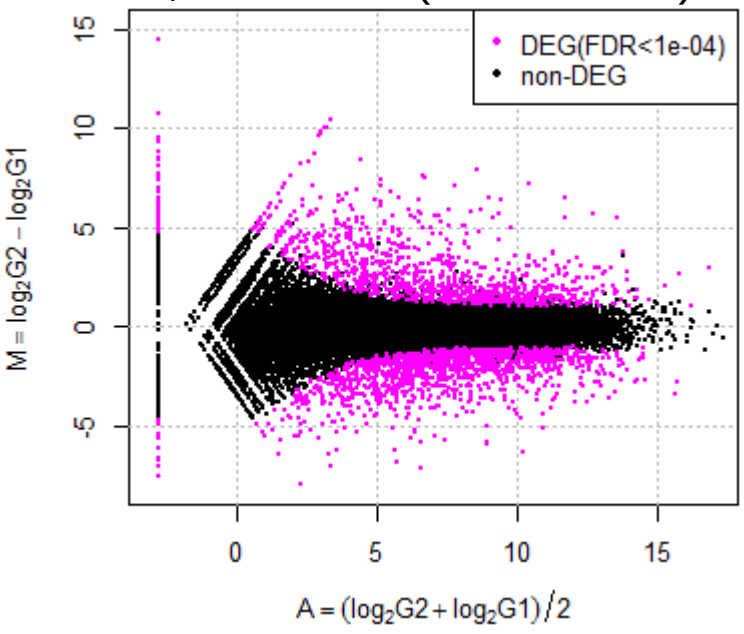
厳しめ ← FDR閾値 → 緩め

黒の分布はnon-DEGの分布に相当

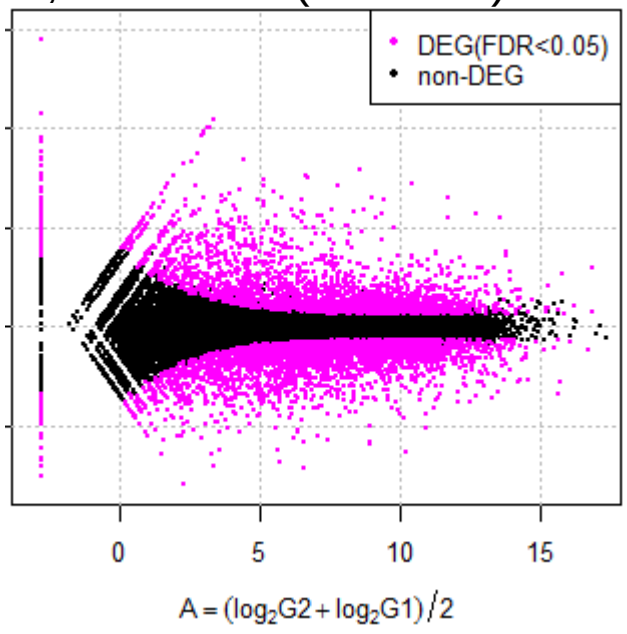
分布やモデル

- TCCパッケージを用いたDEG同定

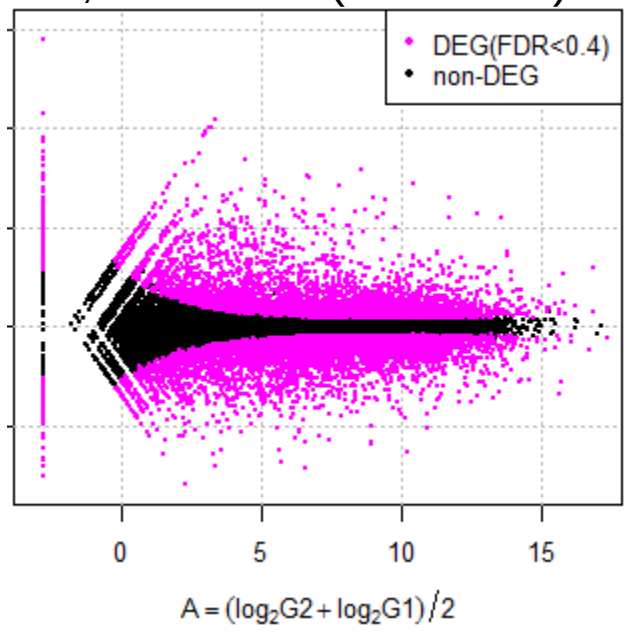
2,314 DEGs (FDR 0.01%)



5,669 DEGs (FDR 5%)



10,053 DEGs (FDR 40%)



厳しめ ← FDR閾値 → 緩め

分布やモデル

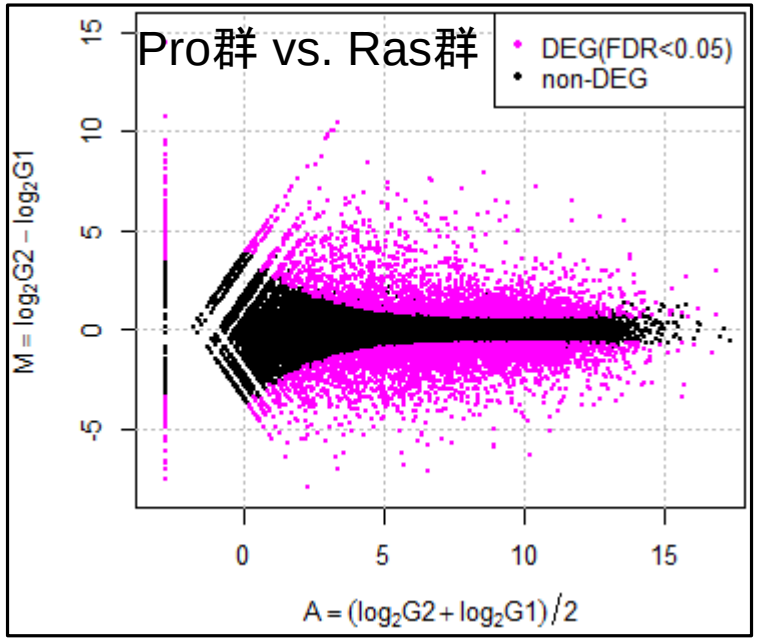
同一群（G1群）　同一群（G2群）

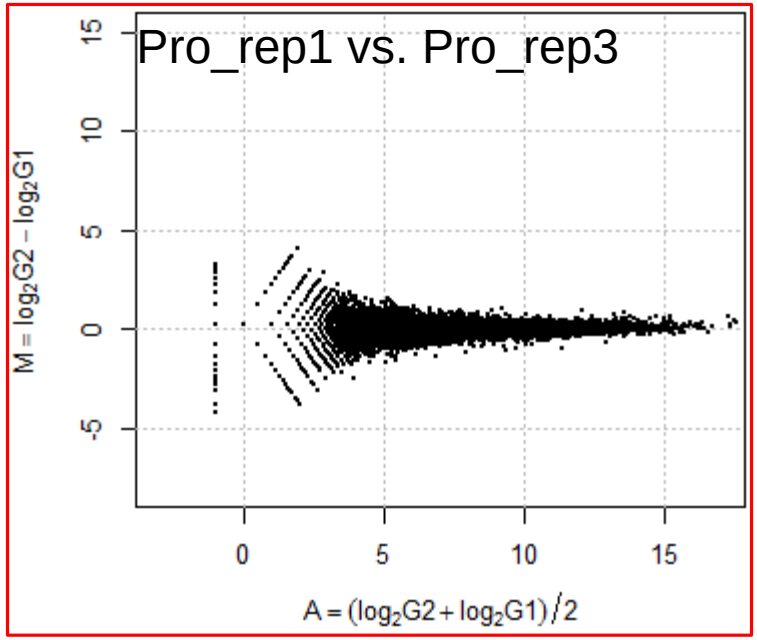
59,857 genes

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

G1群　G2群

黒の分布はnon-DEGの分布に相当





分布やモデル

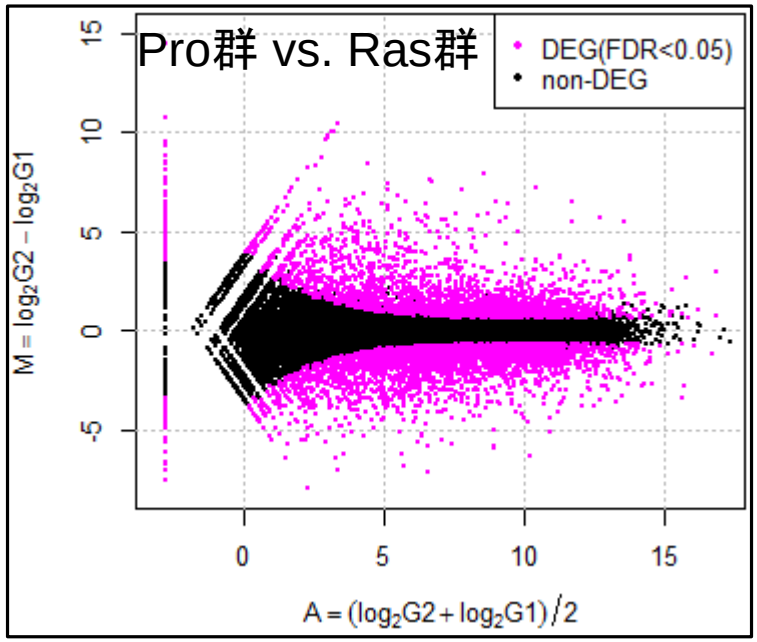
同一群（G1群）　同一群（G2群）

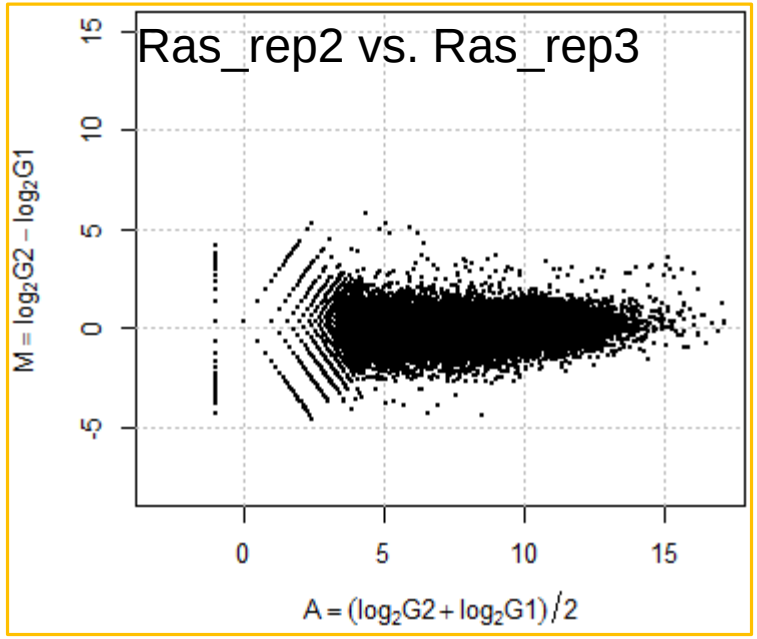
59,857 genes

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

G1群　G2群

黒の分布はnon-DEGの分布に相当





分布やモデル

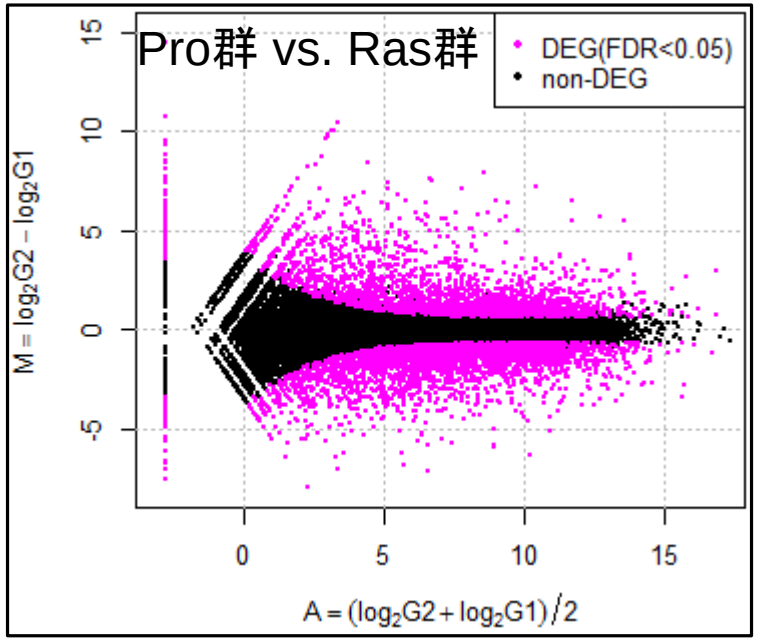
同一群（G1群）　同一群（G2群）

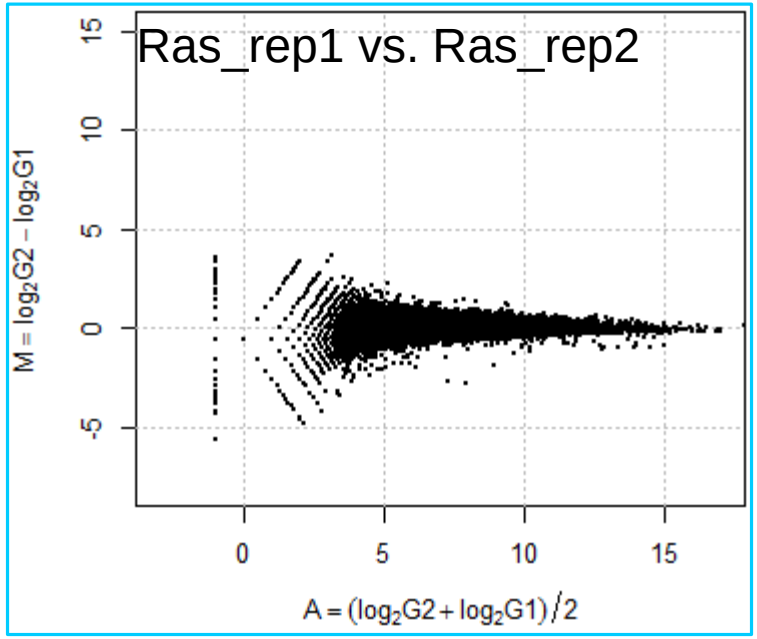
59,857 genes

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

G1群 G2群

黒の分布はnon-DEGの分布に相当





分布やモデル

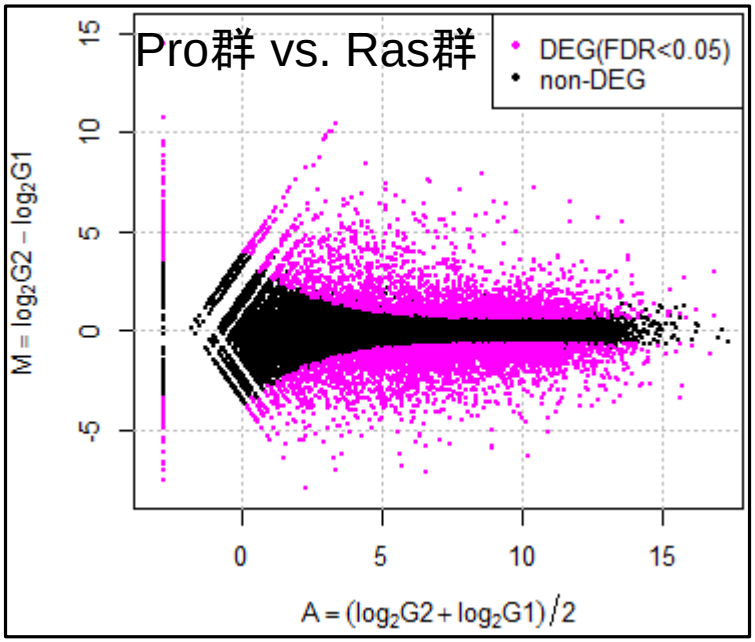
同一群（G1群）　同一群（G2群）

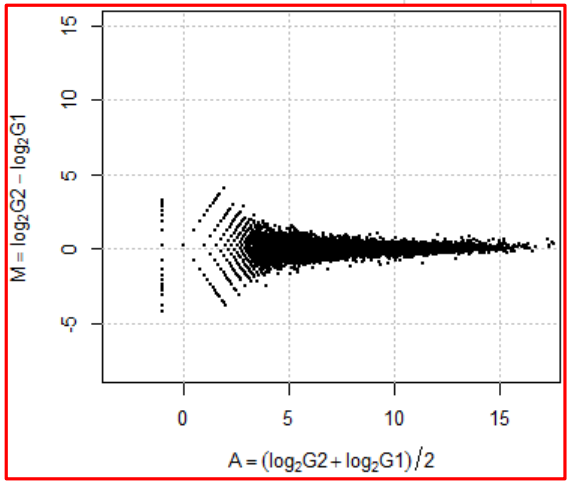
59,857 genes

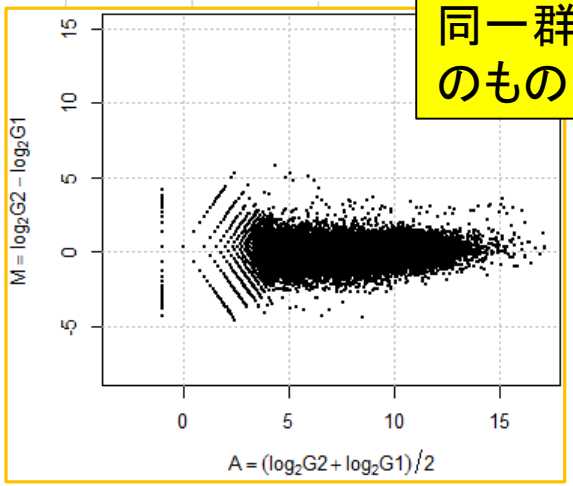
	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

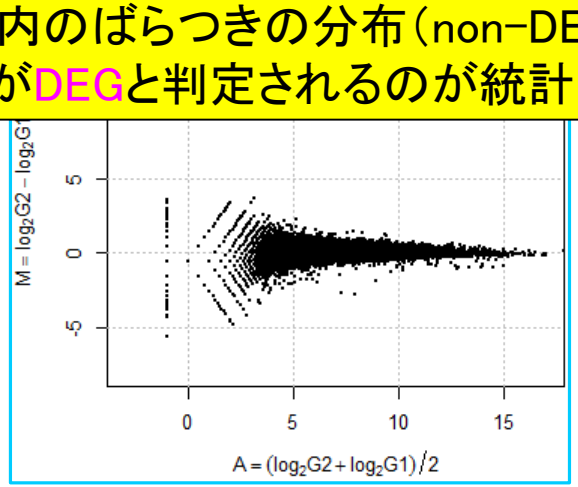
G1群　G2群　G1群　G2群

G1群　G2群









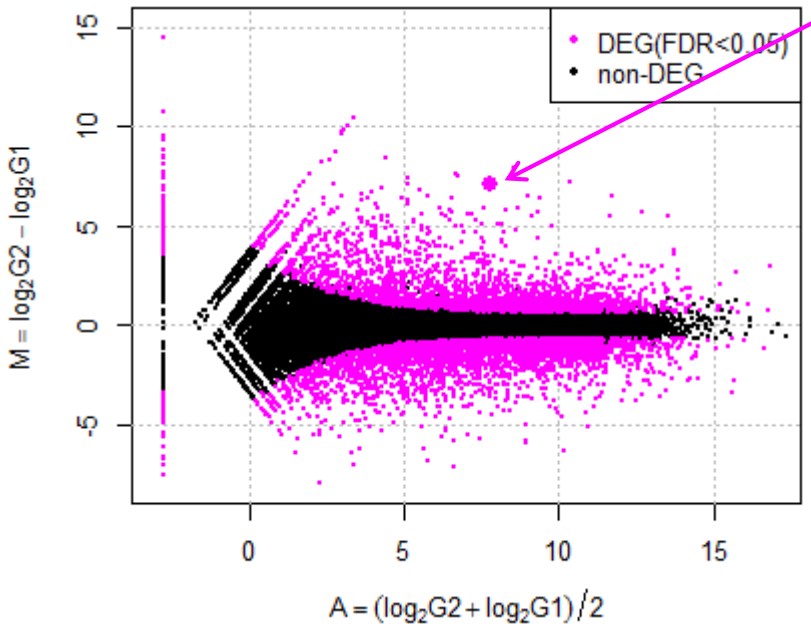
同一群内のばらつきの分布（non-DEG分布）以外のものがDEGと判定されるのが統計的手法の結果

統計的手法とは

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

- 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$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000240386	0.0	0.0	0.0	5608.1	5097.7	8188.0	ENSG00000240386	−2.78	14.48	1.75E−139	1.04E−134	1	1
ENSG00000128564	15.4	22.3	19.1	2856.6	2462.6	2555.3	ENSG00000128564	7.80	7.11	4.03E−107	1.21E−102	2	1
ENSG00000188064	7.7	5.8	10.1	1425.5	1254.0	1486.8	ENSG00000188064	6.71	7.47	2.67E−98	5.33E−94	3	1
ENSG00000101188	6.0	5.0	11.1	1477.4	1407.0	1254.9	ENSG00000101188	6.65	7.55	3.81E−97	5.70E−93	4	1

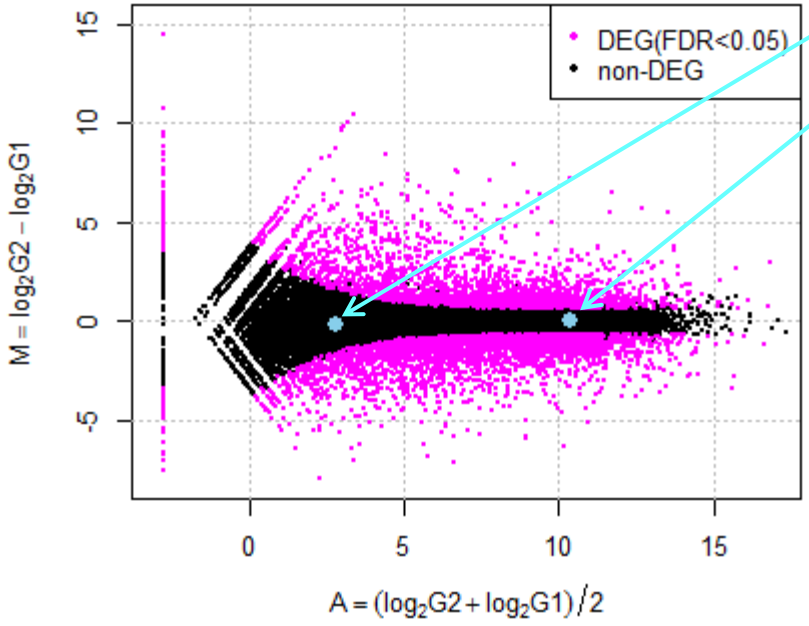


統計的手法とは

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

- 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$coun	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
ENSG00000165660	404.0	390.0	301.3	333.6	386.5	350.2	ENSG00000165660	8.50	−0.03	0.893466	1	24460	0
ENSG00000166359	4.3	7.4	10.1	9.8	3.7	6.0	ENSG00000166359	2.78	−0.16	0.893944	1	24461	0
ENSG00000146676	1141.9	1420.2	1272.8	1156.4	1558.0	1204.7	ENSG00000146676	10.34	0.03	0.89404	1	24462	0
ENSG00000229880	112.1	114.8	94.7	81.3	114.9	133.9	ENSG00000229880	6.76	0.04	0.894049	1	24463	0



倍率変化の結果

同一群内比較でも多数の偽陽性が検出されている

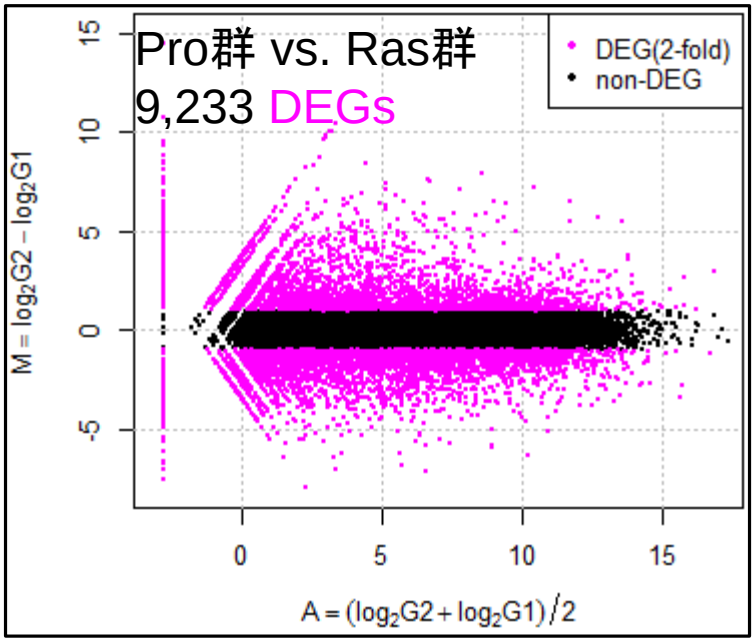
同一群（G1群）　同一群（G2群）

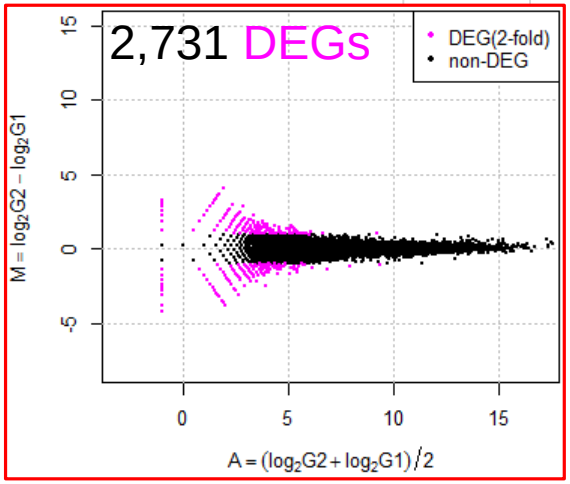
59,857 genes

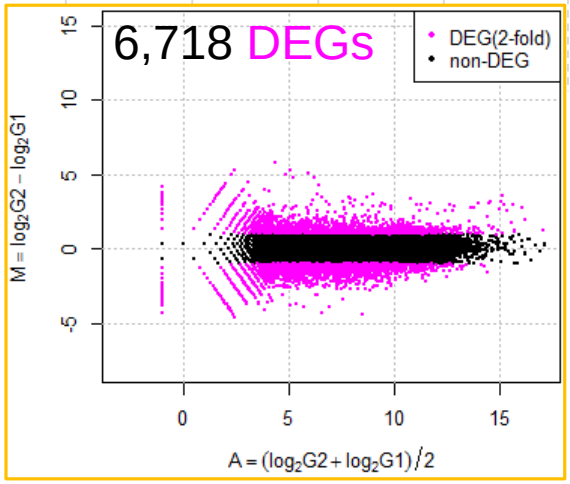
	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

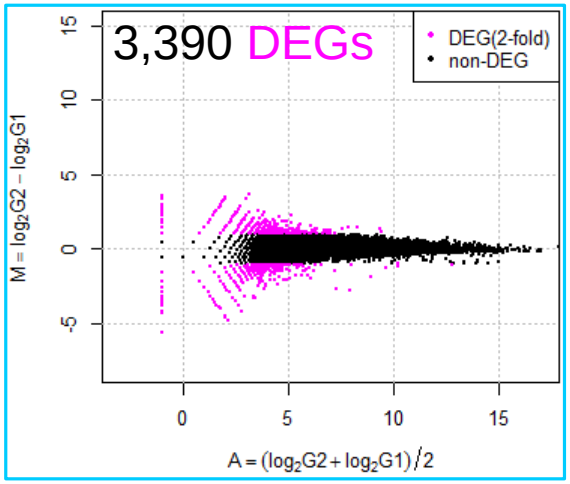
G1群　G2群　G1群　G2群

G1群　G2群









...

統計的手法TCCの結果

同一群内比較でも多少の偽陽性が検出されるが許容範囲

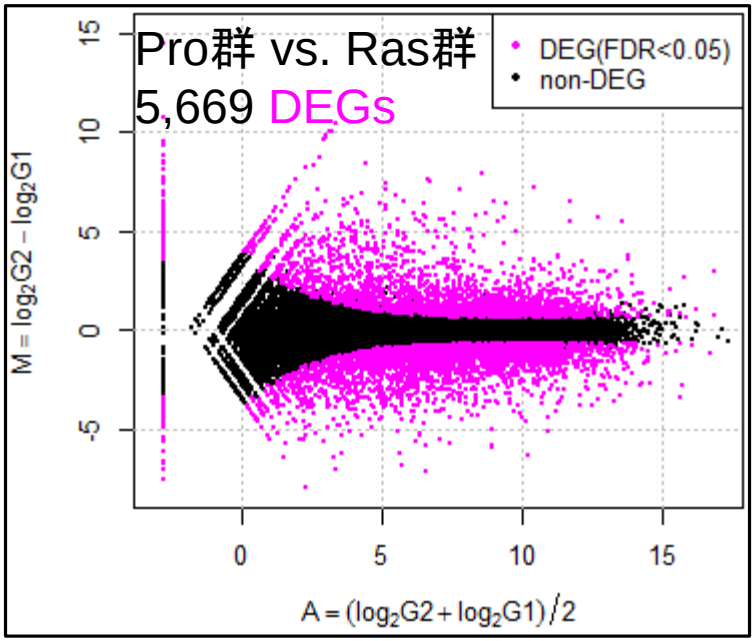
同一群（G1群）　同一群（G2群）

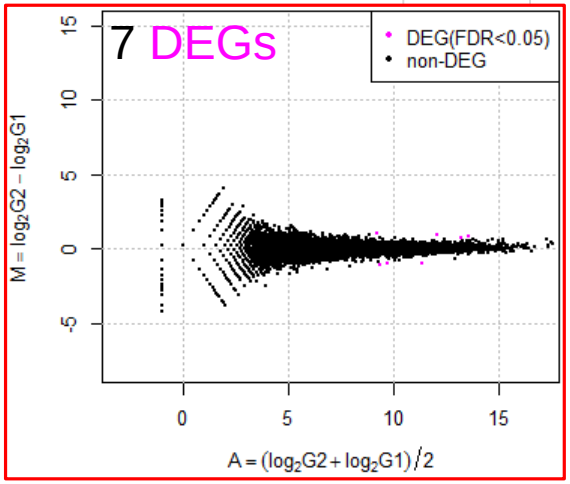
59,857 genes

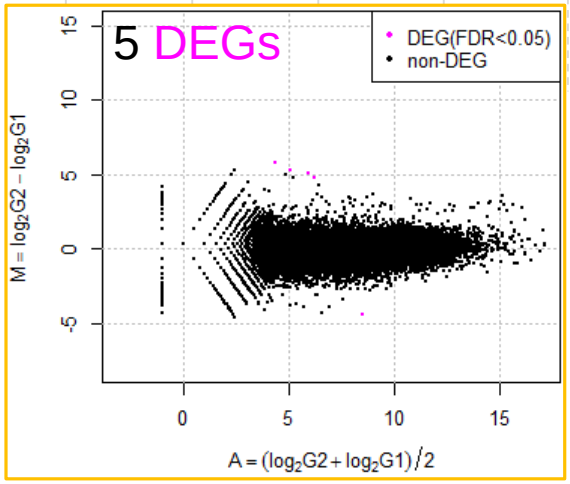
	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

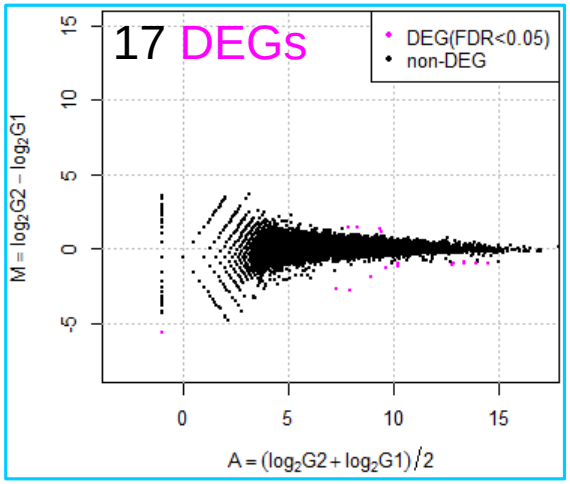
G1群　G2群　G1群　G2群

G1群　G2群







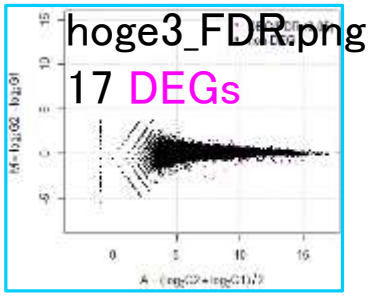


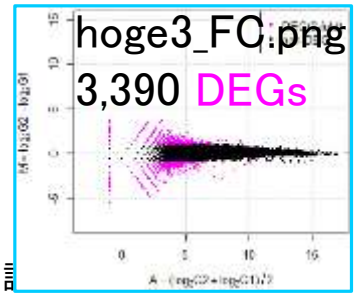
...

```
in_f <- "srp017142_count_bowtie.txt"    #入力ファイル名を指定してin_fに格納
out_f2 <- "hoge3_FDR.png"               #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge3_FC.png"                #出力ファイル名を指定してout_f3に格納
param_subset <- c(4, 5)                 #取り扱いたいサブセット情報を指定
param_G1 <- 1                           #G1群のサンプル数を指定
param_G2 <- 1                           #G2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_FC <- 2                           #fold-change閾値(param_FC倍)を指定
param_fig <- c(400, 390)                #MA-plot描画時の横幅と縦幅を指定(単位はピクセル)

library(TCC)                            #パッケージの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep
data <-data[,param_subset]              #param_subsetで
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を
tcc <- new("TCC", data, data.cl)        #TCCクラスオブ
tcc <- calcNormFactors(tcc, norm.method="deseq", test.
                       iteration=3, FDR=0.1, floorPDEG
tcc <- estimateDE(tcc, test.method="deseq", FDR=param_
result <- getResult(tcc, sort=FALSE)    #p値などの結果を
head(result, n=3)                       #確認してるだけ
sum(tcc$stat$q.value < param_FDR)       #FDR < param_FD
```

rcode_SRP017142_nonDEG.txt。解析したいサンプルの列番号とサンプル数を指定。パッケージのバージョン次第で結果が変わりうるのは確認済み。

```
###################################
### ファイルに保存(M-A plot; FDR)
###################################
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファ
plot(tcc, FDR=param_FDR, xlim=c(-3,17), ylim=c(-8, 15),#指定した閾値を満た
     normalize=T, median.lines=F)    #指定した閾値を満たすDEGをマゼンタ色
legend("topright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG"),
     col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                #おまじない
sum(tcc$stat$q.value < 0.05)             #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)             #FDR < 0.10を満たす遺伝子数を表示

###################################
### ファイルに保存(M-A plot; fold-change; FC)
###################################
M <- getResult(tcc)$m.value              #M-A plotのM値を抽出
hoge <- rep(1, length(M))                #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2        #条件を満たす位置に2を代入
cols <- c("black", "magenta")            #色情報を指定してcolsに格納
png(out_f3, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファ
plot(tcc, col=cols, col.tag=hoge, xlim=c(-3, 17), ylim=c(-8, 15),#M-A plot
     normalize=T, median.lines=F)    #M-A plotを描画
legend("topright", c(paste("DEG(", param_FC, "-fold)", sep=""), "non-DEG"),
     col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                #おまじない
sum(abs(M) > log2(4))                    #4倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(2))                    #2倍以上発現変動する遺伝子数を表示
```





Contents2

- トランスクリプトーム解析
  - イントロダクション
    - 簡単な原理、基本イメージ
  - NGSデータ取得(SRAdb)
    - 公共3大データベース(DDBJ SRA, EMBL-EBI ENA, NCBI SRA)、SRAdb
  - QC(Quality ControlまたはQuality Check)
  - マッピング、カウント情報取得(QuasR, Rbowtie)
  - 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
  - 機能解析、遺伝子セット解析(SeqGSEA)

機能解析

機能解析の実体は、GO解析やパスウェイ解析。2014年6月4日の大学院講義資料がイントロの理解として有用。

- 解析 | 発現変動 | 時系列 | について (last modified 2014/12/19)
- 解析 | 発現変動 | 時系列 | Bayesian model-based clustering(Nascimento 2012) (last modified 2012/09/10)
- 解析 | 発現変動 | 時系列 | maSigPro(Nueda 2014) (last modified 2014/07/18)
- 解析 | 発現変動 | エクソン | について (last modified 2015/01/29) NEW
- 解析 | 発現変動 | エクソン | DEXseq(Anders 2012) (last modified 2014/06/23)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について (last modified 2014/12/22)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA(Wang 2014)(last modified 2014/04/01)推奨
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GOseq(Young 2010) (last modified 2010/11/26)
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | について (last
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | SeqGSEA(W
- 解析 | 菌叢解析 | について (last modified 2014/12/19)
- 解析 | 菌叢解析 | phyloseq(McMurdie 2013) (last modified
- 解析 | エクソーム解析 | について (last modified 2014/07/06
- 解析 | ChIP-seq | について (last modified 2015/02/18) NEW

解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について

RNA-seqなどのタグカウントデータから遺伝子オントロジー(GO)解析を行うためのパッケージもいくつか出ています。遺伝子セット解析という枠組みではGO解析もパスウェイ解析も同じなので、そちらもチェックするといいかもしれません。
2014年12月に調査した結果をリストアップします。

R用:

- GAGE: Luo et al., BMC Bioinformatics, 2009
- goseq: Young et al., Genome Biol., 2010
- GOSemSim: Yu et al., Bioinformatics, 2010
- Rスクリプト: Gao et al., Bioinformatics, 2011
- RamiGO: Schröder et al., Bioinformatics, 2013
- GSVA: Hänzelmann et al., BMC Bioinformatics, 2013
- SeqGSEA: Wang et al., Bioinformatics, 2014

R以外:

-

Review、ガイドライン、パイプライン系:

- 手法比較: Rahmatallah et al., BMC Bioinformatics, 2014

機能解析

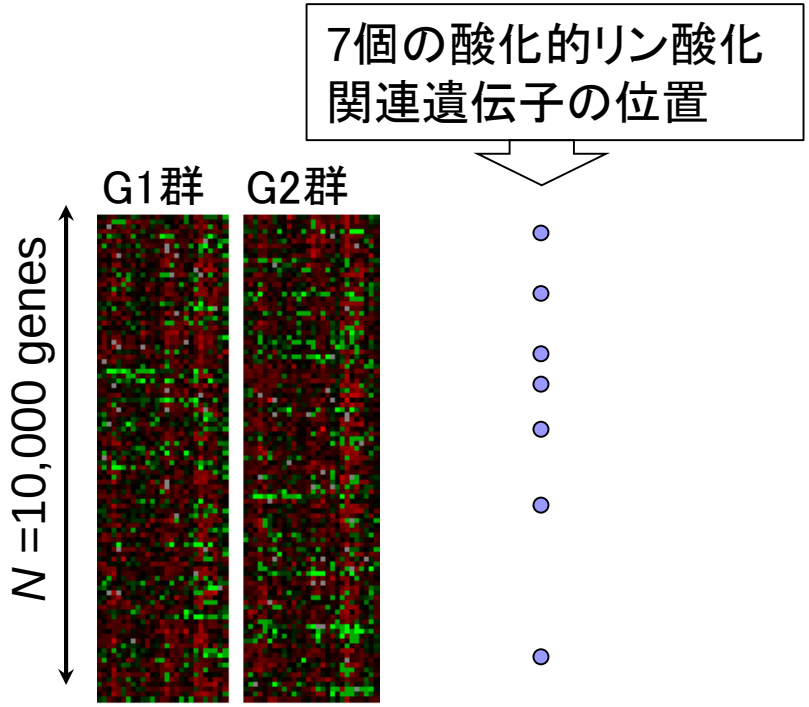
機能解析の実体は、遺伝子セットの発現変動解析。発現に差のある遺伝子セットを探したい、ということ

- Gene Ontology (GO)解析(発現に差のあるGO termを探索)
  - 基本3カテゴリ(Cellular Component (CC), Molecular Function (MF), Biological Process (BP))のどれでも可能
    - 例:肝臓の空腹状態 vs. 満腹状態のGO(BP)解析の結果、「脂肪酸β酸化」関連GO term (GO:0006635)が動いていることが分かった
- パスウェイ解析(発現に差のあるパスウェイを探索)
  - KEGG Pathway, BioCarta, Reactome pathway databaseのどれでも可能
    - 例:酸化的リン酸化パスウェイ関連遺伝子セットが糖尿病患者で動いていた
- モチーフ解析(発現に差のあるモチーフを探索)
  - 同じ3'-UTR microRNA結合モチーフをもつ遺伝子セット
  - 同じ転写因子結合領域(TATA-boxなど)をもつ遺伝子セット
    - 例:TATA-boxをもつ遺伝子セットがG1群 対 G2群比較で動いていた
- ...

機能解析（遺伝子セット解析）

酸化的リン酸化関連遺伝子セットが変動しているかどうかを調べたい、という問題を考える。

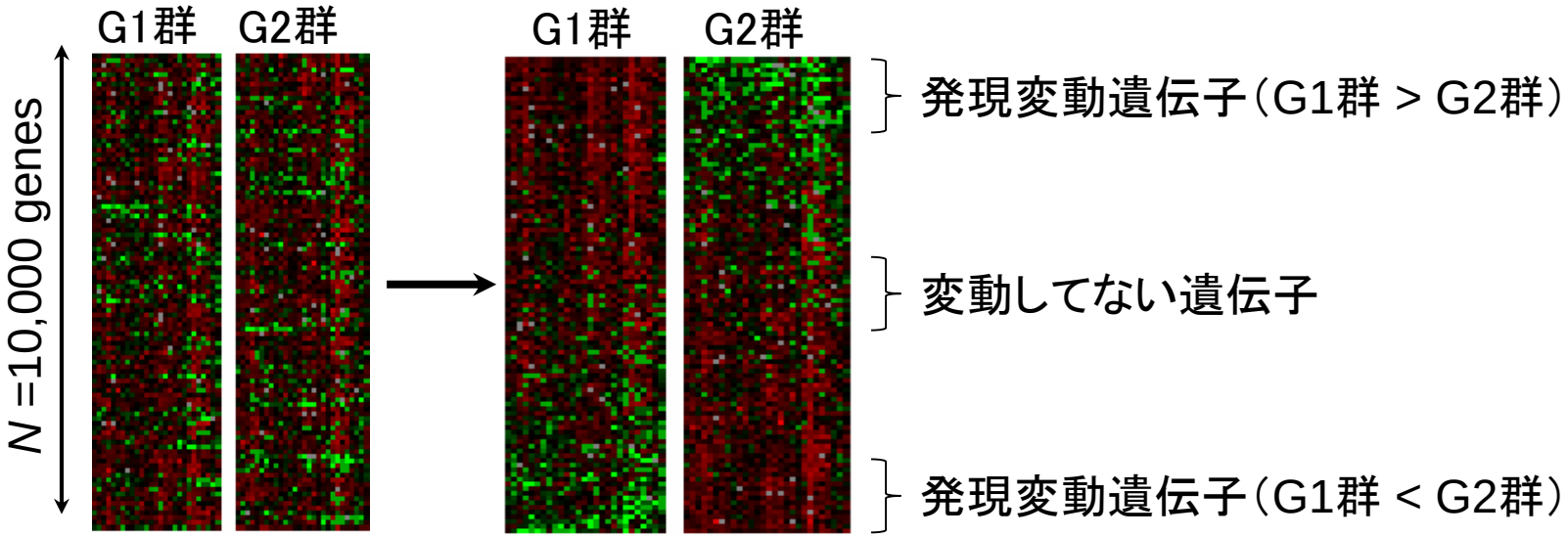
- 発現変動遺伝子セット解析手法（2群間比較用がほとんど）
  - N=10,000個の遺伝子からなる2群間比較用データ
  - この中に、XXX関連遺伝子がn個含まれている
    - 例：酸化的リン酸化（=XXX）関連遺伝子が7（=n）個含まれている



機能解析（遺伝子セット解析）

- 遺伝子ごとの発現変動の度合いを数値化
  - p-valueなどでもよい。
  - edgeR, DESeq, TCCなど一般的な発現変動解析用パッケージを原理的には利用可能
  - 機能解析用パッケージSeqGSEAは、内部的にDESeqを利用

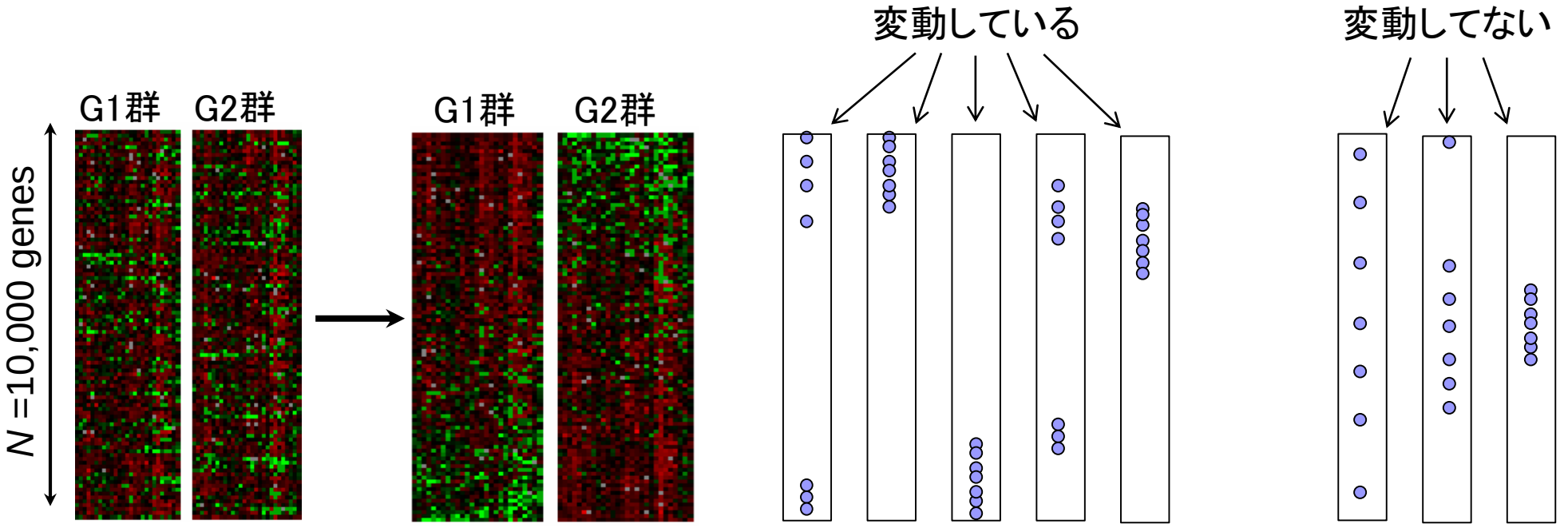
マイクロアレイデータへの適用時に議論されていたことと同様、RNA-seq用もDEG検出法、フィルタリング、関連のないデータやノイズの付与で検出力が上がるなど、まだよく分からないのが正直なところです。



どうやって偏りを評価するのか?

機能解析(遺伝子セット解析)

- 発現変動順にソート後の酸化的リン酸化関連遺伝子セットのステレオタイプな分布

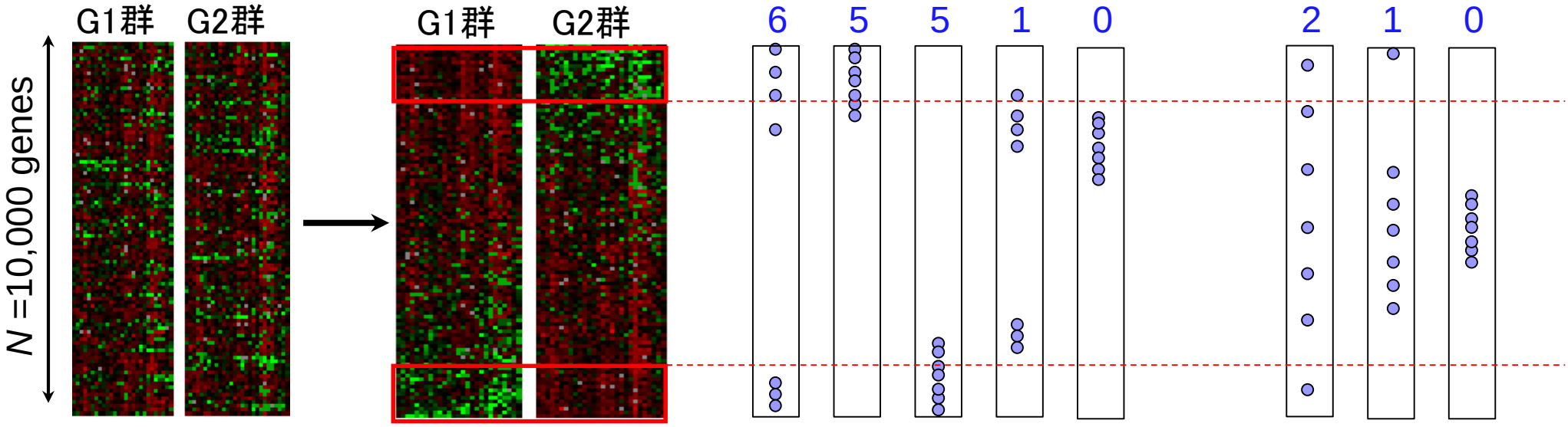


機能解析（遺伝子セット解析）

基本的な考え方は、「全遺伝子」と「上位のサブセット」のみで、調べたい遺伝子セットの割合が不変という帰無仮説のもとで検定

- Over-Representation Analysis (ORA)
  - 何らかの手段で決めた上位X(=1500)個のうち、x個が酸化的リン酸化関連遺伝子であった

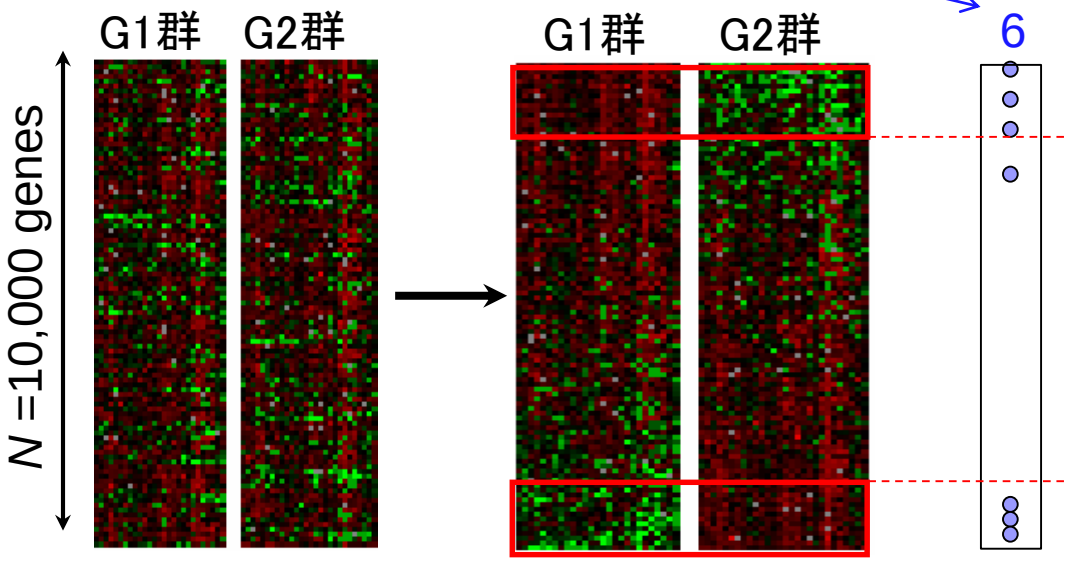
酸化的リン酸化関連遺伝子セット（n=7）が変動していない場合：$x/n \fallingdotseq X/N$ (= 1500/10000)
酸化的リン酸化関連遺伝子セット（n=7）が変動している場合：$x/n >> X/N$ (= 15%)



機能解析（遺伝子セット解析）

2×2分割表（contingency table）に基づく方法。超幾何検定やカイ二乗検定が利用されます。

- Over-Representation Analysis (ORA)
  - 何らかの手段で決めた上位*X*(=1500)個のうち、*x*個が酸化的リン酸化関連遺伝子であった



XXX=酸化的リン酸化関連遺伝子セット

	XXX	XXX以外	計
non-DEG数	1	8500-1	N-X
DEG数	6	1500-6	X
計	n	N-n	

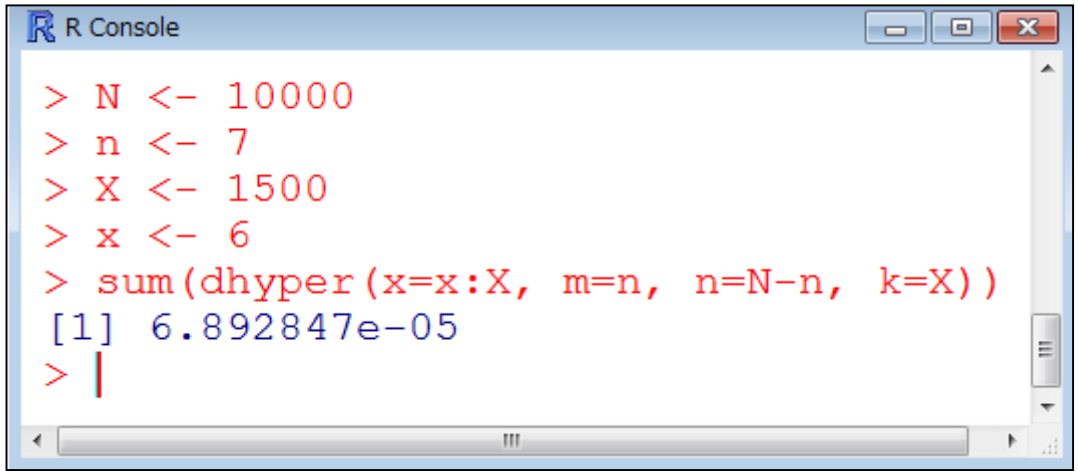
機能解析(超幾何検定)

DEGとして1500個抽出したとき、酸化的リン酸化関連遺伝子が6個以上含まれる確率として算出

- *N*=10000個の遺伝子発現データ中にXXX=酸化的リン酸化関連遺伝子は*n*=7個含まれていた。上位*X*=1500個の発現変動遺伝子(DEG)の中に*x*=6個の酸化的リン酸化関連遺伝子が含まれていた
  - 帰無仮説:酸化的リン酸化関連遺伝子の割合はDEGとnon-DEG間で差がない

	XXX	XXX以外	計
non-DEG数	1	8500-1	8500
DEG数	6	1500-6	1500
計	7	9993	10000

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N



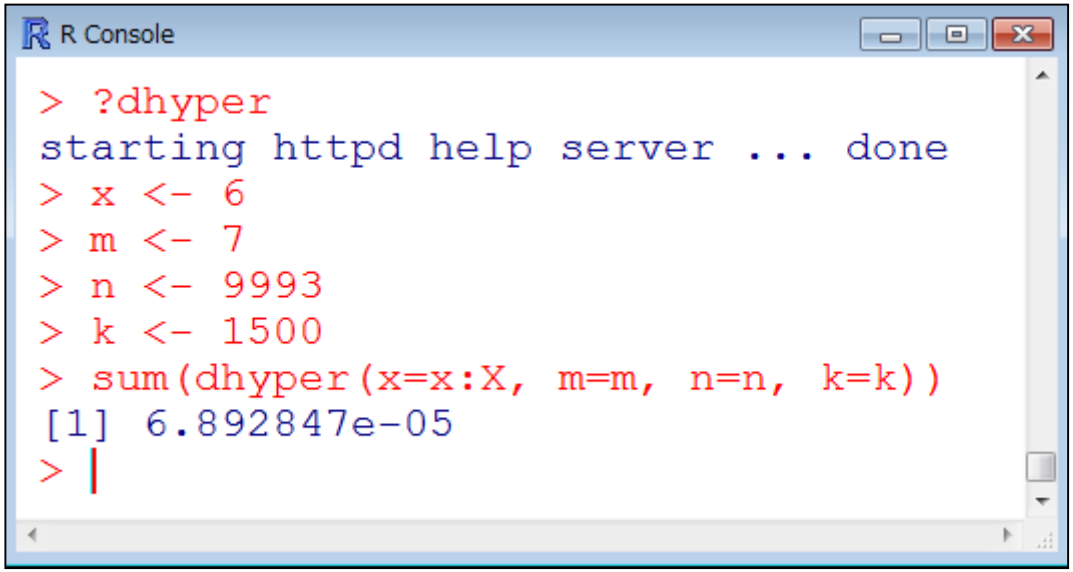
機能解析（超幾何検定）

?dhyperマニュアル中の一般的な説明に置き換えるとこんな感じです

- *m*=7個の白いボールと*n*=9993個の黒いボールが入った箱があります（トータルで*N*=*m*+*n*=10,000個）。この中から*k*=1500個ランダムに取り出したときに*x*=6個以上白いボールが含まれる確率を計算しなさい。

	白	黒	計
箱の中	1	9993−(1500−6)	8500
箱の外	6	1500−6	1500
計	7	9993	10000

	白	黒	計
箱の中	m-x	n-(k-x)	m+n-k
箱の外	x	k-x	k
計	m	n	N



機能解析(カイ二乗検定)

DEGとして1500個抽出したとき、酸化的リン酸化関連遺伝子が6個以上含まれる確率として算出

	XXX	XXX以外	計
non-DEG数	1	8500-1	8500
DEG数	6	1500-6	1500
計	7	9993	10000

	XXX	XXX以外	計
non-DEG数	n-x	$(N-n)$-$(X-x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N

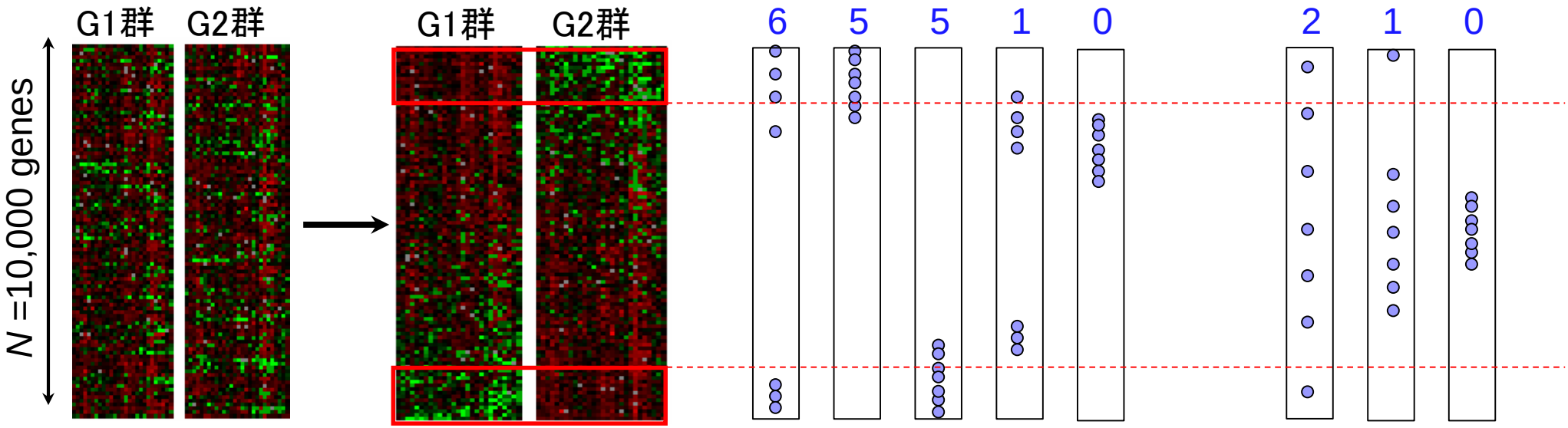
R Console

```
> N <- 10000
> n <- 7
> X <- 1500
> x <- 6
> data <- matrix(c((n-x), (N-n)-(X-x), x, (X-x)), ncol=2, byrow=T)
> data
     [,1] [,2]
[1,]    1 8499
[2,]    6 1494
> chisq.test(data)

        Pearson's Chi-squared test with Yates' continuity
        correction

data:  data
X-squared = 22.2032, df = 1, p-value = 2.453e-06

 警告メッセージ:
In chisq.test(data) :  カイ自乗近似は不正確かもしれません
>
```

機能解析（遺伝子セット解析）

上位1500個のうち、酸化的リン酸化関連遺伝子が7個中4つ以上含まれていればp < 0.05で検出可能ということを意味する。

- Over-Representation Analysis (ORA)



p-value	*x*=6	*x*=5	*x*=4	*x*=3	*x*=2	*x*=1	*x*=0
超幾何検定	6.89E-05	0.0012	0.0121	0.0737	0.2834	0.6795	1.0000
カイ二乗検定	2.45E-06	0.0003	0.0095	0.1247	0.6337	0.6337	0.5603
Fisher test	6.89E-05	0.0012	0.0121	0.0737	0.2834	1.0000	0.6039

$p < 0.05$を灰色で示した

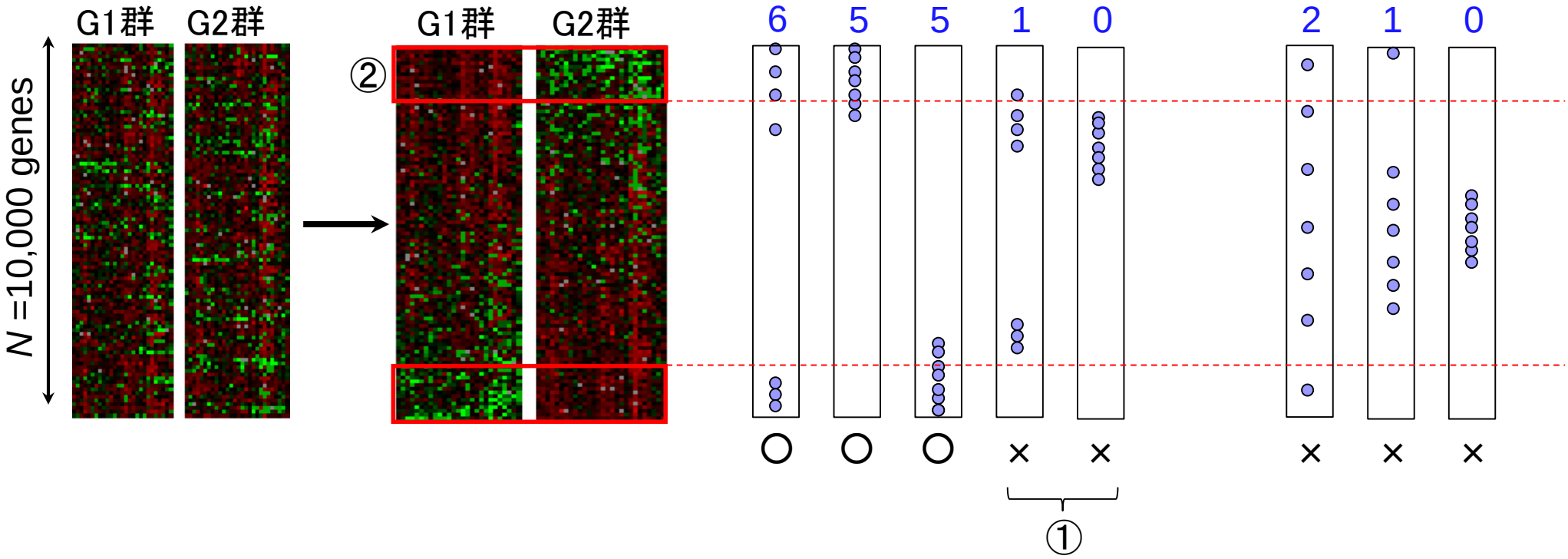
第一世代(ORA)の短所

① 全体的には動いているものの、個々の発現変動の度合いが弱い場合に検出困難
② 上位X個のX次第で結果が変わる
③ 情報量低下(発現変動の度合い → カウント情報)

もちろん分割表ベースの方法(ORA)ではない第二世代以降の方法があります。代表例はGene Set Enrichment Analysis (GSEA)。

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N

③



遺伝子セット解析の課題

RNA-seqでより高い解像度(exon-level)のデータが得られたとしても、ボトルネックは知識の解像度(gene-level)。

- (知識ベースの解析法なので)解析対象がアノテーションの情報の豊富な生物種に限定
  - それ以外の生物種は、まずは地道にアノテーション情報を増やしていくことが先決(ではないだろうか)
  - アノテーションの解像度を上げる努力も大事
- アノテーション情報の信頼度が高いとはいえない
  - なんらかのGO termがついていたとしても、その大部分のevidence codeが自動でつけられたもの(IEA, inferrred from electronic annotations)である…
- 遺伝子セット間の独立性の問題
  - 「数百個程度の遺伝子セットの中から、比較するサンプル間で動いている遺伝子セットはどれか？」という解析を遺伝子セット間の独立性を仮定して調べるが、そもそも独立ではない(GO term間の親子関係などから明らか)

    → いくつくらいの遺伝子セットが動いているのか？という問いに答えるすべがない
- 評価に用いられる「よく研究されているデータセット」は答えが完全に分かっているものではない(the actual biology is never fully known!)
  - “感度が高い”と謳っているだけの方法は…(全部の遺伝子セットが動いている →感度100%)

遺伝子セット解析おさらい

どの遺伝子セットにどの遺伝子が所属しているかというgmt形式ファイルの取得が第一歩

- Gene Ontology (GO)解析(発現に差のあるGO termを探索)
  - 基本3カテゴリ(Cellular component (CC), Molecular Function (MF), Biological Process (BP))のどれでも可能
    - 例:肝臓の空腹状態 vs. 満腹状態のGO(BP)解析の結果、「脂肪酸β酸化」関連GO term (GO:0006635)が動いていることが分かった
- パスウェイ解析(発現に差のあるパスウェイを探索)
  - KEGG, BioCarta, Reactome pathway databaseのどれでも可能
    - 例:酸化的リン酸化パスウェイ関連遺伝子セットが糖尿病患者で動いていた
- モチーフ解析(発現に差のあるモチーフを探索)
  - 同じ3'-UTR microRNA結合モチーフをもつ遺伝子セット
  - 同じ転写因子結合領域(TATA-boxなど)をもつ遺伝子セット
    - 例:TATA-boxをもつ遺伝子セットがG1群 対 G2群比較で動いていた
- …

MSigDB ver. 4.0

MSigDBは、Molecular Signature Databaseの略。様々な遺伝子セット解析を行うためのgmt形式ファイルをダウンロード可能です。

- c1: positional gene sets (326 gene sets)
  - ヒト染色体の位置ごとの遺伝子セットリストファイル (326 gene sets)
- c2: curated gene sets (4,722 gene sets)
  - **CGP**: chemical and genetic perturbations (3,402 gene sets)
  - **CP**: canonical pathways (1,320 gene sets)
  - **CP:BIOCARTA**: BioCarta gene sets (217 gene sets)
  - **CP:KEGG**: KEGG gene sets (186 gene sets)
  - **CP:REACTOME**: Reactome gene sets (674 gene sets)
- c3: motif gene sets (836 gene sets)
  - **MIR**: microRNA targets (221 gene sets)
  - **TFT**: transcription factor targets (615 gene sets)
- c4: computational gene sets (858 gene sets)
  - **CGM**: cancer gene neighborhoods (427 gene sets)
  - **CM**: cancer modules (431 gene sets)
- c5: gene ontology (GO) gene sets (1,454 gene sets)
  - **BP**: biological process (825 gene sets)
  - **CC**: cellular component (233 gene sets)
  - **MF**: molecular function (396 gene sets)
- c6: oncogenic signatures gene sets (189 gene sets)
- c7: immunologic signatures gene sets (1,910 gene sets)

発現変動と関連するKEGGパスウェイを調べたいとき

発現変動と関連するBP中のGO termsを調べたいとき

機能解析

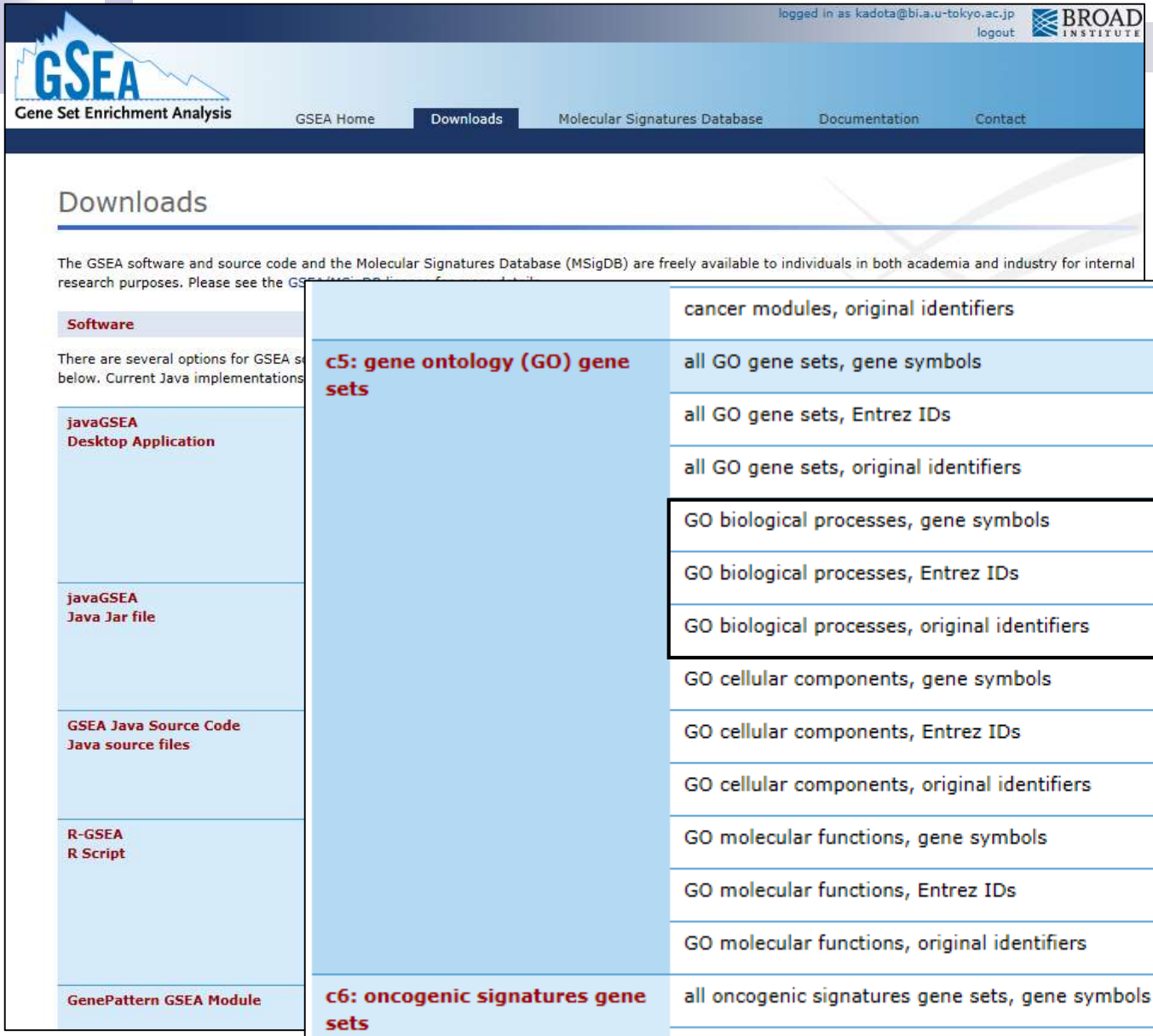
遺伝子セット解析（GO解析）を行うためのgmt形式ファイルのダウンロード方法はこちら

- 解析 | 発現変動 | エクソン | DEXseq(Anders_2012) (last modified 2014/06/23)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について (last modified 2015/02/27) NEW
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA(Wang_2014) (last modified 2015/02/27)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GOseq(Young_2010) (last modified 2010/11/26)
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | について (last modified 2015/02/27) NEW
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | SeqGSEA(Wang_2014) (last modified 2015/02/27)推奨

解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA (Wang_2014) NEW

SeqGSEAを用いてGO解析を行うやり方を示します。このパッケージは、exonレベルのカウントデータを入力として、発現変動遺伝子セットに相当する有意に発現変動したGO termsを出力するのが基本ですが、geneレベルのカウントデータを入力として解析することも可能です。統計的有意性の評価にサンプルラベル情報の並べ替え(permutation)戦略を採用しているため、各グループあたりの反復数が5以上のデータを想定しているようです(Wang et al., 2014)。また、計算時間が半端なくかかります。例えば、並べ替え回数がたったの20回でも2時間ちょっとかかります(Panasonic Let's note CF-SX3本郷モデルの場合)のでご注意ください。推奨は1000回以上と書いてますが、10日ほどかかることになるので個人的にはアリエナイですね...。SeqGSEAでの機能解析の基本は、exonレベルとgeneレベルの発現変動解析結果を組み合わせてGSEAを行うというものです(Wang and Cairns, 2013)。SeqGSEA著者たちは、exonレベルの発現変動解析のことをDifferential splicing (DS) analysisと呼んでいて、おそらくDSGseq (Wang et al., 2013)はSeqGSEA中に組み込まれています。そしてgeneレベルの発現変動解析をDifferential expression (DE) analysisとして、SeqGSEA中ではDESeqを利用しています。GSEAに代表される発現変動遺伝子セット解析は、基本的にGSEAの開発者らが作成した様々な遺伝子セット情報を収めたMolecular Signatures Database (MSigDB)からダウンロードした.gmt形式ファイルを読み込んで解析を行います。*gmt形式ファイルのダウンロード方法は、基本的に以下の通りです：

a. Molecular Signatures Database (MSigDB)の
「register」のページで登録し、遺伝子セットをダウンロード可能な状態にする。
b. Molecular Signatures Database (MSigDB)の
「Download gene sets」の"Download"のところをクリックし、Loginページで登録したe-mail addressを入力。
c. これでMSigDBのダウンロードページに行けるので、
「c5: gene ontology gene sets」の「bp: biological process」を解析したい場合はc5.bp.v4.0.symbols.gmtファイルをダウンロードしておく。
「c5: gene ontology gene sets」の「cc: cellular components」を解析したい場合はc5.cc.v4.0.symbols.gmtファイルをダウンロードしておく。
「c5: gene ontology gene sets」の「mf: molecular functions」を解析したい場合はc5.mf.v4.0.symbols.gmtファイルをダウンロードしておく。



発現変動と関連するbiological processes (BP)中のGO termsを調べたいときは、この中のいずれかのgmtファイルを利用する。

	cancer modules, original identifiers	c4.cm.v4.0.orig.gmt
c5: gene ontology (GO) gene sets	all GO gene sets, gene symbols	c5.all.v4.0.symbols.gmt
	all GO gene sets, Entrez IDs	c5.all.v4.0.entrez.gmt
	all GO gene sets, original identifiers	c5.all.v4.0.orig.gmt
	GO biological processes, gene symbols	c5.bp.v4.0.symbols.gmt
	GO biological processes, Entrez IDs	c5.bp.v4.0.entrez.gmt
	GO biological processes, original identifiers	c5.bp.v4.0.orig.gmt
	GO cellular components, gene symbols	c5.cc.v4.0.symbols.gmt
	GO cellular components, Entrez IDs	c5.cc.v4.0.entrez.gmt
	GO cellular components, original identifiers	c5.cc.v4.0.orig.gmt
	GO molecular functions, gene symbols	c5.mf.v4.0.symbols.gmt
	GO molecular functions, Entrez IDs	c5.mf.v4.0.entrez.gmt
	GO molecular functions, original identifiers	c5.mf.v4.0.orig.gmt
c6: oncogenic signatures gene sets	all oncogenic signatures gene sets, gene symbols	c6.all.v4.0.symbols.gmt
	all oncogenic signatures gene sets, Entrez IDs	c6.all.v4.0.entrez.gmt
	all oncogenic signatures gene sets, original identifiers	c6.all.v4.0.orig.gmt
c7: immunologic signatures	all immunologic signatures gene sets, gene symbols	c7.all.v4.0.symbols.gmt

gmtファイル

基本的にどれを使っても自由だが、利用するRパッケージがどの入力形式を受け付けるかにも依存する。経験上gene symbolsを使っておけば間違いないので、門田は*.symbols.gmt いつも利用しています。

	A	B	C	D	E	F	G
1	TRNA_PROCESSING	http://www	ADAT1	TRNT1	FARS2	METTL1	SARS
2	REGULATION_OF_BIOLOGICAL_QUALI	http://www	DLC1	ALS2	SLC9A7	PTGS2	PTGS1
3	DNA_METABOLIC_PROCESS	http://www	XRCC5	XRCC4	RAD51C	XRCC3	XRCC2
4	AMINO_SUGAR_METABOLIC_PROCESS	http://www	UAP1	CHIA	GNPDA1	GNE	CSGALNAC
5	BIOPOLYMER_CATABOLIC_PROCESS	http://www	BTRC	HNRNPD	USE1	RNASEH1	RNF217

	A	B	C	D	E	F	G
1	TRNA_PROCESSING	http://www	23536	51095	10667	4234	6301
2	REGULATION_OF_BIOLOGICAL_QUALI	http://www	10395	57679	84679	5743	5742
3	DNA_METABOLIC_PROCESS	http://www	7520	7518	5889	7517	7516
4	AMINO_SUGAR_METABOLIC_PROCESS	http://www	6675	27159	10007	10020	55790
5	BIOPOLYMER_CATABOLIC_PROCESS	http://www	8945	3184	55850	246243	154214

c5.bp.v4.0.symbols.gmt
c5.bp.v4.0.entrez.gmt
c5.bp.v4.0.orig.gmt

1列目：遺伝子セット名
2列目：URL
3列目以降：gene ID or symbol

機能解析

MSigDB ver. 4.0で提供されているGO biological processes (BP)のgmtファイル(c5.bp.v4.0.symbols.gmt)中には、全部で825個の遺伝子セットが存在する。手元の2群間比較用カウントデータファイル(srp017142_count_bowtie.txt)を入力として機能解析(この場合GO解析)で行うことは、どの遺伝子セットが比較する2群間で発現変動しているかを調べることに相当する。

srp017142_count_bowtie.txt

59,857 genes

	Pro_rep1	Pro_rep2	Pro_rep3	Ras_rep1	Ras_rep2	Ras_rep3
ENSG00000000003	480	513	366	124	271	366
ENSG00000000005	0	0	0	1	0	0
...						
ENSG00000240386	0	0	0	4001	5500	6851
...						
ENSG00000128564	18	27	19	2038	2657	2138
...						

c5.bp.v4.0.symbols.gmt

	A	B	C	D	E	F	G	
1	TRNA_PROCESSING	http://www	ADAT1	TRNT1	FARS2	METTL1	SARS	AARS
2	REGULATION_OF_BIOLOGICAL_QUALI	http://www	DLC1	ALS2	SLC9A7	PTGS2	PTGS1	MPV
3	DNA_METABOLIC_PROCESS	http://www	XRCC5	XRCC4	RAD51C	XRCC3	XRCC2	XRC
4	AMINO_SUGAR_METABOLIC_PROCESS	http://www	UAP1	CHIA	GNPDA1	GNE	CSGALNAC	CHS
5	BIOPOLYMER_CATABOLIC_PROCESS	http://www	BTRC	HNRNPD	USE1	RNASEH1	RNF217	ISG2
6	DNA_METABOLIC_PROCESS	http://www	HNRNPF	HNRNPD	SYNCRIP	MED24	POPR	MED

機能解析

機能解析の出力は、825個の遺伝子セットごとに計算された統計量(p値やFDR値)である。これらをもとに5% FDRなど任意の閾値を満たす遺伝子セットを探すことになるが、赤枠のような遺伝子セットを構成するメンバー数(これは10 gene symbols)があまりに少なかったり、多すぎるものはデフォルトで解析対象から除外されることもある。

c5.bp.v4.0.symbols.gmt

	A	B	C	D	E	F	G							
797	DETECTION_OF_EXTERNAL_STIMULUS	http://www	RP1	UNC119	TRPC3	PGLYRP1	NR2E3	OP						
798	CELLULAR_RESPONSE_TO_NUTRIENT_	http://www	SREBF1	CHMP1A	CDKN2B	NUAK2	ALB	FAI						
799	NUCLEOCYTOPLASMIC_TRANSPORT	http://www	NCBP2	HTATIP2	XPO6	PDIA3	EIF5A	TNI						
800	M_PHASE_OF_MITOTIC_CELL_CYCLE	http://www	KIF22	KIF25	NBN	KNTC1	TTK	PKI						
801	POSITIVE_REGULATION_OF_CYTOKINE	http://www	TRAF2	ATP6AP2	MALT1	AFAP1L2	SOD1	CAI						
802	SPERMATID_DIFFERENTIATION	http://www	CSNK2A2	NME5	H1FNT	MAST2	SPAG6	CEI						
803	RESPONSE_TO_ORGANIC_SUBSTANCE	http://www	ME1	AQP9	DUOX2	DUOX1	GLRX2	GOT2	APOA2	NDUFS4	DGKD	BCL2	GYS2	CHRNA4
804	RHO_PROTEIN_SIGNAL_TRANSDUCTIC	http://www	FGD2	ALS2	ARHGDIG	FGD1	RTKN	ARF6	ABCA1	ARHGAP4	ARHGAP5	APOA1	ROPN1B	APOE
805	CHROMOSOME_CONDENSATION	http://www	PAM	CHMP1A	NCAPH	AIFM2	DFFB	NUSAP1	ACIN1	TOP2A	CDCA5	SMC4		
806	CELL_FATE_COMMITMENT	http://www	POU6F2	JAG2	MYEF2	SMAD2	TRIM15	GLI2	SHH	WNT1	NOTCH2	EYA2	NOTCH4	IFRD1
807	RESPONSE_TO_HYPOXIA	http://www	CLDN3	PDIA2	ARNT2	PML	EGLN2	BNIP3	EGLN1	TGFB2	ALAS2	PLOD1	PLOD2	CXCR4
808	MICROTUBULE_POLYMERIZATION_OR	http://www	ARHGEF2	MAPT	TPPP	KATNB1	CLASP1	STMN1	MAPRE1	CLASP2	MID1IP1	CENPJ	APC	
809	REPRODUCTION	http://www	CGB	VAPB	CRHBP	LHCGR	DNAJB13	PTTG1	SHH	WWP2	ADAM2	WWP1	OR7C1	GDF9
810	CAMP_MEDIATED_SIGNALING	http://www	MCHR1	ADORA3	ADCY7	WASF2	GABBR1	GABBR2	GHRHR	EDNRA	EDNRB	NDUFS4	GALR1	GALR3
811	REGULATION_OF_CELL_PROLIFERATIC	http://www	RARRES3	RARRES1	AIF1	PGF	BTC	EDN1	MXI1	SSR1	MAGED1	CUL3	CD47	CUL2
812	NITROGEN_COMPOUND_CATABOLIC_F	http://www	BCKDK	GLUD1	AMT	ASL	IDO1	GLDC	FAH	GOT2	MCCC2	ARG1	ASPA	GAD2
813	REGULATION_OF_I_KAPPAB_KINASE_NI	http://www	RPL17	SLC44A2	VAPA	SLC20A1	TBK1	TSPAN6	GJA1	FASLG	MUL1	TLR6	SHISA5	CANT1
814	RESPONSE_TO_DRUG	http://www	NAT8	SLC22A18	CTPS	ABCB1	ABCA2	SOD1	ABCA3	GCLM	ABCB4	ABCG2	TOP1	LCK
815	POSITIVE_REGULATION_OF_PEPTIDYL	http://www	IL3	IL5	IL29	LYN	HCLS1	IL31RA	IL20	CD80	CLCF1	TDGF1	CD81	IL12A
816	ECTODERM_DEVELOPMENT	http://www	ASPRV1	PPARD	FGF7	BTD	S100A7	CRABP2	FST	SPINK5	CDSN	GLI1	TGFB2	PLOD1
817	LEUKOCYTE_MIGRATION	http://www	IL8	PF4	ITGB3	CX3CL1	IL10	TGFB2	CXCR2	DOCK2	MIA3	SAA1	CD34	CKLF
818	REGULATION_OF_ENDOTHELIAL_CELL	http://www	CDH13	AGGF1	ANG	NF1	TNFSF15	GHRL	AIMP1	ATPIF1	TNFSF12	SCG2		
819	REGULATION_OF_MEMBRANE_POTENT	http://www	KCNMB3	KCNMB4	AIFM3	GLRA1	BNIP3	BCL2L1	SOD1	KCNMB2	BAK1	BCL2	BAX	CHRNA1
820	REGULATION_OF_CELL_GROWTH	http://www	DLC1	DCBLD2	BMP10	DERL2	BCAR1	PML	OSGIN1	CDHR2	PPT1	TGFB1	TGFB2	ACVR1B
821	REGULATION_OF_VIRAL_REPRODUCTI	http://www	TARBP2	CCL3	HS3ST5	RSF1	IL8	PPIA	MDFIC	VAPB	APOBEC3C	TOP2A	APOBEC3F	TNIP1
822	G1_S_TRANSITION_OF_MITOTIC_CELL_	http://www	BCAT1	PPP6C	DBF4	RCC1	LATS2	CUL3	ACVR1B	CUL2	CUL5	CDKN2A	CDKN2C	CDKN2D
823	CASPASE_ACTIVATION	http://www	TNFSF15	PMAIP1	MOAP1	NLRC4	CDKN2A	CASP8AP2	CASP9	MTCH1	DIABLO	HIP1	COL4A3	AIFM3
824	CELLULAR_RESPIRATION	http://www	UQCRC2	OXA1L	ME3	UQCRC1	ACO2	CYCS	PPARGC1A	PDHB	SDHA	SDHB	SLC25A14	NDUFS4
825	INDUCTION_OF_APOPTOSIS_BY_INTRA	http://www	AIFM1	TP53	LGALS12	CIDEB	PML	CIDEA	IFI16	SFN	CHEK2	BRCA1	TP73	C16orf5

機能解析

やってみましょう。入力ファイルは「デスクトップ – hoge – functional_analysis」にあります。exon-levelではなく**gene-levelカウントデータのほう**です。

- 解析 | 発現変動 | エクソン | DEXseq(Anders_2012) (last modified 2014/06/23)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について (last modified 2015/02/27) NEW
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA(Wang_2014) (last modified 2015/02/27)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GOseq(Young_2010) (last modified 2010/11/26)
- 解析 | 機能解析 | パスウェイ(Pathway)解析 | について (last modified 2015/02/27) NEW

解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | SeqGSEA (Wang_2014) NEW

SeqGSEAを用いてGO解析を行うやり方を示します。このパッケージは、exonレベルのカウントデータを入力として、発現変動遺伝子セットに

2. geneレベルのカウントデータファイル(srp017142_count_bowtie.txt)の場合:

パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn_2013) のStep2の出力結果ファイルです。SeqGSEA (Wang et al., 2014)は、exonレベル(Differential Splicing; DS)とgeneレベル(Differential Expression; DE)の2つの発現変動解析結果を統合してよりよい遺伝子セット解析(Gene Set Enrichment Analysis; GSEA; Subramanian et al., 2005)を行うという手法ですが、選択的スプライシング(Alternative Splicing; AS)が少ないあるいはない高等生物以外にも適用可能です。ここでは、geneレベルの発現変動解析のみに基づくDE-only GSEAのやり方を示します。計算時間のボトルネックになっていたexonレベルの発現変動解析を含まないので高速に計算可能なため、並べ替え回数を多くすることが可能です(500回で100分程度)。以下では"c5.bp.v4.0.symbols.gmt"の解析を行っています。並べ替えを500回行って得られた srp017142_SeqGSEA_c5bp_gene500.txtでは、FDR < 0.05を満たすGO termは10個であることがわかります。

```
in_f1 <- "srp017142_count_bowtie.txt"   #入力ファイル名を指定してin_f1に格納
in_f2 <- "c5.bp.v4.0.symbols.gmt"       #入力ファイル名を指定してin_f2に格納
out_f <- "hoge2.txt"                    #出力ファイル名を指定してout_fに格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2群のサンプル数を指定
param_perm <- 40                        #並べ替え回数を指定(数値が大きいほどより正確だがその分た

#必要なパッケージをロード
library(SeqGSEA)                        #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_f1, header=TRUE, row.names=1, sep="\t", quote="")#in_f4で指定したファ
dim(data)                               #確認してるだけです
tmp_colname <- colnames(data)           #SeqGSEAが判別可能なサンプルラベル情報に変更
```

機能解析

十数分(門田の環境では736秒)かかりますが、無事出力ファイルが出来ています。

2. geneレベルのカウントデータファイル(srp017142_count_bowtie.txt)の場合:

パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn_2013) のStep2の出力結果ファイルです。SeqGSEA (Wang et al., 2014)は、exonレベル(Differential Splicing; DS)とgeneレベル(Differential Expression; DE)の2つの発現変動解析結果を統合して よりよい遺伝子セット解析(Gene Set Enrichment Analysis; GSEA; Subramanian et al., 2005)を行うという手法ですが、選択的スプライシング(Alternative Splicing; AS)が少ないあるいはない高等生物以外にも適用可能です。ここでは、geneレベルの発現変動解析のみに基づくDE-only GSEAのやり方を示します。計算時間のボトルネックになっていたexonレベルの発現変動解析を含まないので高速に計算可能なため、並べ替え回数を多くすることが可能です(500回で100分程度)。以下では"c5.bp.v4.0.symbols.gmt"の解析を行っています。並べ替えを500回行って得られた srp017142_SeqGSEA_c5bp_gene500.txtでは、FDR < 0.05を満たすGO termは10個であることがわかります。

```
in_f1 <- "srp017142_count_bowtie.txt"
in_f2 <- "c5.bp.v4.0.symbols.gmt"
out_f <- "hoge2.txt"
param_G1 <- 3
param_G2 <- 3
param_perm <- 40

#必要なパッケージをロード
library(SeqGSEA)

#入力ファイルの読み込みとラベル情報の作成
data <- read.table(in_f1, header=TRUE
dim(data)
tmp_colname <- colnames(data)
colnames(data) <- c(paste("E", 1:para
data.cl <- c(rep(1, param_G1), rep(2,

#本番(Differential expression (DE) an
time_s <- proc.time()
DEG <- runDESeq(data, as.factor(data.
DEGres <- DENBStat4GSEA(DEG)
permuteMat <- genpermuteMat(as.factor
```

R Console

```
> GS <- loadGenesets(in_f2, rownames(data), geneID.type = "e$
> GS <- GSEnrichAnalyze(GS, gene.score, gene.score.perm, wei$
> time_e <- proc.time()                    #計算時間を計測する$
> time_e - time_s                          #計算時間を表示(単$
   ユーザ    システム        経過
    533.78      14.44     736.11
>
> #ファイルに保存
> #tmp <- GSEAresultTable(GS, GSDesc = TRUE)#SeqGSEA実行結果$
> tmp <- topGeneSets(GS, n=length(GS@GSNames), sortBy="FDR")$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.n$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/functional_analysis"
> list.files()
[1] "c5.bp.v4.0.symbols.gmt"     "hoge2.txt"
[3] "rcode_ORA_basic.txt"        "srp017142_count_bowtie.txt"
>
```

機能解析

以下の結果では、FDR < 0.05を満たすのは15 gene setsであることが分かります(ヒトによって若干違います)。

hoge2.txt

	A	B	C	D	E	F	G
1	GSName	GSSize	ES	ES.pos	pval	FDR	FWER
2	GLYCOLIPID_METABOLIC_PROCESS	16	1.250307	7456	0.125	0	0.375
3	MONOCARBOXYLIC_ACID_TRANSPORT	10	1.260423	10098	0.1	0	0.25
4	MITOTIC_SISTER_CHROMATID_SEGREGATION	16	1.353241	7100	0.125	0	0.125
5	VITAMIN_TRANSPORT	13	1.281971	4862	0.1	0	0.25
6	ENDOTHELIAL_CELL_PROLIFERATION	12	1.250099	5702	0.1	0	0.375
7	ESTABLISHMENT_OF_ORGANELLE_LOCALIZATION	18	1.294689	7372	0.1	0	0.25
8	MORPHOGENESIS_OF_AN_EPITHELIUM	16	1.243735	4511	0.1	0	0.375
9	G1_PHASE_OF_MITOTIC_CELL_CYCLE	13	1.358698	7004	0.1	0	0.125
10	RIBONUCLEOTIDE_METABOLIC_PROCESS	14	1.296188	6923	0.1	0	0.25
11	G1_PHASE	15	1.36737	7004	0.1	0	0.125
12	REGULATION_OF_GTPASE_ACTIVITY	15	1.292059	5858	0.1	0	0.25
13	SISTER_CHROMATID_SEGREGATION	17	1.342555	7100	0.1	0	0.125
14	POSITIVE_REGULATION_OF_CELL_CYCLE	16	1.300093	6197	0.1	0	0.25
15	CHROMOSOME_CONDENSATION	10	1.33073	7817	0.1	0	0.125
16	REGULATION_OF_ENDOTHELIAL_CELL_PROLIFERATION	10	1.276724	5702	0.1	0	0.25
17	GLYCOSPHINGOLIPID_METABOLIC_PROCESS	12	1.197646	7201	0.15	0.073171	0.8
18	NEGATIVE_REGULATION_OF_INTRACELLULAR_TRANSPORT	17	1.201119	5880	0.1	0.075	0.8
19	PROTEOGLYCAN_METABOLIC_PROCESS	21	1.201798	10475	0.1	0.076923	0.8
20	REGULATION_OF_RAS_GTPASE_ACTIVITY	11	1.216984	5858	0.1	0.076923	0.8
21	CARBOXYLIC_ACID_TRANSPORT	41	1.202437	10497	0.1	0.078947	0.8
22	INTERPHASE	68	1.218318	8323	0.2	0.08	0.8

機能解析

もちろんわざわざExcelで開かなくても、R上で結果を概観できます。

2. geneレベルのカウントデータファイル(srp017142_count_bowtie.txt)の場合:

パイプライン | ゲノム | 発現変動 | 2群間 | 対応なし | 複製あり | SRP017142(Neyret-Kahn_2013) のStep2の出力結果ファイルです。SeqGSEA (Wang et al., 2014)は、exonレベル(Differential Splicing; DS)とgeneレベル(Differential Expression; DE)の2つの発現変動解析結果を統合して よりよい遺伝子セット解析(Gene Set Enrichment Analysis; GSEA; Subramanian et al., 2005)を行うという手法ですが、選択的スプライシング(Alternative Splicing; AS)が少ないあるいはない高等生物以外にも適用可能です。ここでは、geneレベルの発現変動解析のみに基づくDE-only GSEAのやり方を示します。計算時間のボトルネックになっていたexonレベルの発現変動解析を含まないので高速に計算可能なため、並べ替え回数を多くすることが可能です(500回で100分程度)。以下では"c5.bp.v4.0.symbols.gmt"の解析を行っています。並べ替えを500回行って得られた srp017142_SeqGSEA
とがわかります。

```
time_s <- proc.time()
DEG <- runDESeq(data, as.factor(data
DEGres <- DENBStat4GSEA(DEG)
permuteMat <- genpermuteMat(as.facto
DEpermNBstat <- DENBStatPermut4GSEA(
DEscore.normFac <- normFactor(DEperm
DEscore <- scoreNormalization(DEGres
DEscore.perm <- scoreNormalization(D

#本番(Integrative GSEA; Wang and Cair
gene.score <- geneScore(DEscore, DEw
gene.score.perm <- genePermuteScore(
GS <- loadGenesets(in_f2, rownames(d
GS <- GSEnrichAnalyze(GS, gene.score
time_e <- proc.time()
time_e - time_s

#ファイルに保存
#tmp <- GSEAresultTable(GS, GSDesc =
tmp <- topGeneSets(GS, n=length(GS@G
write.table(tmp, out_f, sep="\t", ap
```

R Console

```
> head(tmp)
                                    GSName GSSize       ES
52              GLYCOLIPID_METABOLIC_PROCESS     16 1.250307
209           MONOCARBOXYLIC_ACID_TRANSPORT     10 1.260423
239    MITOTIC_SISTER_CHROMATID_SEGREGATION     16 1.353241
242                      VITAMIN_TRANSPORT     13 1.281971
353         ENDOTHELIAL_CELL_PROLIFERATION     12 1.250099
385 ESTABLISHMENT_OF_ORGANELLE_LOCALIZATION     18 1.294689
    ES.pos  pval FDR  FWER
52    7456 0.125   0 0.375
209  10098 0.100   0 0.250
239   7100 0.125   0 0.125
242   4862 0.100   0 0.250
353   5702 0.100   0 0.375
385   7372 0.100   0 0.250
> sum(tmp$FDR < 0.05)
[1] 15
>
```

謝辞

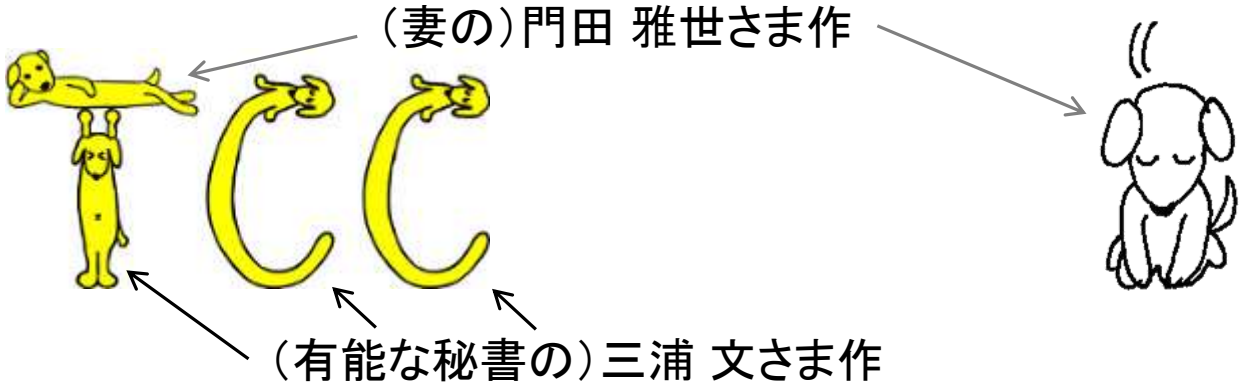
共同研究者

清水 謙多郎　先生（東京大学・大学院農学生命科学研究科）
西山 智明　先生（金沢大学・学際科学実験センター）
孫 建強　氏（東京大学・大学院農学生命科学研究科・大学院生）
湯 敏　氏（東京大学・大学院農学生命科学研究科・大学院生）

グラント

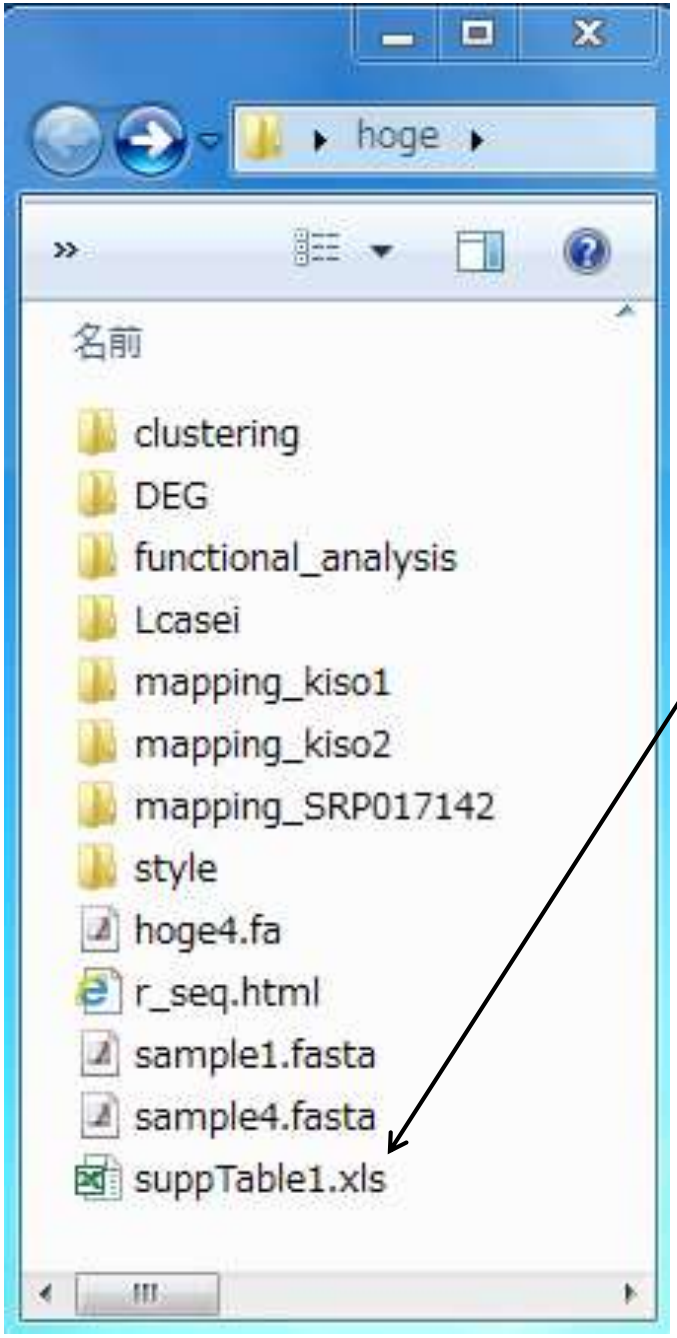
- 基盤研究(C)（H24−26年度）:「シークエンスに基づく比較トランスクリプトーム解析のためのガイドライン構築」(代表)
- 新学術領域研究(研究領域提案型)(H22−26年度):「非モデル生物におけるゲノム解析法の確立」(分担;研究代表者:西山智明)
- NBDCとの共同研究(H26年度)

挿絵やTCCのロゴなど



おまけ

論文の付録で公開されているカウントデータを解析する。



データ取得

GEOのサイトからもカウントデータが提供されている。GSE17274で検索。

- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (las
- 書籍
- 書籍
- 書籍

サンプルデータ NEW

1.

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

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

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

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

データ取得

①xls形式のエクセルファイルを通常の手順で読み込むことができる。②それほど大きなサイズでなければ、ネットワーク経由で直接読み込むこともできる。

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

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

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

#サブセットの取得
data <- cbind(                              #必要な列名の情
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, ho
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, ho
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, ho
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, ho
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, ho
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, ho
colnames(data) <- c(                        #列名を付加
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L2.HSF2"
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L8.HSM2"
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF2"
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L6.PTM2"
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF2"
```

①

②

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
 [1] "clustering"          "DEG"
 [3] "functional_analysis" "hoge4.fa"
 [5] "Lcasei"              "mapping_kiso1"
 [7] "mapping_kiso2"       "mapping_SRP017142"
 [9] "r_seq.html"          "sample1.fasta"
[11] "sample4.fasta"       "style"
[13] "suppTable1.xls"
>
```

データ取得

①出力ファイルは20,689遺伝子×36サンプルのカウントデータ。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種のデータ。各12サンプル。ここのコードは、予め列名を把握しておき、②欲しいサブセットのみ任意の列の順番で並べ替えて、③任意の列名を与えて出力させています。

41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Supplementary Table1で提供されている
(http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls; 約4.3MB)からカウン
に整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ(samp
験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS), チンパンジー(Pan troglod
macaque; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3
れており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各
が取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サン
の計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトの
トのオス(HSM1, HSM2, HSM3)」,「チンパンジーのメス(PTF1, PTF2, PTF3)」,「チンパンジーのオス(PTM1,
のメス(RMF1, RMF2, RMF3)」,「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものを
う少し美しくやることも原理的には可能ですが、そこは本
手順で行っています。当然ながら、エクセルなどでファ
尚、"R1L4.HSF1"と"R4L2.HSF1"が「HSF1というヒトのメ

```
#in_f <- "http://genome.cshlp.org/content
in_f <- "suppTable1.xls"                    #
out_f <- "sample_blekhman_36.txt"           #出

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

#サブセットの取得
data <- cbind(                              #必
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2
colnames(data) <- c(                        #列
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2",
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2",
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2",
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2",
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2",
```

②

R Console

①

```
> dim(data)                                  #行数と列数を$
[1] 20689    36
>
> #ファイルに保存(テキストファイル)
> tmp <- cbind(rownames(data), data)         #保存したい情$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, $
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
 [1] "clustering"              "DEG"
 [3] "functional_analysis"     "hoge4.fa"
 [5] "Lcasei"                  "mapping_kiso1"
 [7] "mapping_kiso2"           "mapping_SRP017142"
 [9] "r_seq.html"              "sample_blekhman_36.txt"
[11] "sample1.fasta"           "sample4.fasta"
[13] "style"                   "suppTable1.xls"
>
```

③

クラスタリング

①出力ファイルは20,689遺伝子×36サンプルのカウントデータ。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種のデータ。各12サンプル。サンプル間クラスタリング結果から、実験デザインに関する説明を行います。

- 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 (Si ... 4) (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」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

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

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

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

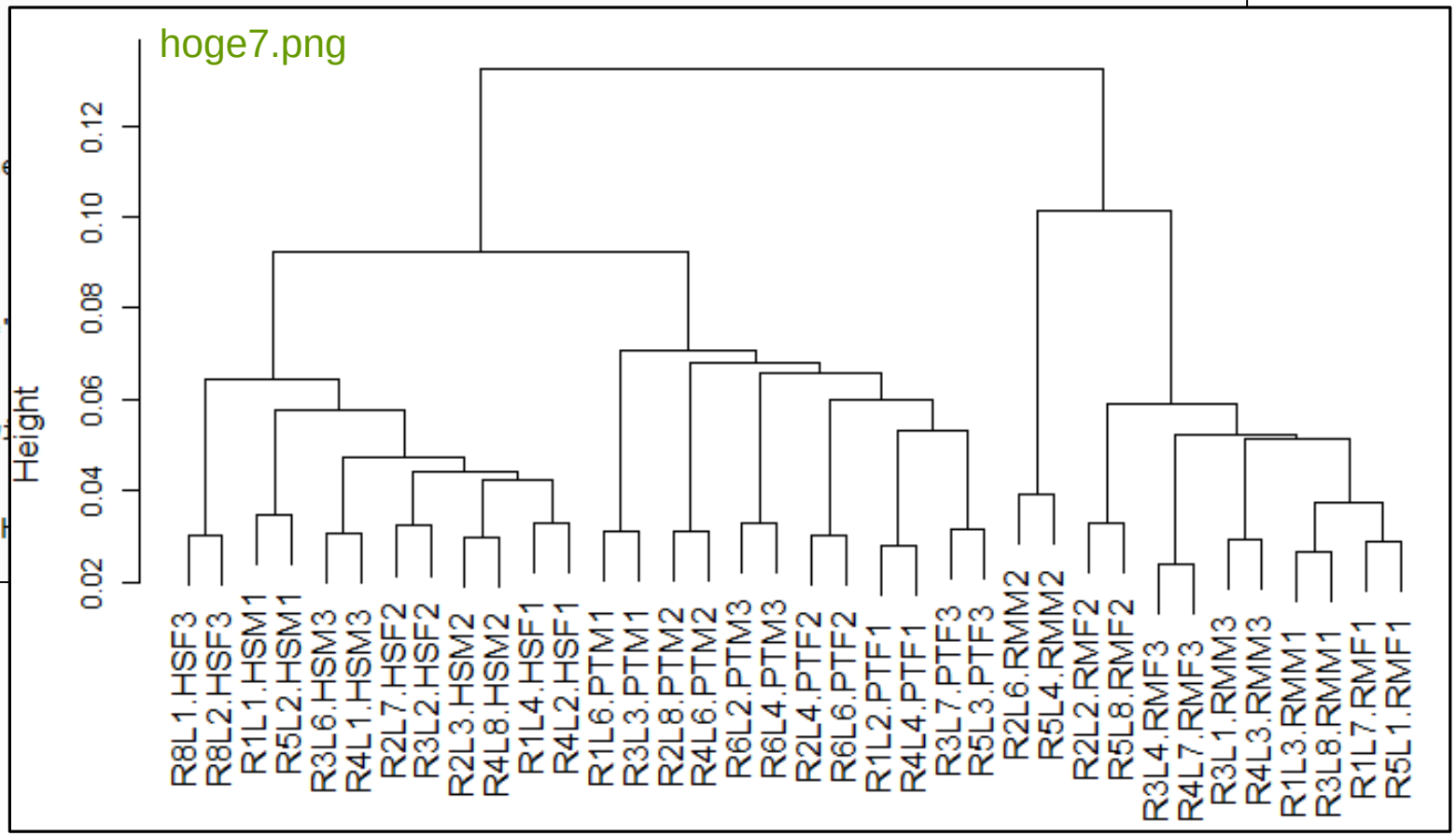
クラスタリング

①出力ファイルは20,689遺伝子×36サンプルのカウントデータ。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種のデータ。各12サンプル。

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

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

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

#必要なパッケージをロード
library(TCC)

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

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

#ファイルに保存
png(out_f, pointsize=13, wi
par(mar=c(0, 4, 1, 0))
plot(out, sub="", xlab="",
  cex=1.3, main="", ylab="H
dev.off()
```

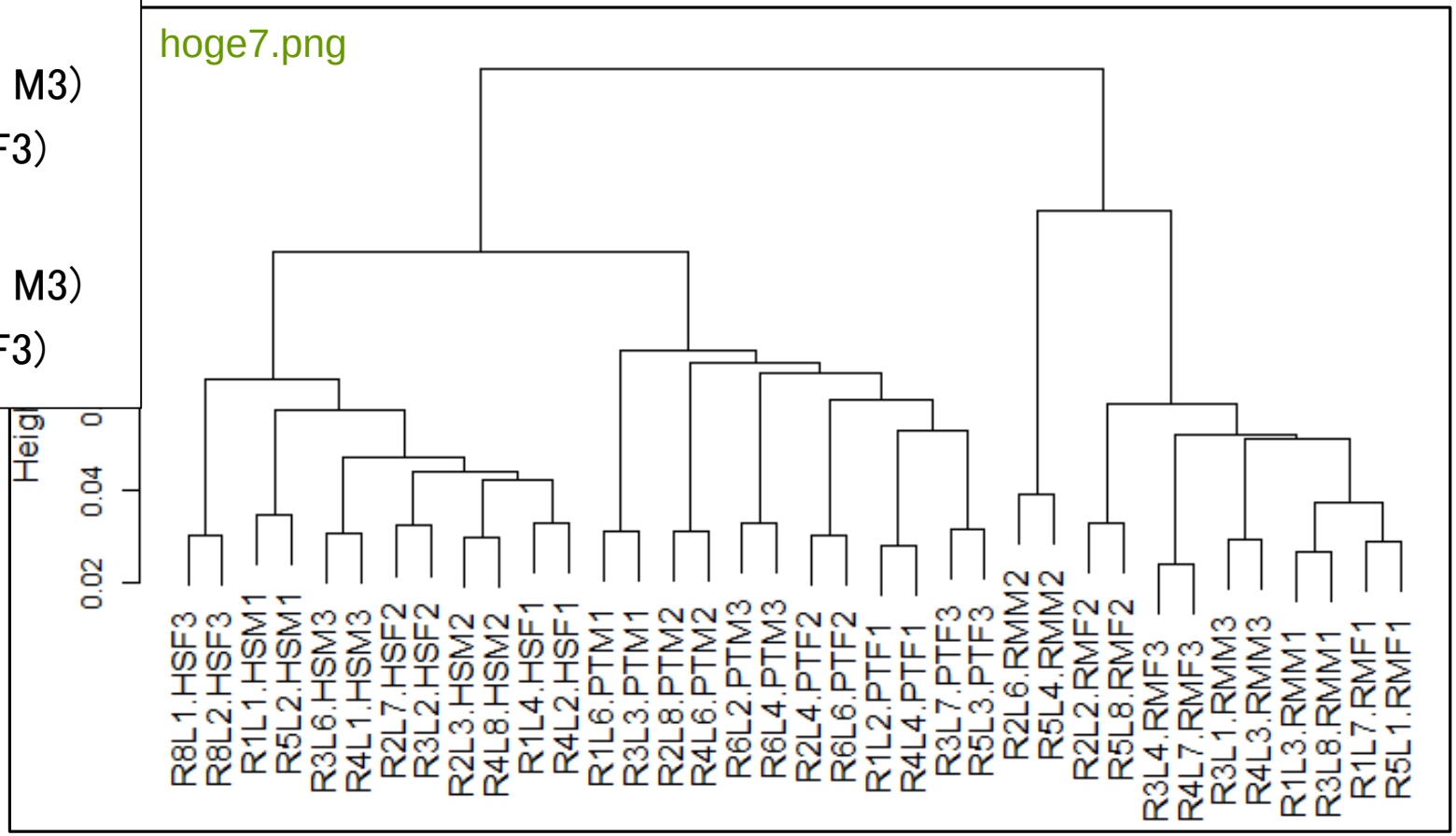


クラスタリング

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

生物種ごとに用いた個体数は6。雄雌を考慮しなければbiological replicates(生物学的な反復)は6。個体ごとにサンプルを分割して得たデータがあり、technical replicates(技術的な反復)は2。これらを合わせることで生物種ごとに計12サンプルになる。

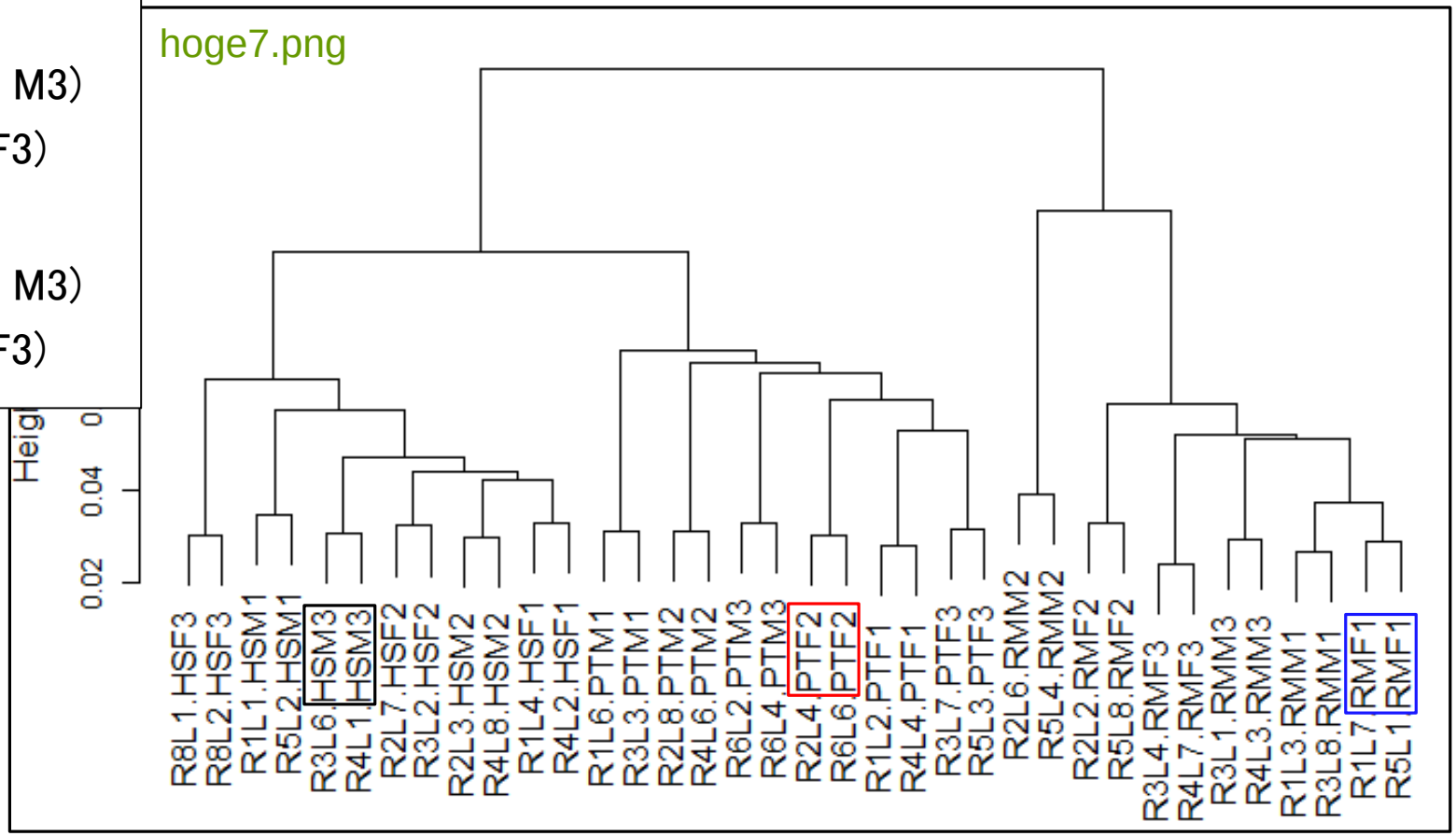
hoge7.png



クラスタリング

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

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

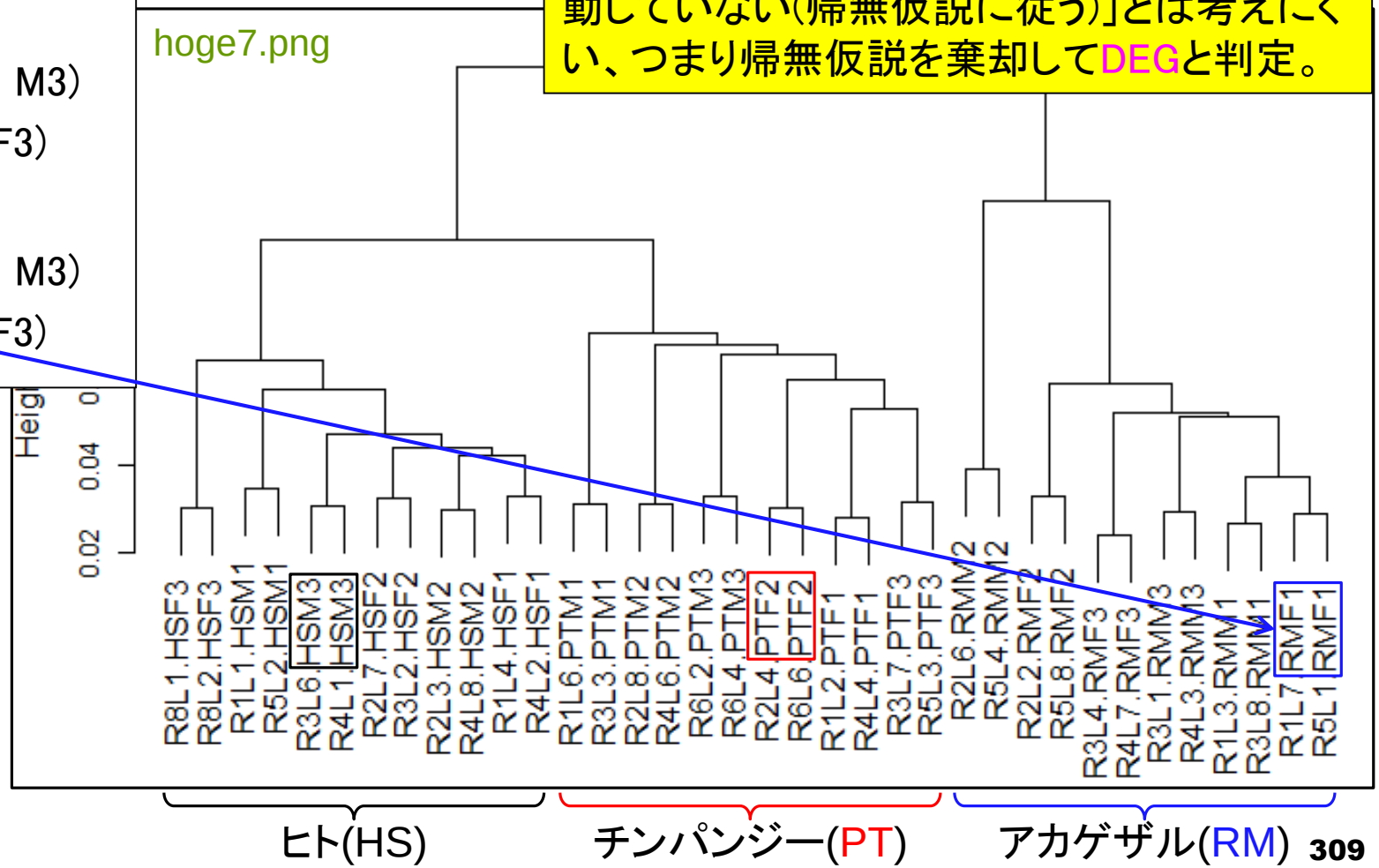


クラスタリング

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

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

hoge7.png



データ取得

統計的手法の多くは、biological replicatesのデータを前提としている。Technical replicatesのデータをマージ(merge; collapseともいうらしい)したものを作成。サンプル名部分は余計なものを削除している。

- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (las
- 書籍
- 書籍
- 書籍

サンプルデータ NEW

42. Blekhman et al., Genome Res., 2010のリアルカウントデータです。
1つ前のsample41.txtとは違って、technical replicatesの2列分のデータは足して1列分のデータとしています。20,689 genes×18 samplesのカウントデータ(sample_blekhman_18.txt)です。

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

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

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

クラスタリング

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

- 解析 | 発現量推定(トラン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 modifie 2015/02/26) NEW
- 解析 | クラスタリング | サンプル間 | TCC(Sun_2013) modified 2015/03/02) NEW
- 解析 | クラスタリング | 遺伝子間 | MBCluster.Seq (Si 14) (last modified 2014/02/05)
- 解析
- 解析

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」-「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、以下をコピペ

1. 59,

Neyret-
ン | ゲノ

```
in_f
out_f
param

#必要
libra

#入力
data
dim(d

#本番
out <
```

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。

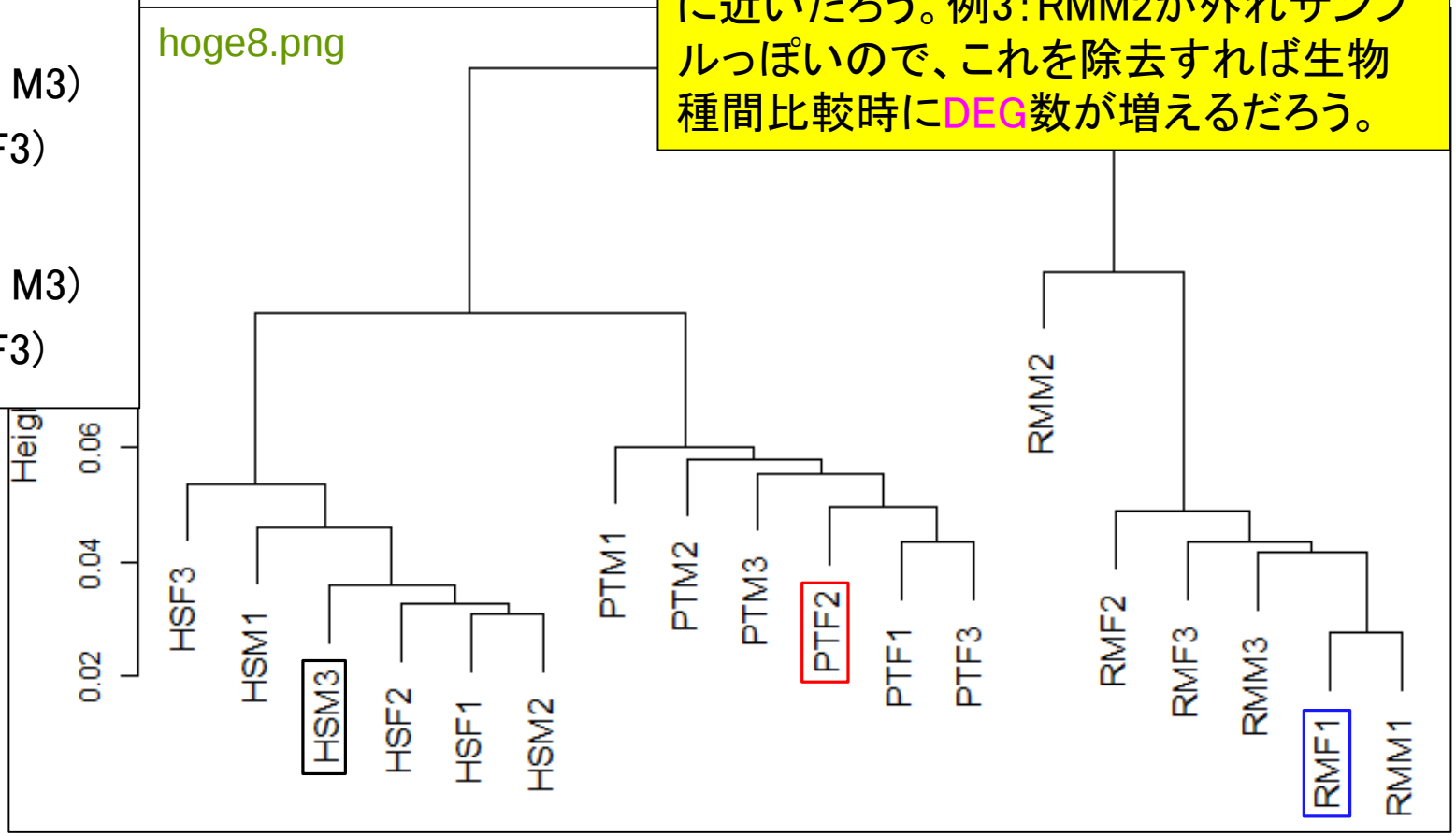
```
in_f <- "sample_blekhman_18.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.png"                      #出力ファイル名を指定してout_fに格納
param_fig <- c(700, 400)                  #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#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, sep="\t", quote="")#in_fで指定したファイ
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
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
```

クラスタリング

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

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

hoge8.png



3群間比較

オス・メスの違いを気にせずに、3生物種間比較を行うやり方です。

- 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 | 3群間 | 対応なし | について (last modified 2015/02/10) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | DESeq2(Love 201… (last modified 2015/02/04) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | TCC(Sun 2013) …modified 2015/02/03)推奨 NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | EBSeq(Leng 2013 …last modified 2015/02/10) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | SAMseq(Li 2013) (last modified 2015/02/10) NEW
- 解析
- 解析

解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | TCC(Sun_2013) NEW

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GLM LRT法で発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeRという解析パイプラインに相当します。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…出」です。

「ファイ…

1. サンプル…

シミュレー…
(gene_1~…
群で6倍…

```
in_f <
out_f
param_
param_
param_
param_

#必要な
librar

#入力フ
data <
```

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で指定したファイ

#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
tcc <- new("TCC", data, data.cl)          #TCCクラスオブジェクトtccを作成
```

3群間比較

①発現パターンごとの分類もしたい場合に便利。②post-hoc test的なことをやりたいときの項目。

- 解析 | 発現変動 | 2群間 | 対応なし | NBPSeq(Di 2011) (last modified 2012/03/15)
- 解析 | 発現変動 | 2群間 | 対応あり | について (last modified 2014/12/27)
- 解析 | 発現変動 | 2群間 | 対応あり | 複製なし | TCC中のDEGES/edgeR-edgeR(Sun 2013) (last modified 2014/03/13)
- 解析 | 発現変動 | 2群間 | 対応あり | 複製なし | TCC中のDEGES/DESeq-DESeq(Sun 2013) (last modified 2014/03/13)
- 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 | 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 | 3群間 | 対応なし | について (last modified 2015/02/10) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | DESeq2(Love 2014) (last modified 2015/02/04) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | TCC(Sun 2013) (last modified 2015/03/04)推奨 NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | EBSeq(Leng 2013) (last modified 2015/02/10) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | SAMseq(Li 2013) (last modified 2015/02/10) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | edgeR(Robinson 2010) (last modified 2015/02/03) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 複製あり | 基礎編 | 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 | 応用編 | TCC(Sun 2013) (last modified 2015/01/29)推奨
- 解析 | 発現変動 | 5群間 | 対応なし | 複製あり | TCC(Sun 2013) (last modified 2014/08/22)推奨
- 解析 | 発現変動 | 時系列 | について (last modified 2014/12/19)
- 解析 | 発現変動 | 時系列 | Bayesian model-based clustering(Nascimento 2012) (last modified 2012/09/10)
- 解析 | 発現変動 | 時系列 | maSigPro(Nueda 2014) (last modified 2014/07/18)
- 解析 | 発現変動 | エクソン | について (last modified 2015/01/29)
- 解析 | 発現変動 | エクソン | DEXseq(Anders 2012) (last modified 2014/06/23)

シミュレーションデータ

他にも多くの解析用パッケージが存在し、このウェブページ上で紹介しきれていないものが多く存在します。また、バージョンアップなどに追い付いていない項目も多くあります。そのため、正しい手順で解析できているのかが不安な局面があるでしょう。そういうときはTCCパッケージ中のシミュレーションデータ作成関数を利用して、「これがDEG検出結果の上位に来ていないやり方はオカシイはず」というようなデータを自分で作成して検証するのです。

- 解析 | クラスタリング | 遺伝子間 | MBCluster.Seq (Si 2014) (last modified 201
- 解析 | シミュレーションカウントデータ | について (last modified 2015/01/25)
- 解析 | シミュレーションカウントデータ | Technical rep.(ポアソン分布) (last modified 2015/01/23)
- 解析 | シミュレーションカウントデータ | Biological rep. | 基礎編 (last modified 2015/01/23)
- 解析 | シミュレーションカウントデータ | Biological rep. | 2群間 | 基礎編 | TCC(Sun 2013) (last modified 2015/01/28)
- 解析 | シミュレーションカウントデータ | Biological rep. | 2群間 | 応用編 | TCC(Sun 2013) (last modified 2015/01/28)
- 解析 | シミュレーションカウントデータ | Biological rep. | 3群間 | 基礎編 | TCC(Sun 2013) modified 2015/01/28)
- 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 2015/02/02)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC(Sun 2013) (last modified 2015/02/26)推奨 NEW