2016.07.06版

スライド10までは自習。当日はスライド11から始める予定です。スライド122-133のPacBioのイントロ部分は、疲れたころに初耳状態で聞くと相当疲弊すると思われます。ここも当日は復習程度の位置づけで聞けるよう全体像を理解し頭を整理しておきましょう

第3部：NGS解析（中～上級）

～ クラウド環境との連携、ロングリードデータの解析 ～

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/

忘れたらココを思い出してね

利用プログラムの簡単な解説

- DDBJ Pipeline (Nagasaki et al., *DNA Res.*, 2013)
  - マッピングや*de novo*アセンブリを行ってくれるウェブツール（クラウド解析環境）
- FaQCs (Lo and Chain, *BMC Bioinformatics*, 2014)
  - Quality Control用プログラム。クオリティフィルタリングやアダプター除去が主目的
- FastQC
  - Quality Control用プログラム。アダプターの混入などNGSデータのクオリティチェックが主目的
- HGAP (Chin et al., *Nat Methods*, 2013)
  - PacBio用*de novo*ゲノムアセンブラ。.bax.h5ファイルのみを入力として受け付ける
- KmerGenie (Chikhi and Medvedev, *Bioinformatics*, 2014)
  - *de novo*ゲノムアセンブリ時に用いる最適なk値を算出してくれる。推定ゲノムサイズも返す
- Platanus (Kajitani et al., *Genome Res.*, 2014)
  - *de novo*ゲノムアセンブラ。複数のk値を利用するので、KmerGenieとは無関係
- SRA Toolkit
  - sra形式ファイル処理用のプログラム群。FASTQに変換するfastq-dump利用が主目的
- Velvet (Zerbino and Birney, *Genome Res.*, 2008)
  - *de novo*ゲノムアセンブラ。単一のk値を利用するので、KmerGenieと関係あり

おさらい

解析データは、①乳酸菌(*L. hokkaidonensis* LOOC260^{T})ゲノム塩基配列。②原著論文

http://www.ncbi.nlm.nih.gov/pubmed/25879859

Complete genome sequ...

NCBI Resources How To Sign in to NCBI

PubMed.gov
US National Library of Medicine
National Institutes of Health

PubMed Search
Advanced Help

Abstract Send to:

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

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

Tanizawa Y[1,2], Tohno M[3], Kaminuma E[4], Nakamura Y[5], Arita M[6,7].

Author information

Abstract

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

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

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

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

Full text links
Read free full text at BioMed Central
PMC FREE Full text

Save items
Add to Favorites

Similar articles
Lactobacillus hokkaidonensis sp. no [Int J Syst Evol Microbiol. 2013]
Insights into the completely annotated ger [J Biotechnol. 2012]
Lactobacillus wasatchensis sp. no [Int J Syst Evol Microbiol. 2015]
Review Genomic organization of [Antonie Van Leeuwenhoek. 1996]
Review Low-redundancy [Antonie Van Leeuwenhoek. 1999]
See reviews...
See all...

Related information

①

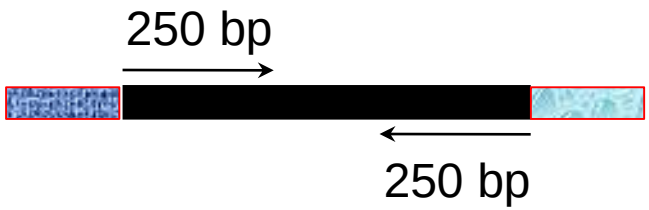
②

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

おさらい

①乳酸菌ゲノム配列決定論文は、2種類のNGS機器から得られたデータを併用している。第6回はIllumina MiSeqデータ(DRR024501)を、そして第7回はPacBioデータを取り扱っている

- PacBio RS IIデータ(DRR024500)
  - DRR024500は登録内容に問題があったことが判明し消滅
  - 4セル分のデータ。DRR054113-054116に差し替えられている
  - セルあたり約15万リード。4セル分なので約60万リード
- Illumina MiSeqデータ(DRR024501)
  - paired-endゲノムデータ
  - リード長は、forward側とreverse側共に250 bp
  - オリジナルは2,971,310リード。最初の300,000リードを解析
  - forward側(DRR024501sub_1.fastq.gz)
  - reverse側(DRR024501sub_2.fastq.gz)
- FaQCs実行結果(第6回W5-4)
  - 300,000リード → 297,633リード (W5-2)
  - forward側(QC.1.trimmed.fastq.gz)
  - reverse側(QC.2.trimmed.fastq.gz)
  - ファイルの場所:~/Documents/DRR024501/result



①

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

第6回原稿PDFのp45

尚、このデータの正解は、配列数が3（1 chromosome + 2 plasmids）、2,400,586 bp（約2.4MB）である[4]。k値の選択の重要性がよくわかる例といえよう。

通常、Velvetを実行する場合は複数の異なるk値を用いてアセンブルを行い、それらの結果を眺める［W10］。ここでは、計10個のk値（k=31, 61, 91, 111, 121, 131, 151, 171, 181, 191）で実行した結果を眺め、主に配列数の観点から、k=171周辺の結果が一番よさそうだと解釈する。もちろんこのデータの場合は、「真のゲノムサイズは約2.4MB」だという答えがわかった状態でアセンブリ結果の評価を行っていることになるが、実際には近縁種との比較により妥当と考えられるゲノムサイズを検討する。ここではそのような情報が得られなかったと仮定して「ゲノムサイズ推定」を行い、アセンブリ結果の評価を行う。

ゲノムサイズ推定 ①

ゲノムサイズの推定は、フローサイトメトリー（flow cytometry）という手法を用いて実験的に求めるやり方

②

複数のk値でVelvetアセンブリを行い、主観でk=171がいいと判断したところまでが2016.08.02の内容。今日は、Velvetなど単一のk値を指定してアセンブリを実行する際に、客観的に最もよいと思われるk値を出力してくれるKmerGenieのインストールの話からスタート。KmerGenieは、makeコマンドを利用してインストールする2例目として、また最適k値と同時に出力する①ゲノムサイズ推定周辺を主目的として紹介。KmerGenie出力結果の一部には、2016.07.20で眺めたk-mer出現頻度分布もある。②はゲノムサイズ推定の意義について…

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

ゲノムサイズ推定

②ゲノムサイズ推定、からスタート。③KmerGenie。スライドを見るだけ

(Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデ

(last modified 2016/06/03, since 2011)

What's n

- このウ
  リーソ
  法(Wi

- 書籍 | 日本乳酸菌学会誌 | 第3回Linux環境構築からNGSデータ取得まで (last modified 2015/1
- 書籍 | 日本乳酸菌学会誌 | 第4回クオリティコントロールとプログラムのインストール (last modif
- 書籍 | 日本乳酸菌学会誌 | 第5回アセンブル、マッピング、そしてQC (last modified 2016/06/13)
- 書籍 | 日本乳酸菌学会誌 | 第6回ゲノムアセンブリ (last modified 2016/06/15) NEW
- 書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ (last modified 2016/05/12)
- イントロ
- イントロ
- イントロ
- イントロ

① ②

書籍 | 日本乳酸菌学会誌 | 第6回ゲノムアセンブリ NEW

日本乳酸菌学会誌の第6回分です。Linuxコマンドのリンク先は主に日経BP社様です。原稿PDFの「はじめに」項目のところにtypoがあります。誤:するであれば、正:する"の"であれば、ですねm(_ _)m。また、「配列長によるフィルタリング」項目の最後の文章「これらについては、第7回で詳述する予定である。」についてですが、これは「これらについては、"第8回以降"で詳述する予定である。」と読み替えてください。第7回ドラフト 原稿作成時点
加)。

- 原稿PDF
- ウェブ資料PDF
  - Windows用(2016.03.29版;
  - Macintosh用
- (共有フォルダ設定情報を含む)連
  - Windows用(2015.12.28版;
  - Macintosh用(2015.12.28版

ゲノムサイズ推定

- ゲノムサイズの調べ方(OKWAVE)
- Jerryfish: Marçais and Kingsford, Bioinformatics, 2011
- KmerGenie: Chikh and Medvedev, Bioinformatics, 2014
- KmerStream: Melsted and Halldórsson, Bioinformatics, 2014
- KmerGenieダウンロードと解凍[W11-2]

③

```
cd ~/Downloads
ls
wget -cq http://kmergenie.bx.psu.edu/kmergenie-1.6982.tar.gz
#cp ~/Desktop/backup/kmergenie-1.6982.tar.gz .
ls -l kmer*
tar -zxvf kmergenie-1.6982.tar.gz
ls -ld kmer*
```

W11-1：KmerGenie

2016年6月15日現在のKmerGenieのバージョンは1.7016だが、第6回ウェブ資料作成当時（2016年1月5日）のKmerGenie（ver. 1.6982）をダウンロードできるのでそれで説明する。
①で右クリックし、②ショートカットのコピーで、wgetでダウンロードするときに必要なURL情報を取得できる。講習会ではver. 1.6982のtar.gzファイルをダウンロード済みですが、サイズは約400KBなので、時間をずらして休憩時間に最新版をダウンロードしてみてもよい

Software: KmerGenie

KmerGenie estimates the best k-mer length for genome de novo assembly. Given a set of reads, KmerGenie first computes the k-mer abundance histogram for many values of k. Then, for each value of k, it predicts the number of distinct genomic k-mers in the dataset, and returns the k-mer length which maximizes this number. Experiments show that KmerGenie's choices lead to assemblies that are close to the best possible over all k-mer lengths.
KmerGenie predictions can be applied to single-k genome assemblers (e.g. Velvet, SOAPdenovo 2, ABySS, Minia). However, multi-k genome assemblers (e.g. SPAdes, IDBA) generally perform better with default parameters (using multiple k values), rather than the single best k predicted by KmerGenie.

See a sample report generated by KmerGenie from a dataset of bacterial reads.

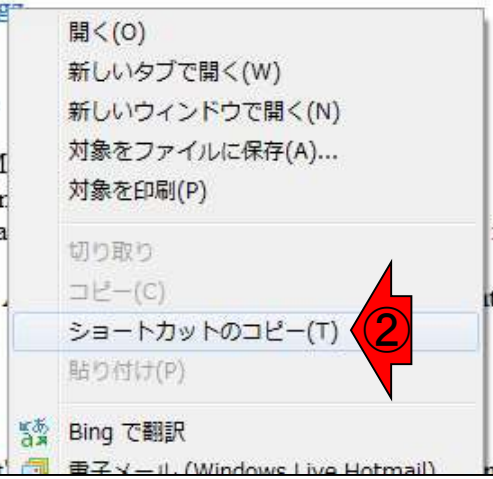
Download

①

Download KmerGenie sources here: kmergenie-1.6982.tar.gz
You will need Python and R.

Latest README and CHANGELOG. Major changes since

- 1.6213 (1/30/14): Advanced Help section in the HTM
- 1.5621 (8/02/13): HTML report for easier results exar
- 1.5378 (6/25/13): Suitable histogram resolution (-e pa for small (bacterial) genomes.
- 1.5260 (5/26/13): Improved model R code (thanks to tion. Histograms are automatically plotted.



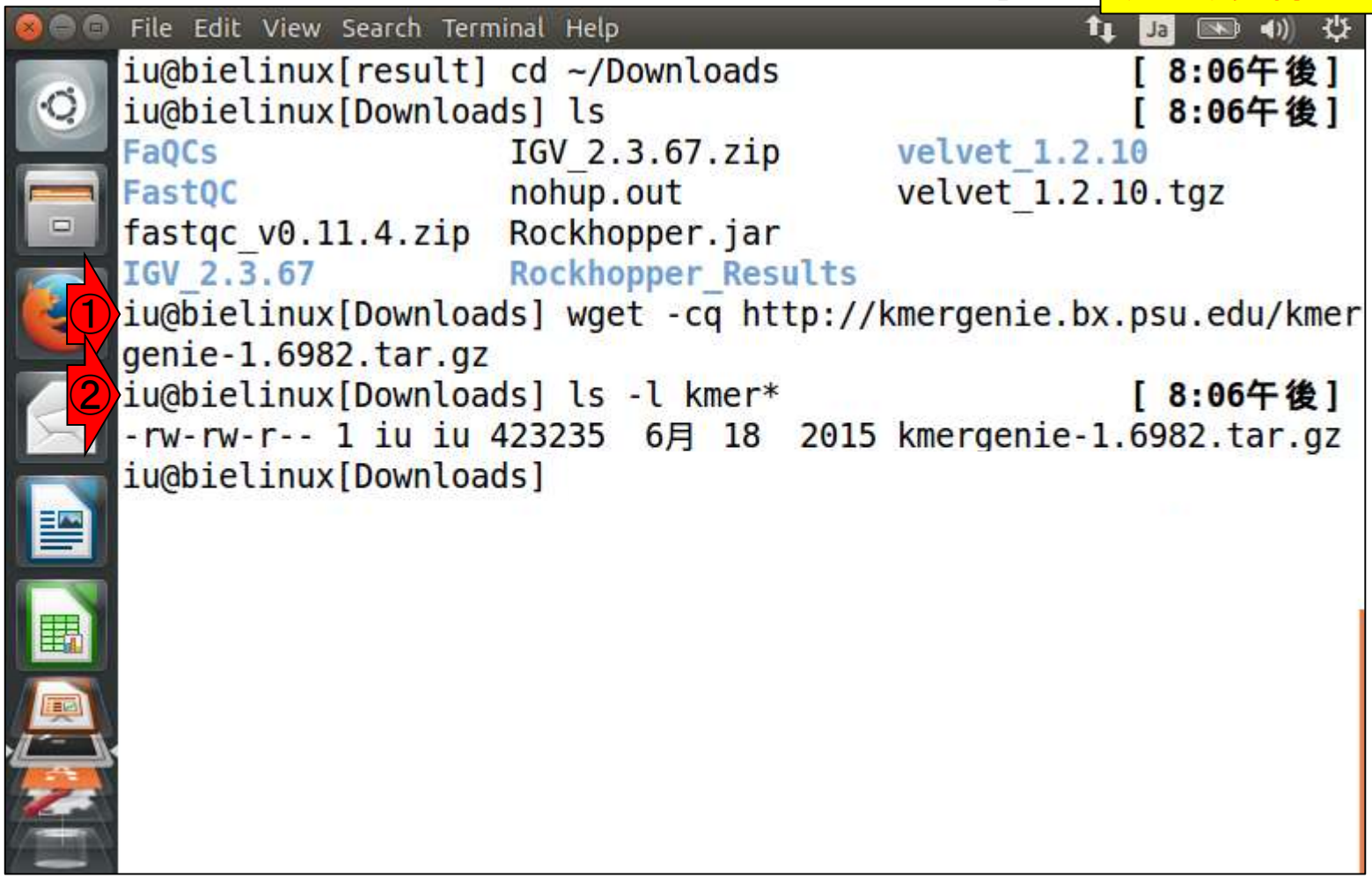
Support

Please use Biostars (dynamic FAQ system, click "New Post" ny question (An

Chikhi and Medvedev, *Bioinformatics*, **30**: 31-37, 2014

W11-2:ダウンロードと解凍

①ダウンロードと解凍。講習会ではver. 1.6982のtar.gzファイルをダウンロード済み。実際に行うのは、②の確認のみ



ゲノムサイズ推定

もし、.tar.gzファイルを間違って消してしまったり、その後のwgetでしくじった場合は、①でリカバリしてください

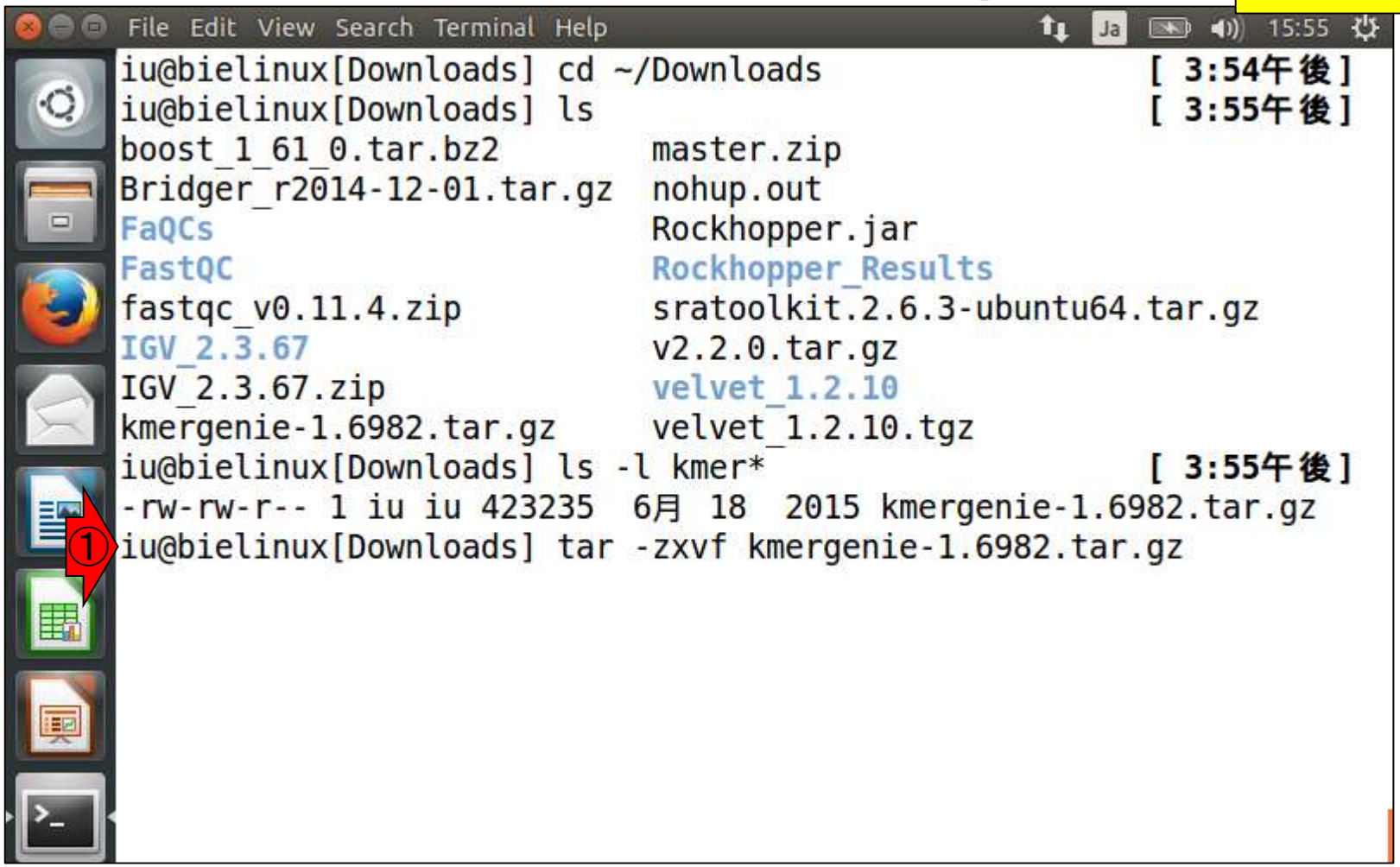
ゲノムサイズ推定

- ゲノムサイズの調べ方(OKWAVE)
- Jerryfish: Marçais and Kingsford, Bioinformatics, 2011
- KmerGenie: Chikh and Medvedev, Bioinformatics, 2014
- KmerStream: Melsted and Halldórsson, Bioinformatics, 2014
- KmerGenieダウンロードと解凍[W11-2]

```
cd ~/Downloads
ls
wget -cq http://kmergenie.bx.psu.edu/kmergenie-1.6982.tar.gz
#cp ~/Desktop/backup/kmergenie-1.6982.tar.gz .
ls -l kmer*
tar -zxvf kmergenie-1.6982.tar.gz
ls -ld kmer*
```

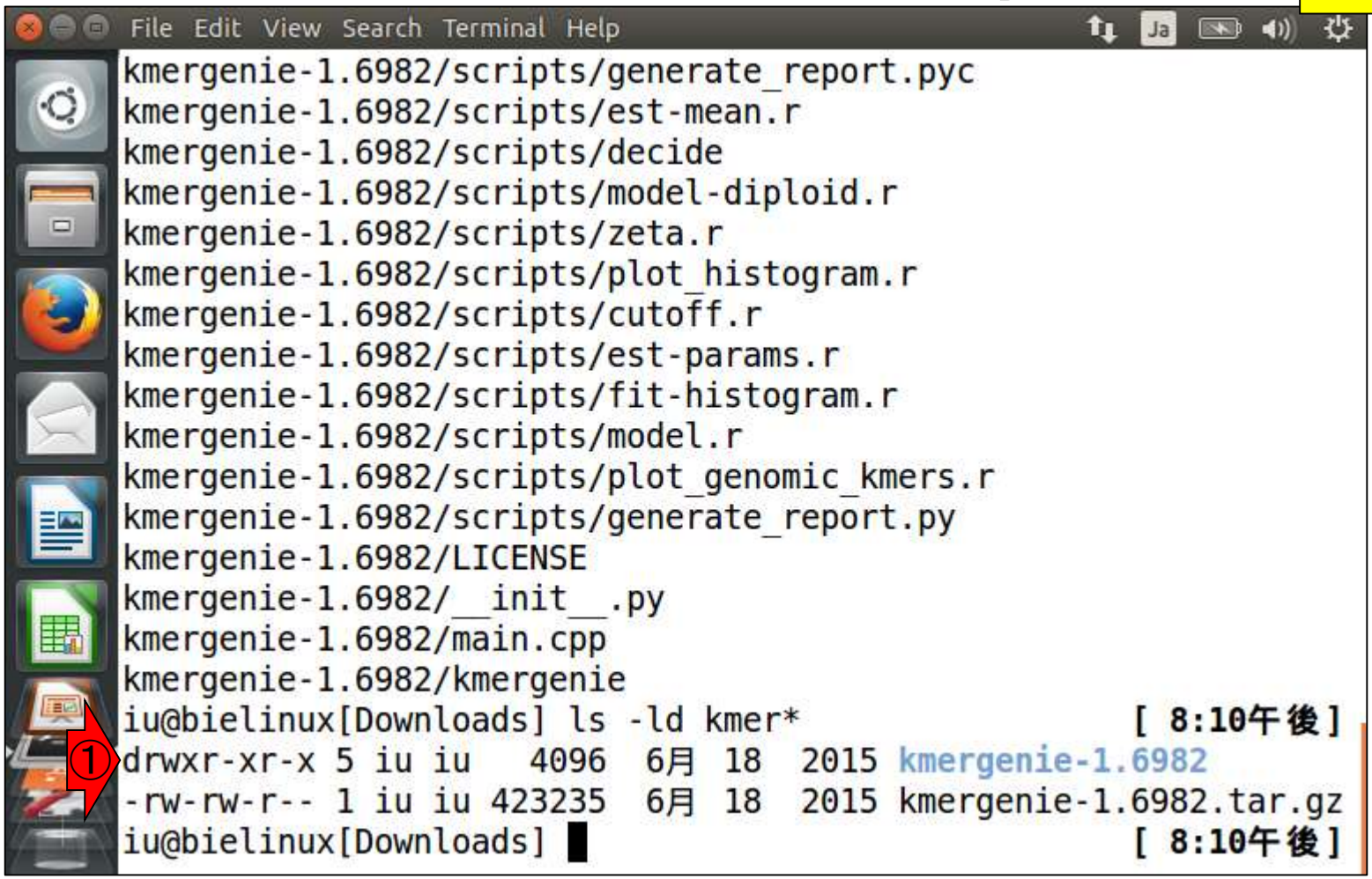
W11-2:ダウンロードと解凍

①解凍。.tar.gzファイルなので、解凍コマンドはW9-3と同じ。
ここから手を動かす



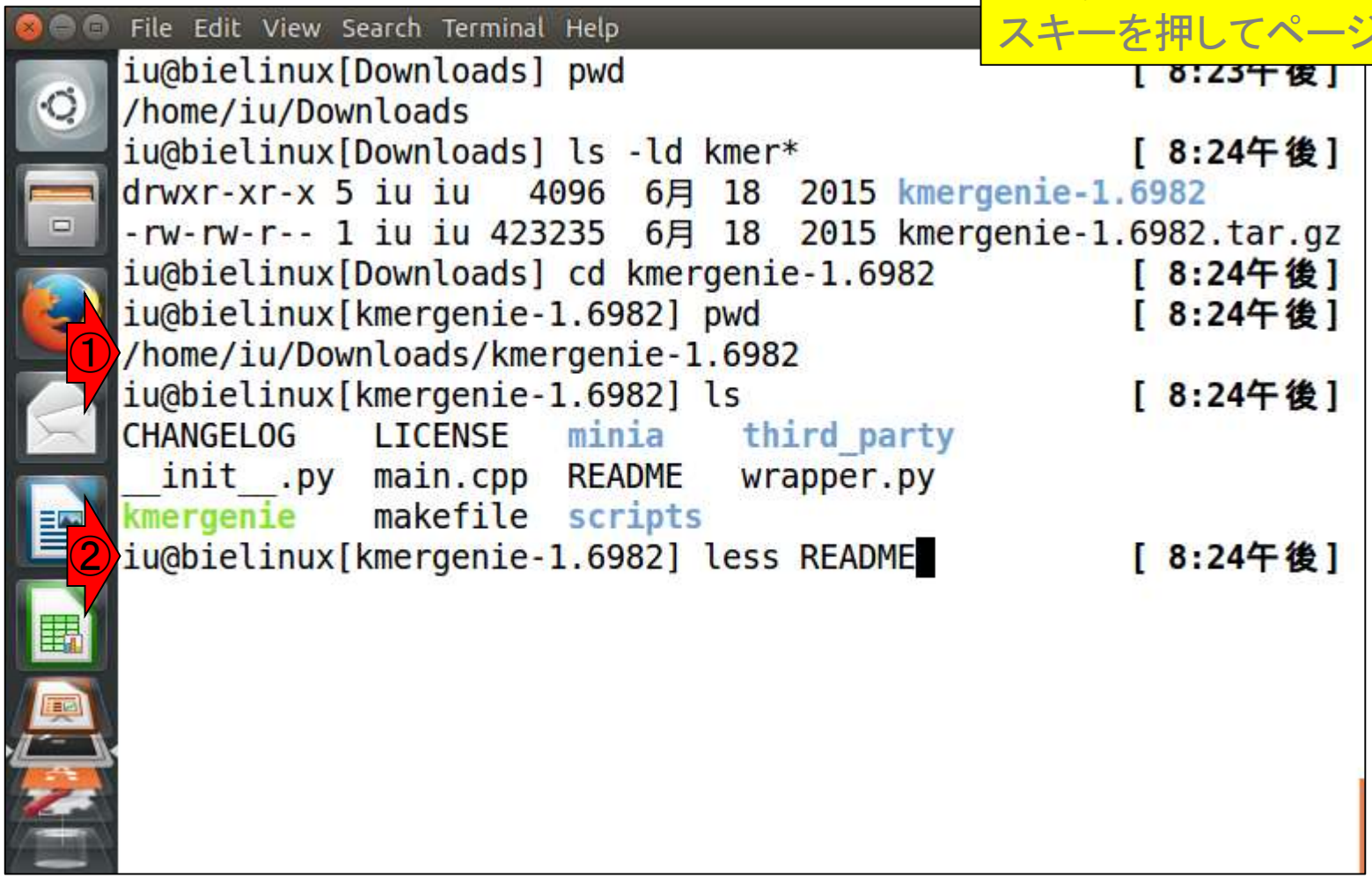
W11-2：ダウンロードと解凍

①解凍が無事終わると、.tar.gzを除いた部分の名前のディレクトリが作成される



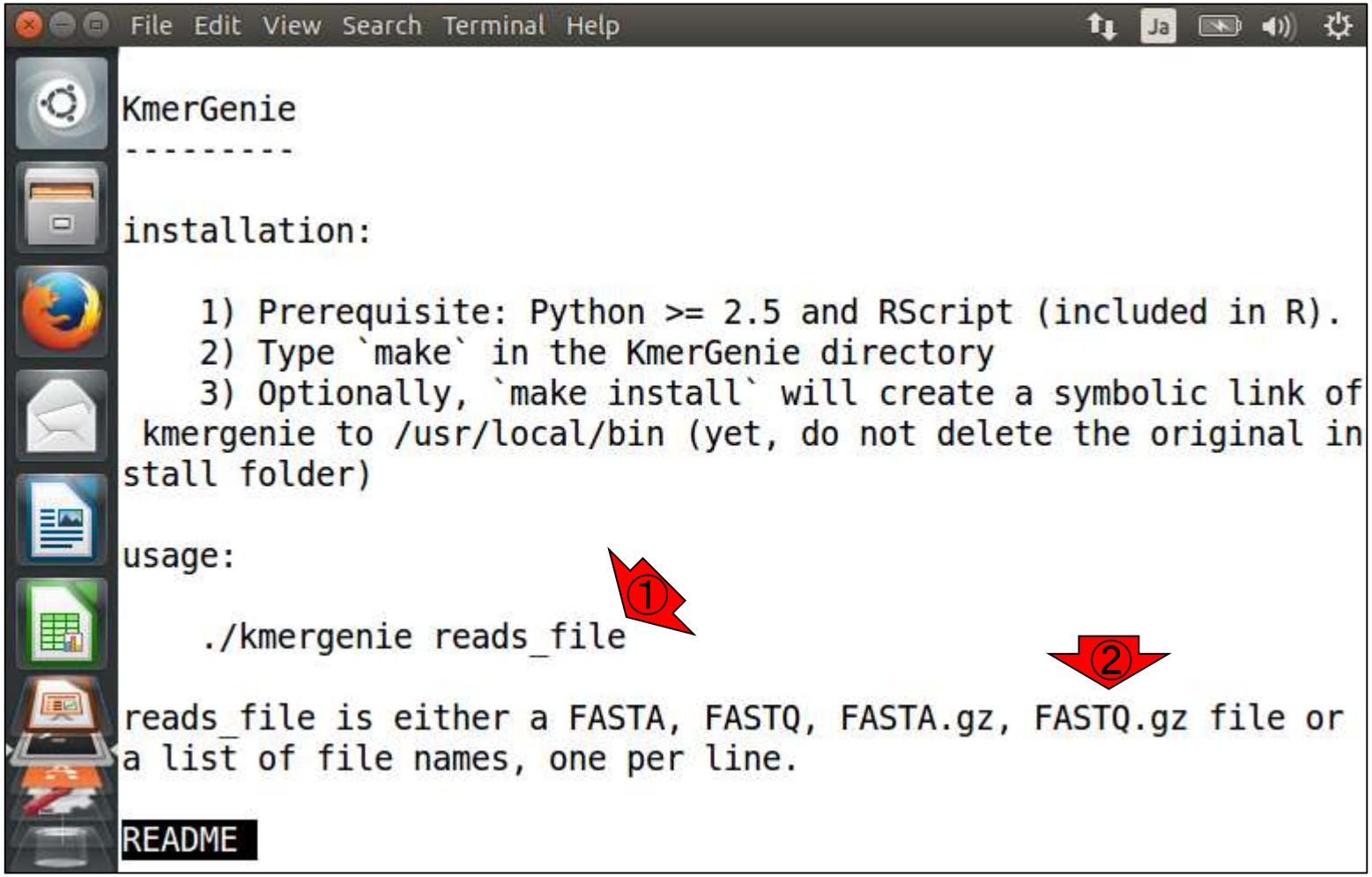
W11-3：README

基本的には、①の場所でmakeと打てばいいが、今一度マニュアル(README)を②lessで眺める。lessの利用法は、第3回W14-6。（スペースキーを押してページスクロール、qで抜ける）



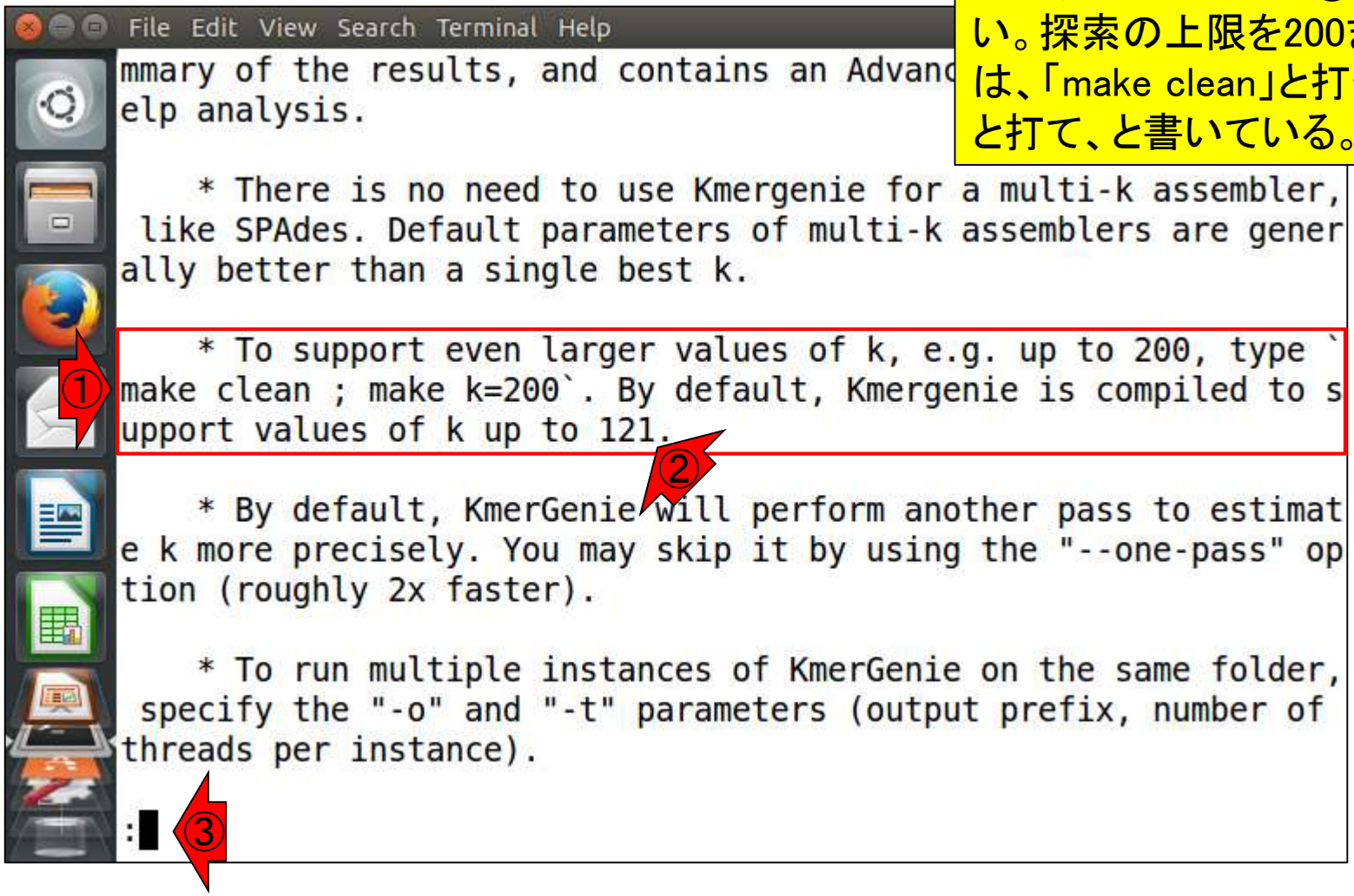
基本的な使用法は①のような感じで、②.fastq.gzも読み込めることを確認

W11-3：README



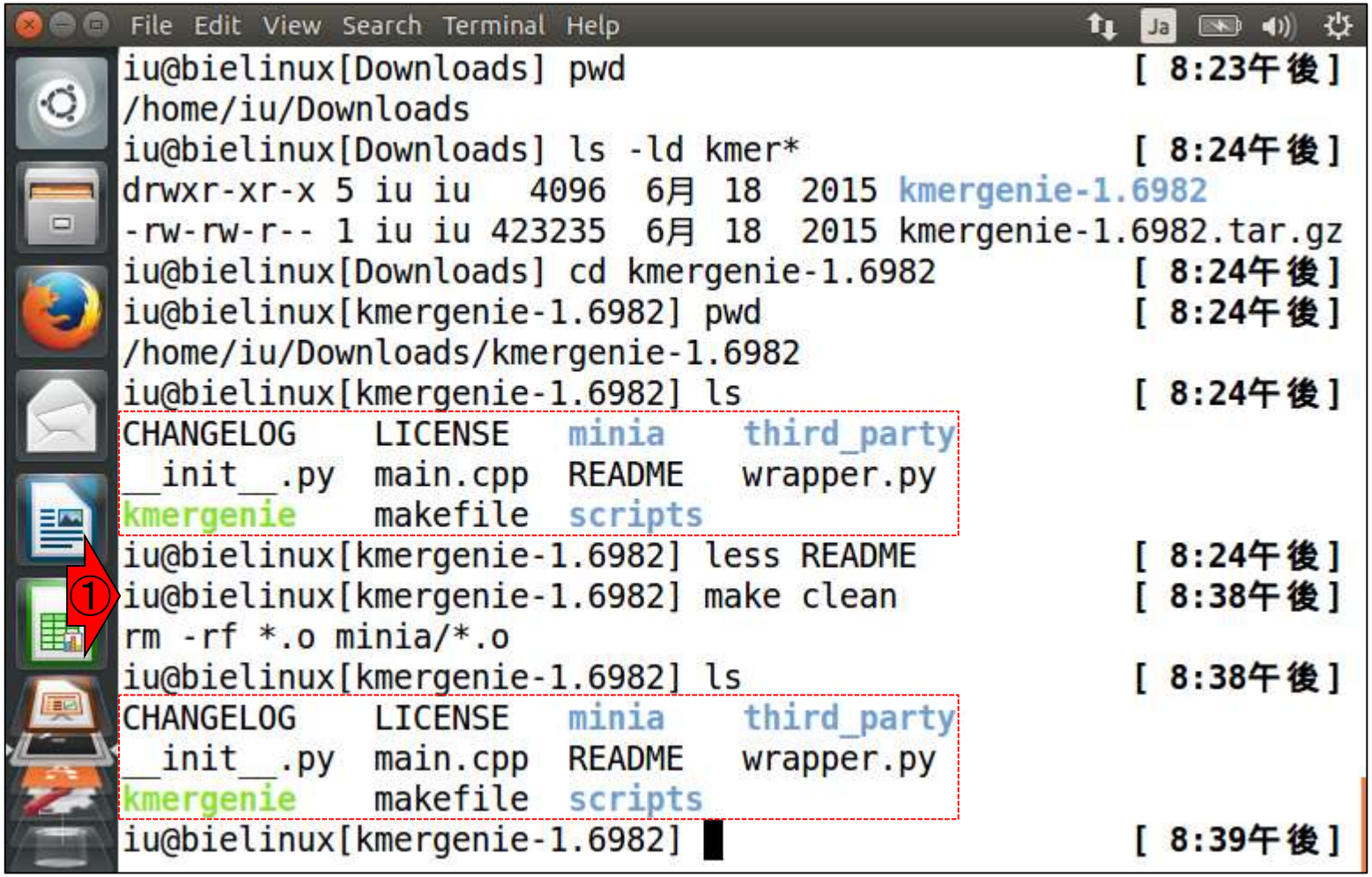
W11-3：README

前のスライドからスペースキーを2回押したあたりの画面。①の赤枠部分に注目。デフォルトでは、KmerGenieは②k=121までしか探索しない。探索の上限を200まで引き上げたい場合は、「make clean」と打ったのち、「make k=200」と打て、と書いている。③qと打ってlessを終了



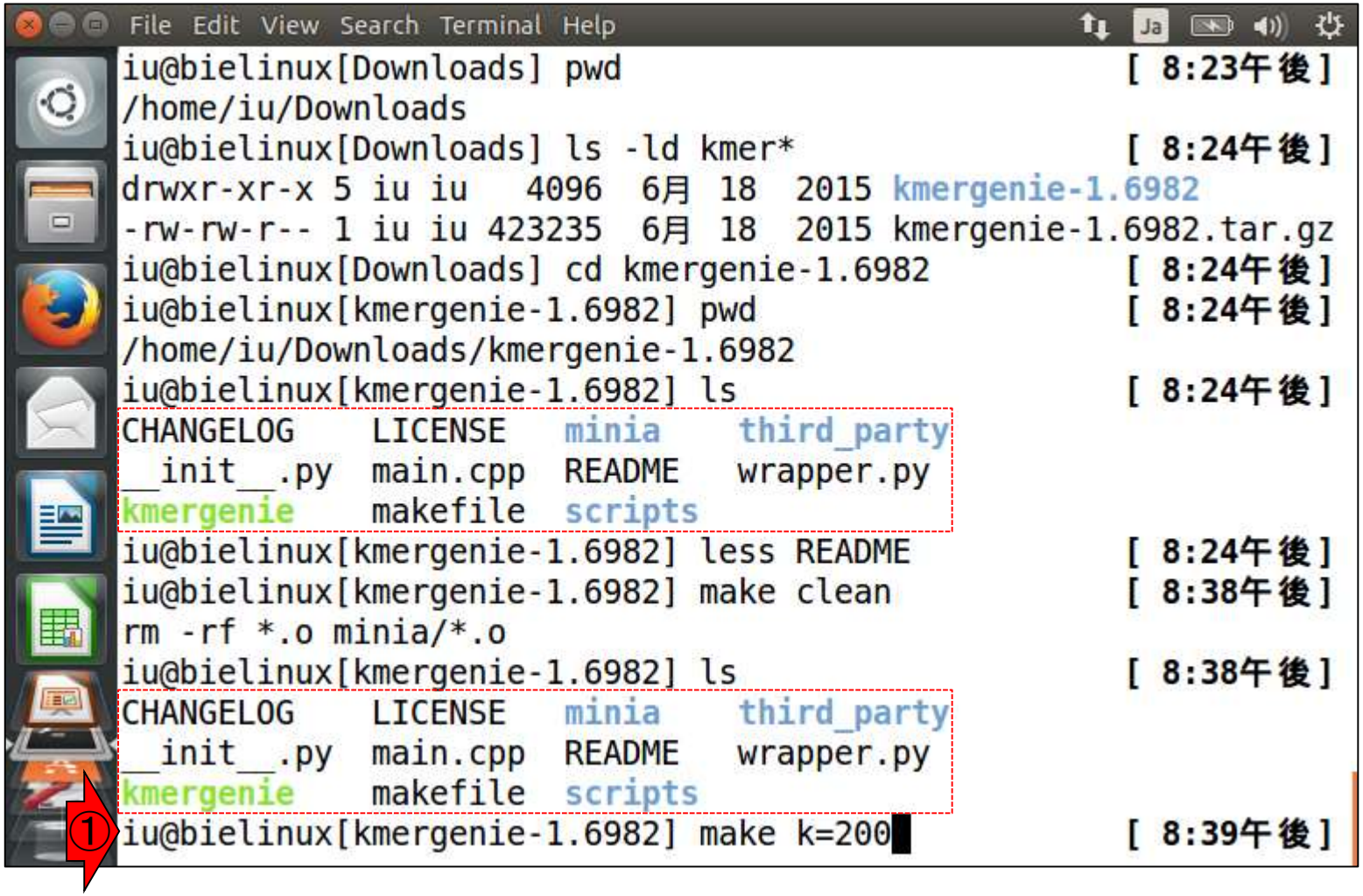
W11-4：make clean

まずは①make clean。実行前後で特に変わった様子はないようだ



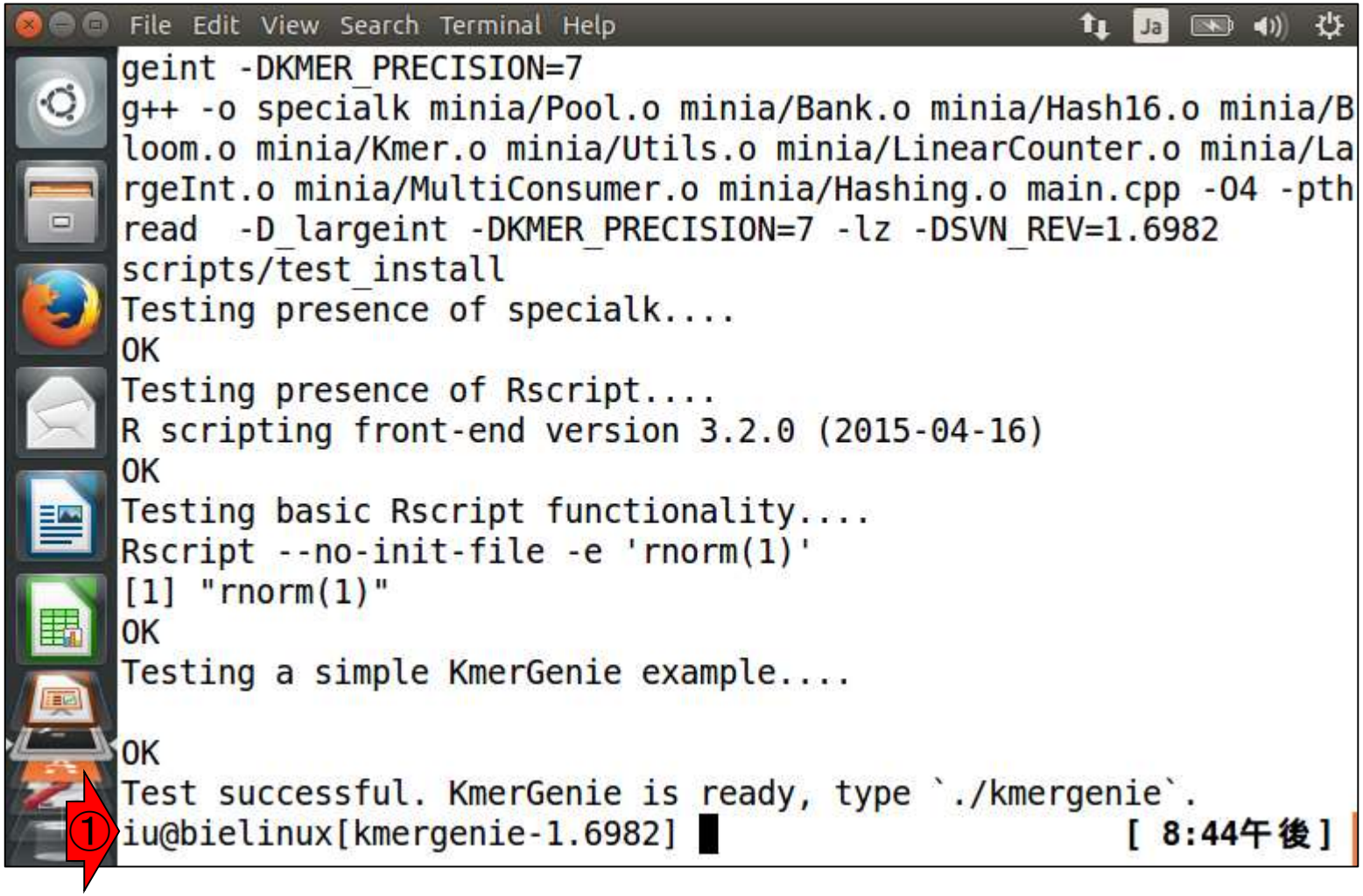
引き続いて①make k=200。約10秒

W11-5：make k=200



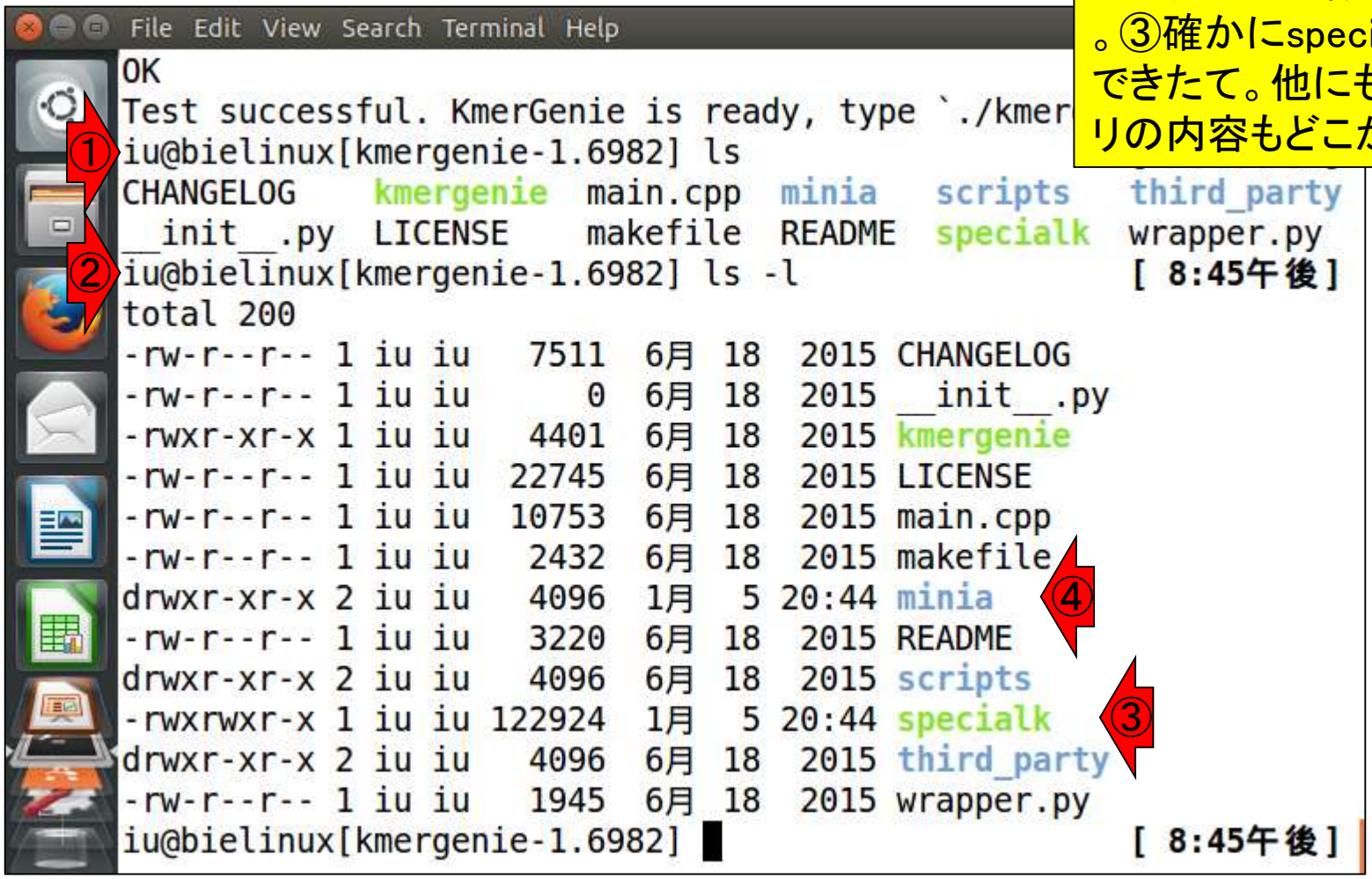
W11-5：make k=200

無事インストールできたようだ。これを「makeが無事に通る」などと呼ぶ



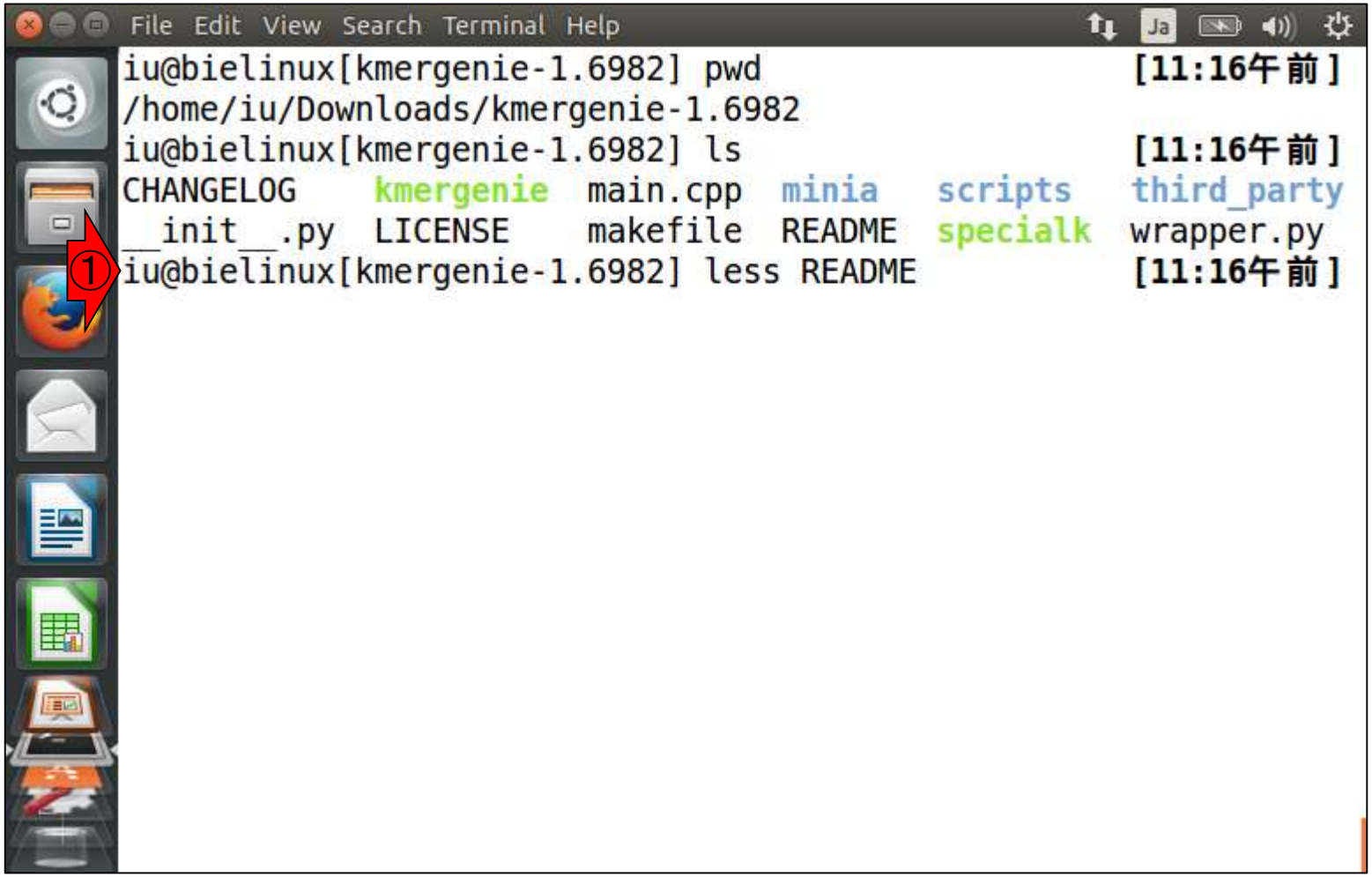
W11-5：make k=200

2016年1月5日に作業している。①11個だったのが12個に増えている。specialkというものが増えたもののようだ。②ls -l。③確かにspecialkという実行ファイルができたて。他にも④miniaというディレクトリの内容もどこかが変更されたようだ



W11-6：README

利用法を見るべく、①もう一度READMEファイルをlessする



W11-6：README

①これが基本的な使い方。赤下線左端の./を見て、「kmergenieのパスを通さなければ、絶対パスで書かないといけないのか…」と認識する。そして、②赤枠内のOptionally以下の文章に目が留まる。実行プログラム(この場合kmergenie)のシンボリックリンクを/usr/local/binに置けばいいことは第4回でも説明した。第4回W9-5で紹介した「ln -s …」の記述法は結構面倒だが、「make install」と打てばいいらしいと解釈

File Edit View Search Terminal Help

```
KmerGenie
---------

installation:

    1) Prerequisite: Python >= 2.5 and RScript (included in R).
    2) Type `make` in the KmerGenie directory
    3) Optionally, `make install` will create a symbolic link of
 kmergenie to /usr/local/bin (yet, do not delete the original in
stall folder)

usage:

    ./kmergenie reads_file

reads_file is either a FASTA, FASTQ, FASTA.gz, FASTQ.gz file or
a list of file names, one per line.

README
```

①

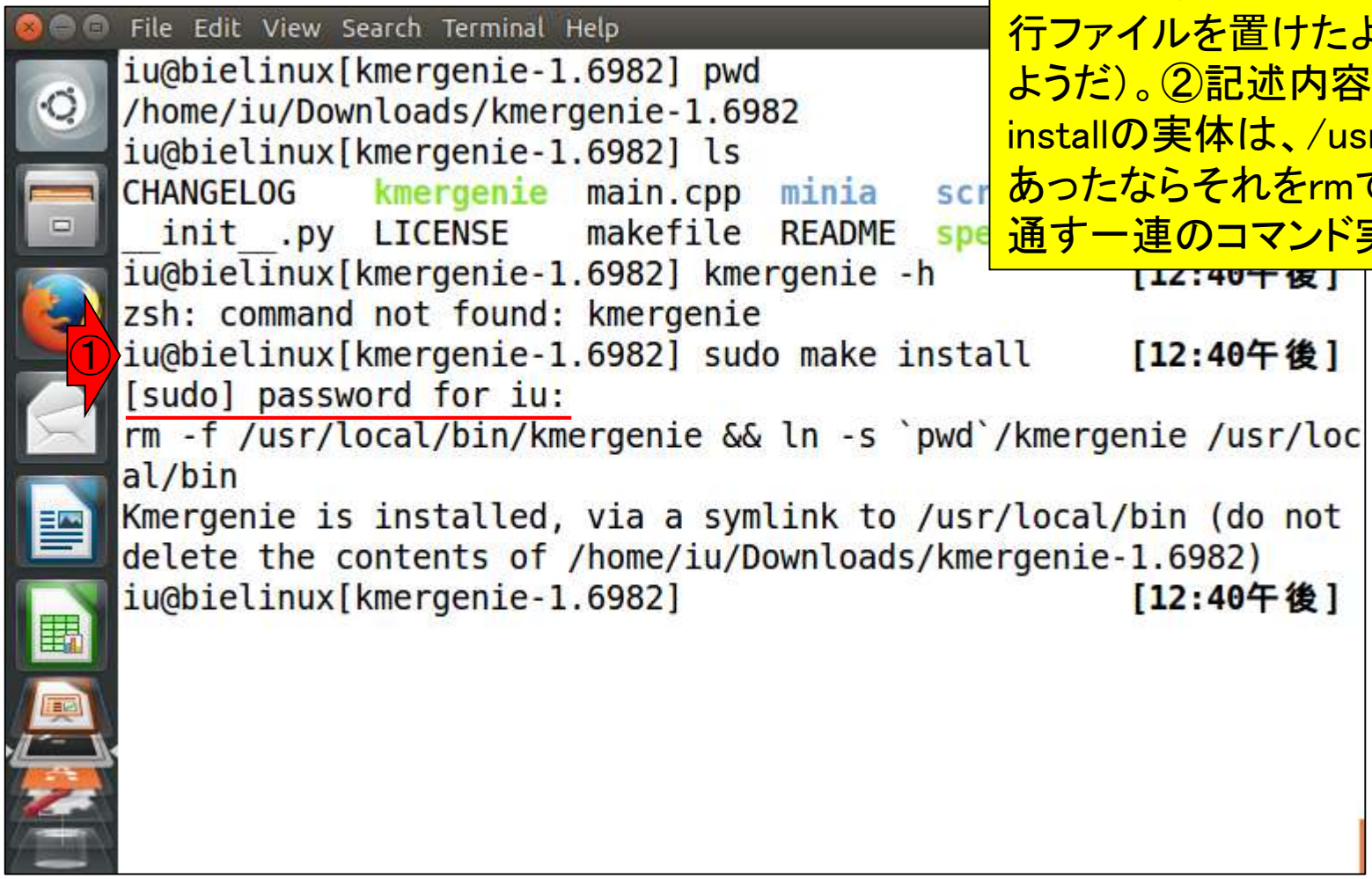
②

W11-7：make install

```
File  Edit  View  Search  Terminal  Help
iu@bielinux[kmergenie-1.6982] pwd
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls
CHANGELOG    kmergenie  main.cpp  minia   scr
__init__.py  LICENSE    makefile  README  spe
iu@bielinux[kmergenie-1.6982] less README
①iu@bielinux[kmergenie-1.6982] make install
rm -f /usr/local/bin/kmergenie && ln -s `pwd`/kmergenie /usr/loc
al/bin
ln: failed to create symbolic link '/usr/local/bin/kmergenie': P
ermission denied
make: *** [install] Error 1
②iu@bielinux[kmergenie-1.6982] ls -ld /usr/local/bin [12:09午後]
drwxr-xr-x 2 root root 4096 12月  11 19:36 /usr/local/bin
③iu@bielinux[kmergenie-1.6982] whoami                [12:09午後]
iu
iu@bielinux[kmergenie-1.6982]                        [12:09午後]
```

(qと打ってlessから抜けて)①make install。エラーが出て失敗していることがわかる。この理由は、第4回W9-5でも説明しているが/usr/local/binディレクトリの所有者はrootさんで、rootさん以外のヒトは書き込み権限が与えられていない(つまりdrwxr-xr-xとなっており、wがない)。そこに③iuさんが書き込みを行おうとしたからPermission denied(権限がない)と文句を言われたというオチです

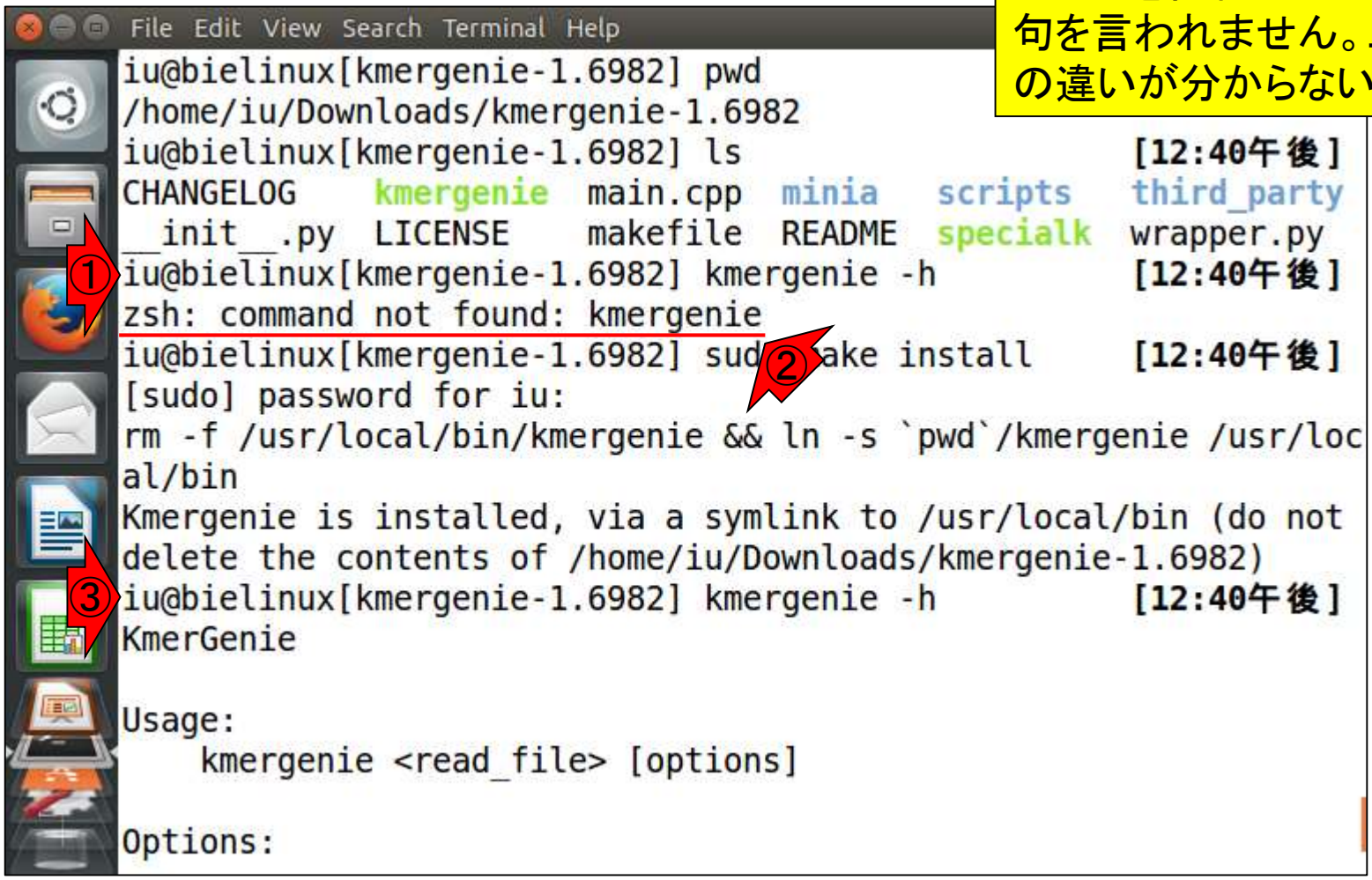
W11-7：make install



①sudo make install。赤下線のところでパスワードを打ち込むよう促されたらpass1409。エラーが出ずに/usr/local/binにkmergenieの実行ファイルを置けたようだ（無事パスを通せたようだ）。②記述内容を眺めることでmake installの実体は、/usr/local/bin/kmergenieがあったならそれをrmで消して、ln –sでパスを通す一連のコマンド実行のようだと学習する

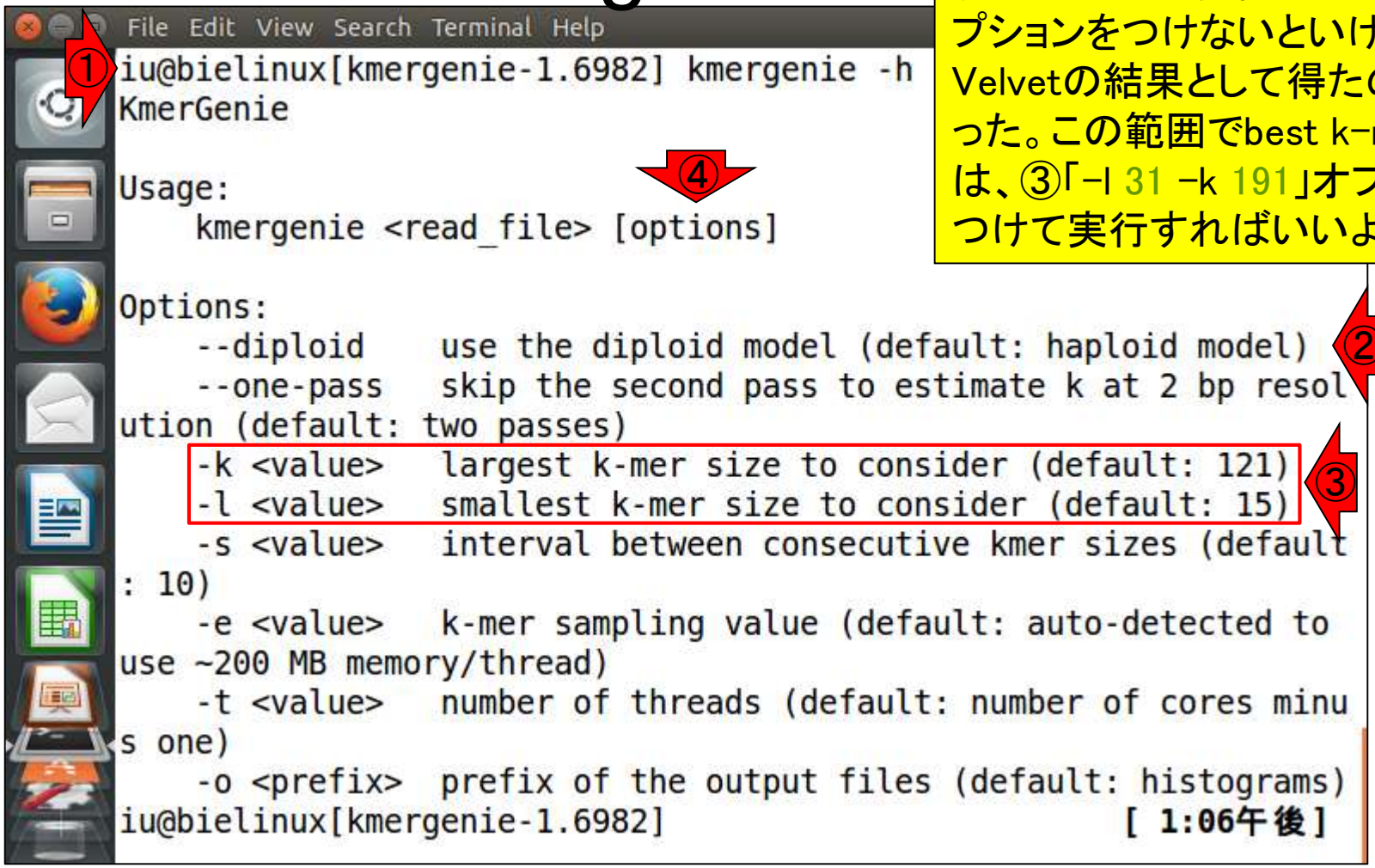
W11-7：make install

①パスを通す前は、kmergenieのヘルプを表示させようとしても②「そんなコマンドはない」と文句を言われるが、③パスを通した後は文句を言われません。./kmergenieとkmergenieの違いが分からないヒトは、第4回W9で復習



W11-7 : kmergenie -h

①kmergenie -hの結果を眺める。②乳酸菌(haploid; 一倍体)の場合は気にしなくていいが、ヒトなどの二倍体ゲノムの場合は--diploidオプションをつけないといけないようだ。W10-6でVelvetの結果として得たのはk=31から191であった。この範囲でbest k-merを探索したい場合は、③「-l 31 -k 191」オプションを、④の位置につけて実行すればいいようだ

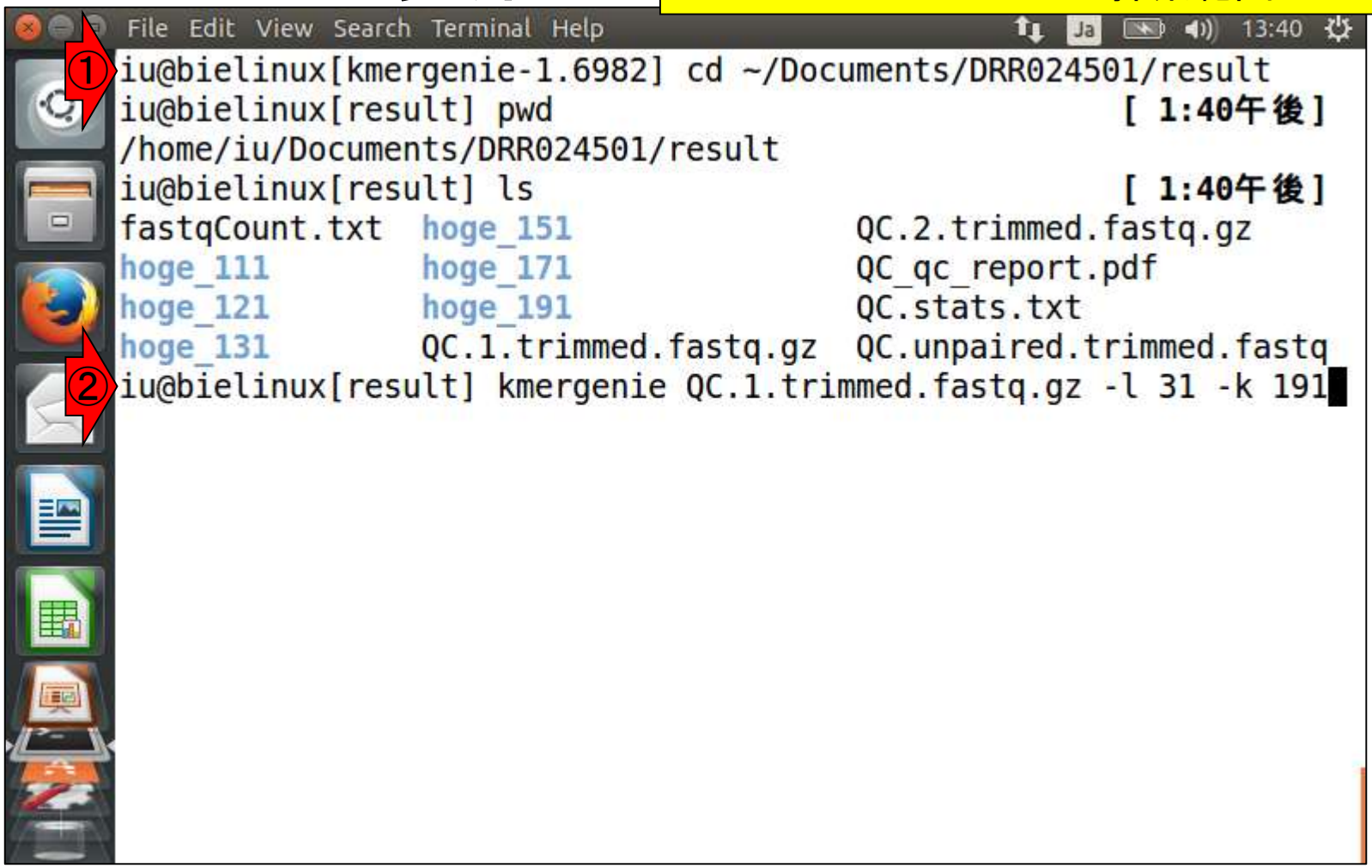


Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

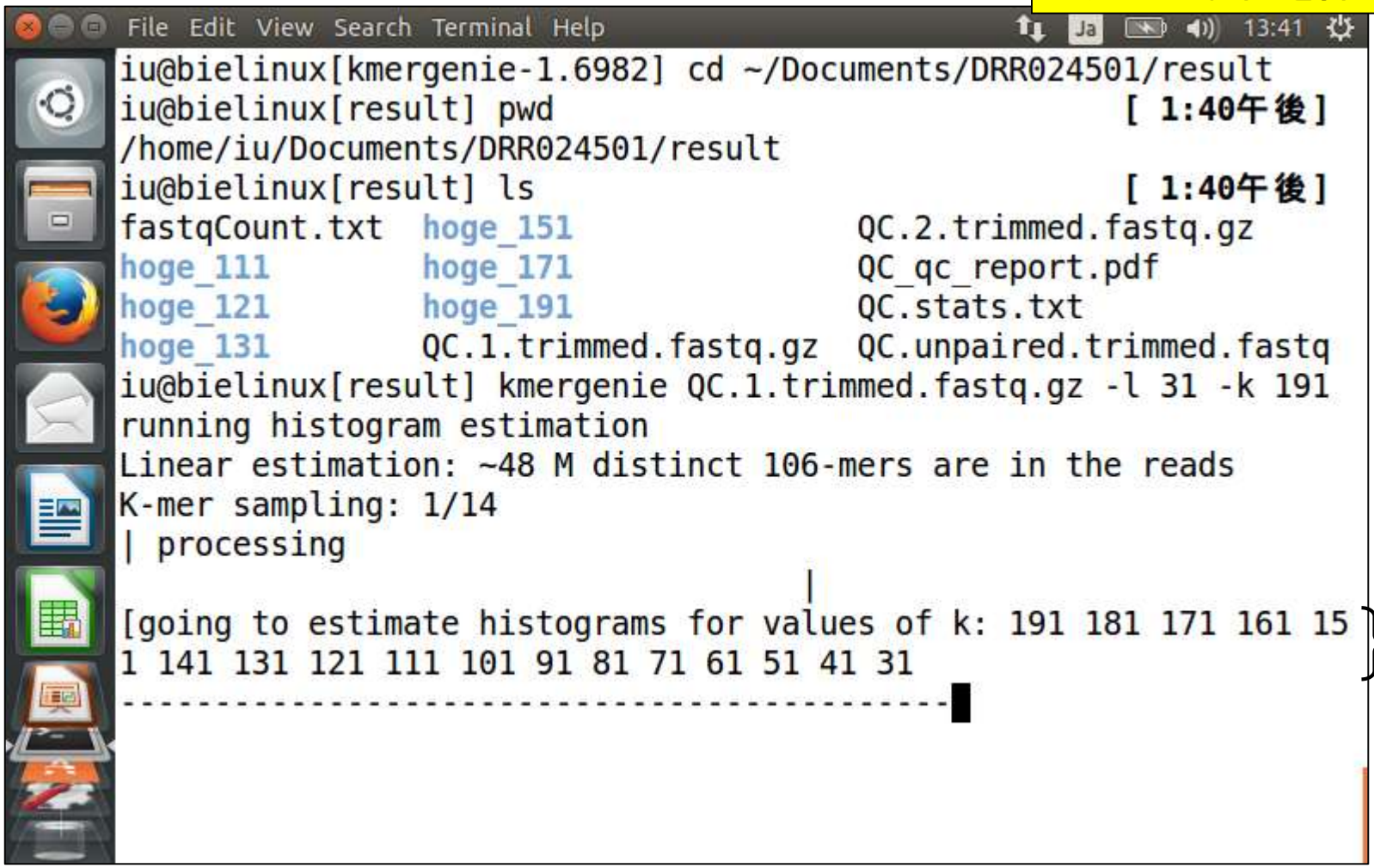
W11-8：実行

一通りの実行方法がわかったので、とりあえず①W5-4で作成したforward側のファイル(QC.1.trimmed.fastq.gz)のみを入力として31から191のk-merの探索範囲でkmergenieを実行。約15分



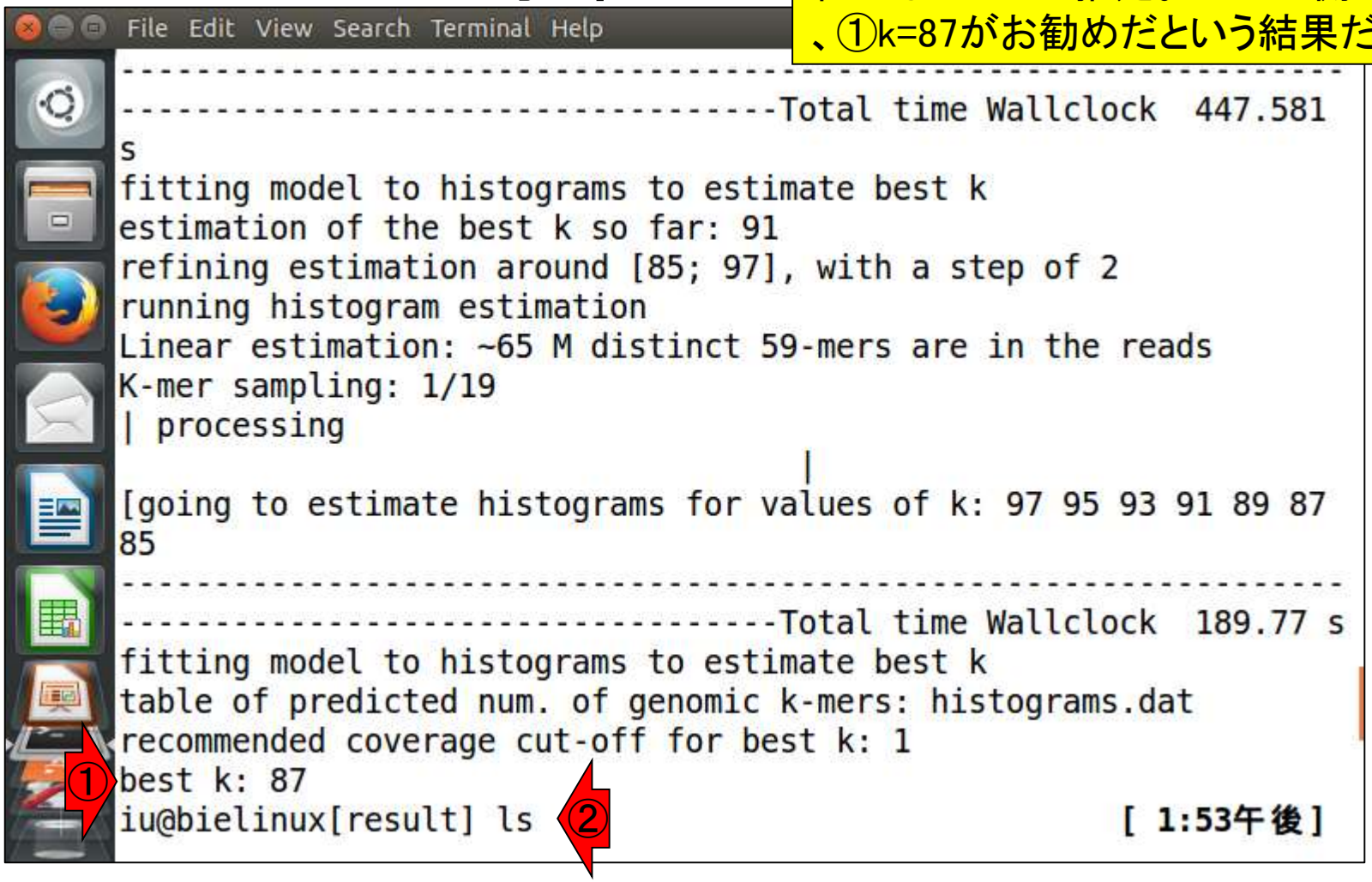
W11-8：途中経過1

リターンキーを押してから、確か10秒後くらいの状態。①確かにオプションで指定した通りの範囲を探索してくれているようだ



W11-8：無事終了

終了後の状態。KmerGenieプログラムの主な目的は、Velvetなどのアセンブリプログラム実行時に最適だと思われるk-merの推定。forward側のみで実行した結果は、①k=87がお勧めだという結果だった。とりあえず②ls

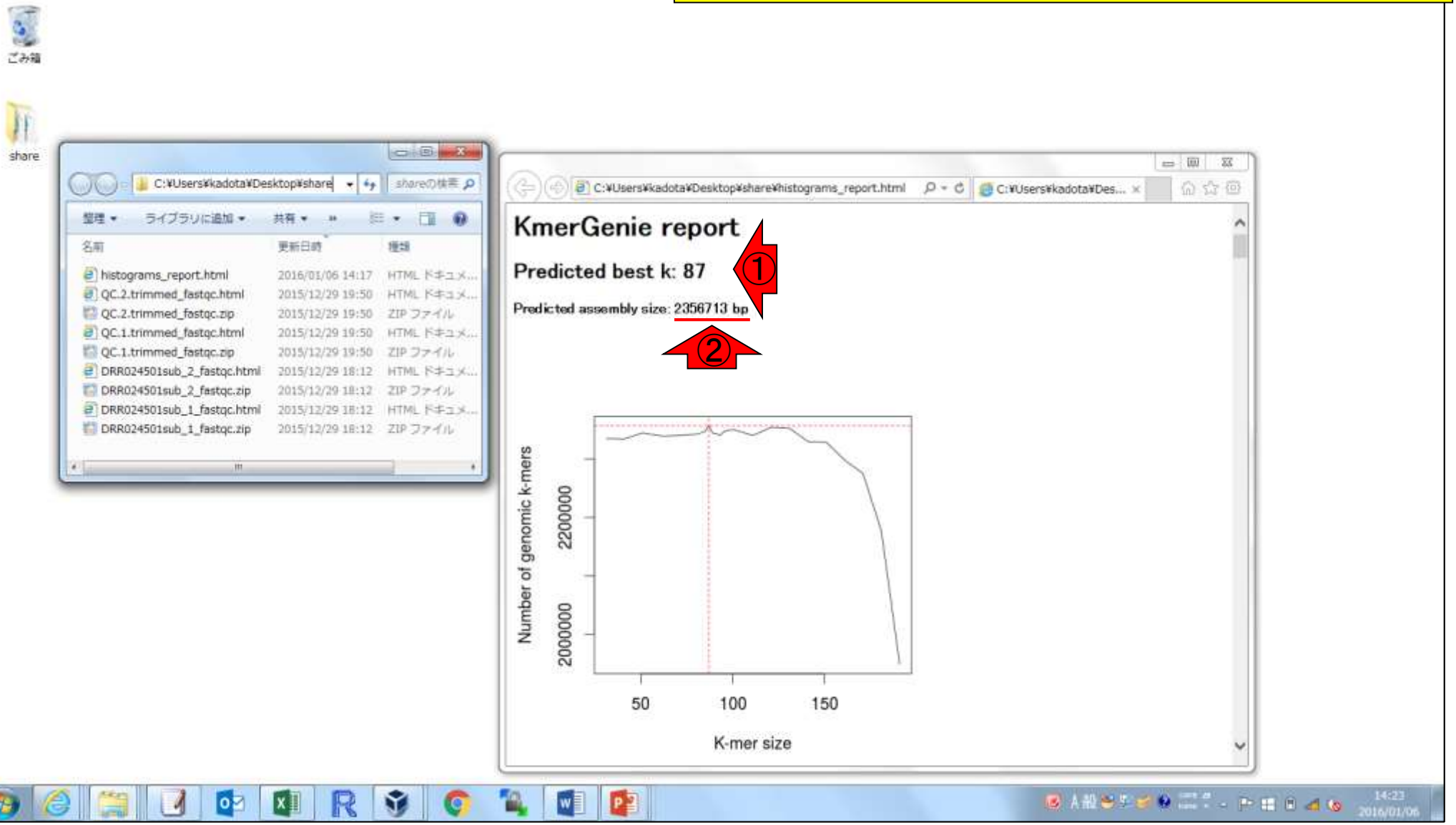


W11-9：確認

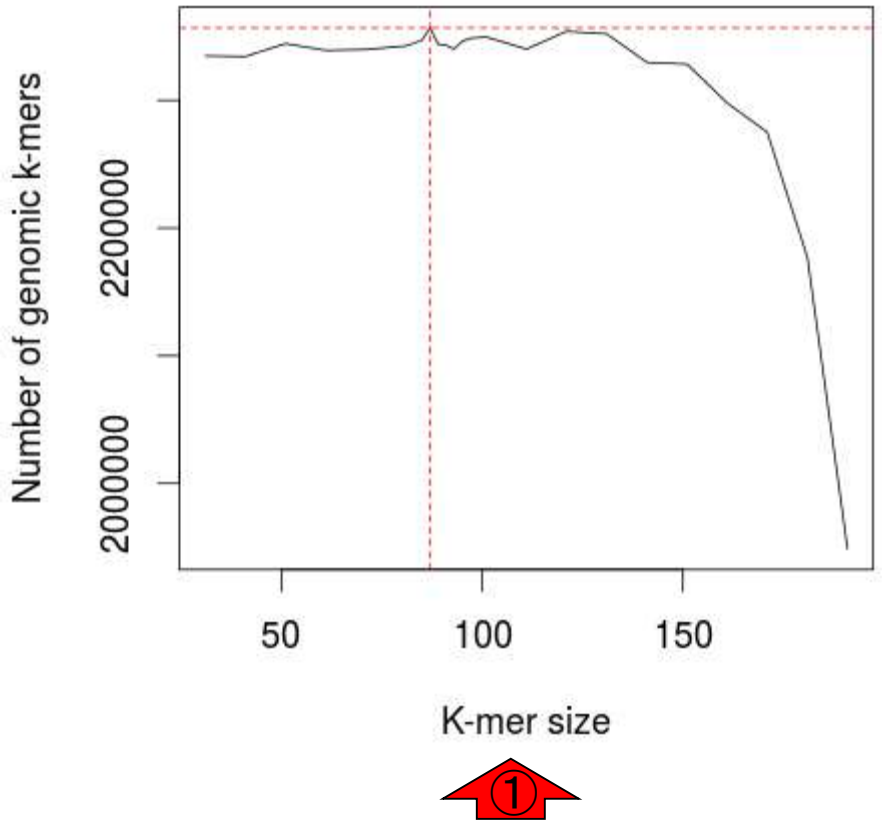
画面が一気に流れる。KmerGenieは沢山のファイルをカレントディレクトリ上に吐き出すようだ。ファイル群histograms-k*.histo*は、個別のk値の結果を見たいときに必要となるのだろう。①このhtmlファイルさえ見れば基本的にOK。ゲストOS(Bio-Linux)上で「firefox histograms_report.html」とやってもよいが、画面が小さく見づらい。ここでは、②共有フォルダ経由でホストOSのshareフォルダにコピーして、ホストOS上でhtmlファイルを眺める

W11-9：確認

ホストOS上でhtmlファイルを眺めた画面。①アセンブリ時に用いるベストなk値は87だという結果。②が推定ゲノムサイズ。2,356,713 bp (約2.4MB)という結果

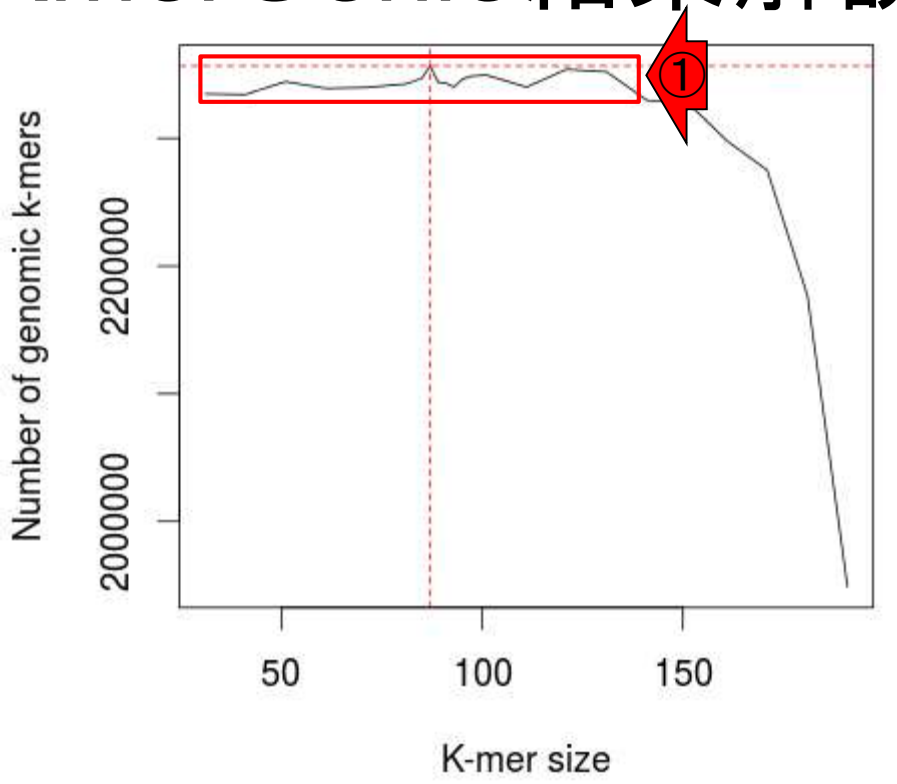


KmerGenie結果解説



もう少し詳しく説明。①横軸が調べたk-merのk値。指定した探索範囲は、31から191なので妥当。縦軸の名称はNumber of *genomic* k-mers。*genomic*という形容詞がついていることからも想像できるが、これが第1部初日(2016.07.20)の最後のほうで示した「シークエンスエラー由来のものを除いたk-merの種類数」です

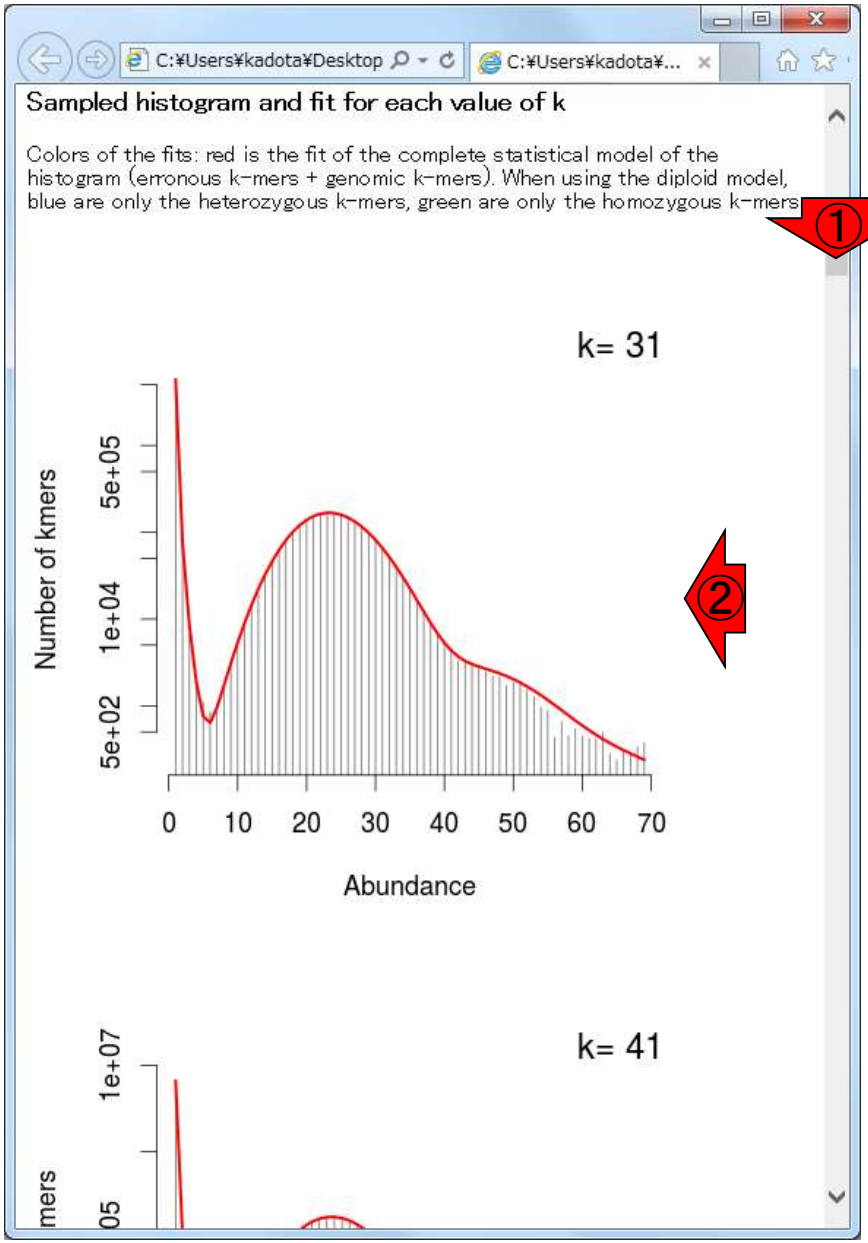
KmerGenie結果解説



①のあたりを眺めることで、k=30-130くらいの範囲でどのk-merを用いてもゲノムサイズが2.3-2.4MBという結果になるのだろうと解釈する。但し、これは「*genomic* k-mersの種類数」なのでシークエンスエラー由来k-merのフィルタリングが実装されていない（甘い）アセンブリプログラムだとそうはいかないだろう、とも想像する

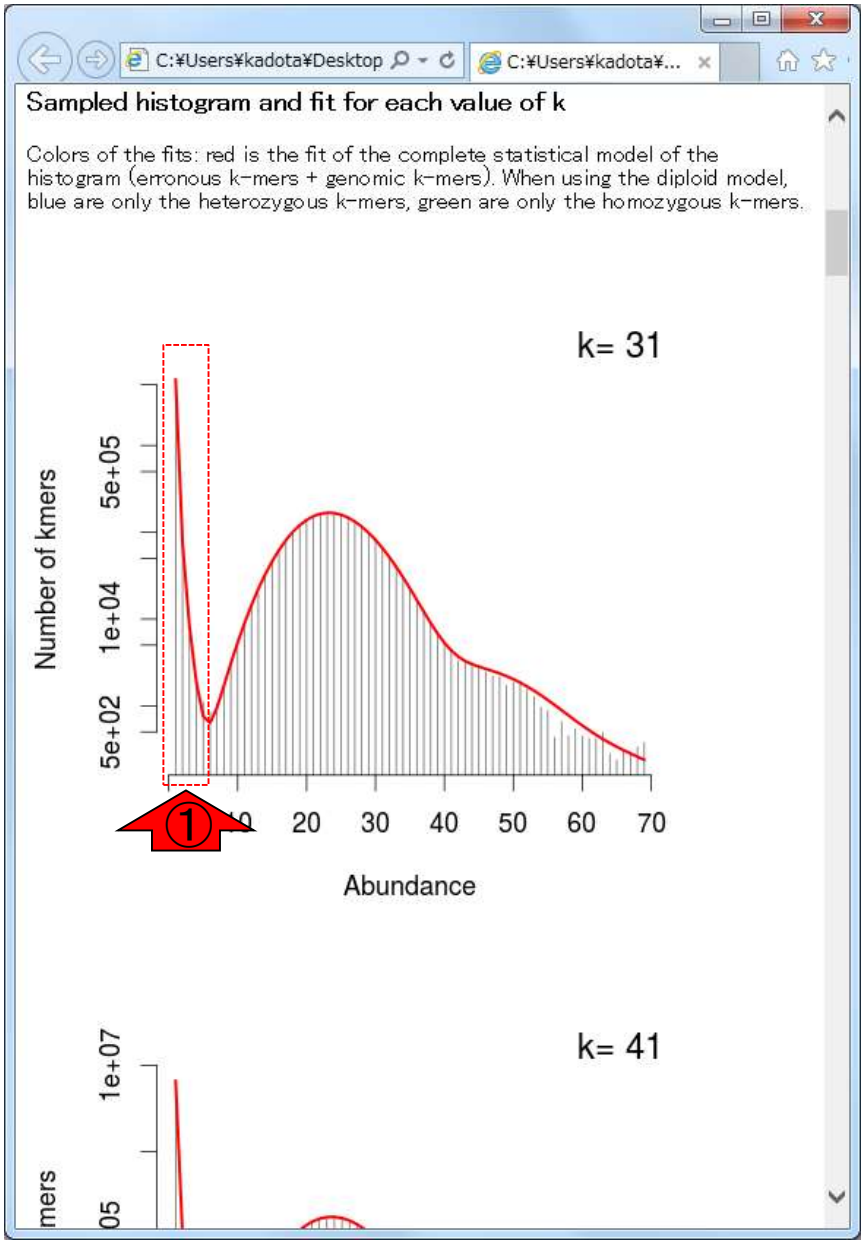
KmerGenie結果解説

htmlファイル自体は結構縦長である。①のあたりから、オプションで指定した31から191のk値の個別の解析結果が見られる。例えば②がk=31のときのk-mer出現頻度分布

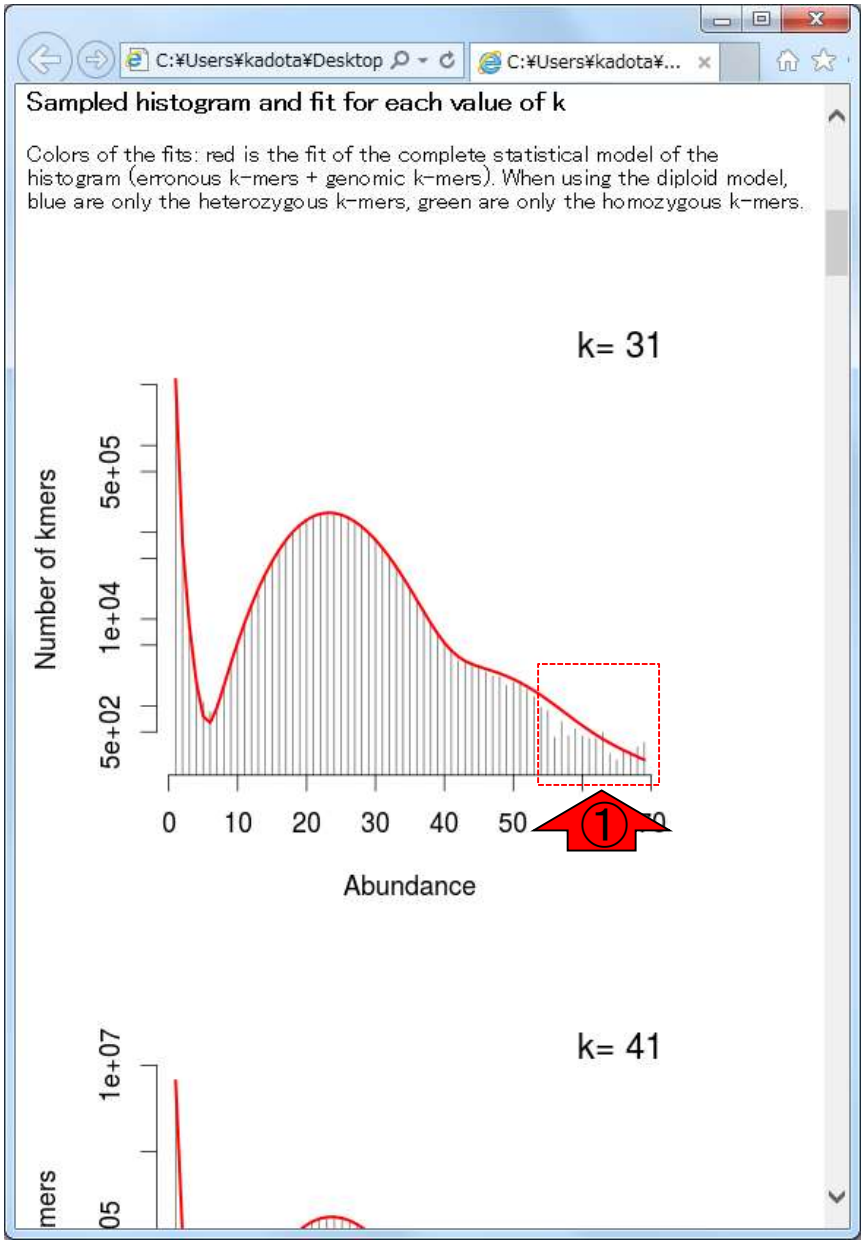


KmerGenie結果解説

KmerGenieは、①のシークエンスエラー由来k-merを除いた残りのk-merの種類数を*genomic*と判定しているのだろう



KmerGenie結果解説



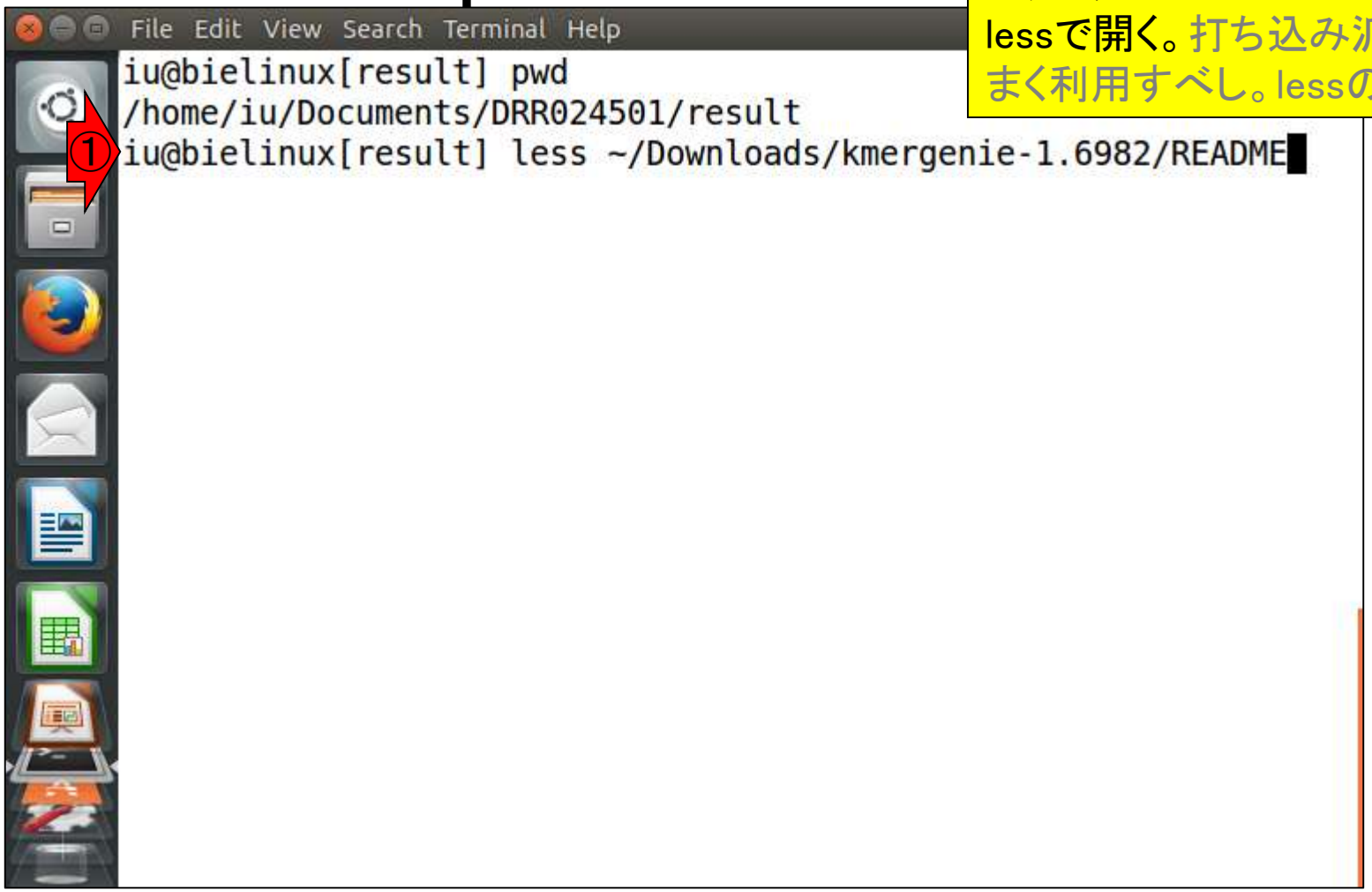
ほぼ余談。実際には、①の付近のやたら沢山出現するk-merもフィルタリングしている(はず)。このあたりの多くは、リピート配列由来k-merの領域だからです

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

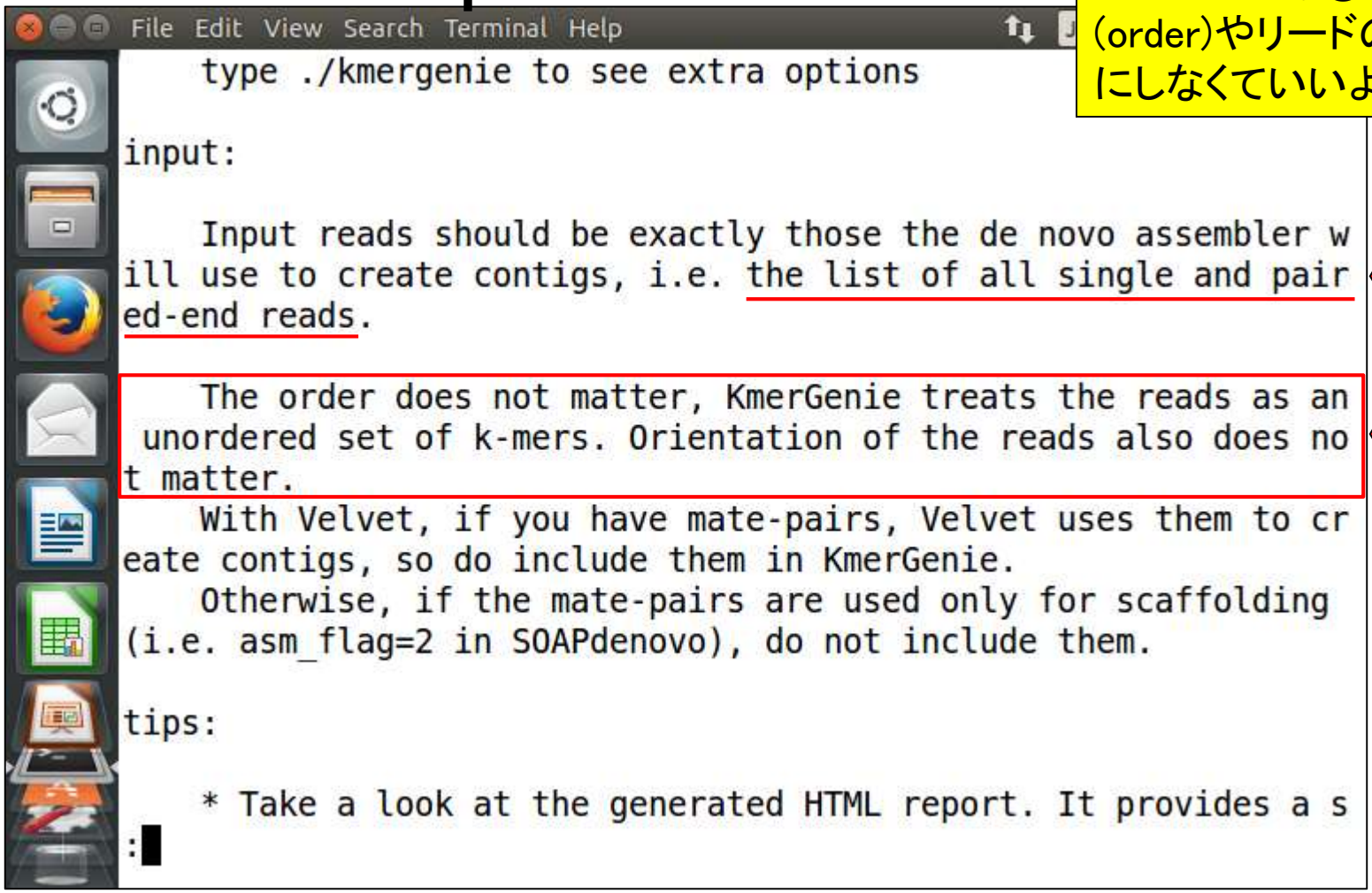
W11-10：paired-end

paired-endの場合に、どう指定すればいいのか調べる。ここでは①入力ファイルがあるディレクトリ上でKmerGenieのREADMEファイルをlessで開く。打ち込み派のヒトは、タブ補完をうまく利用すべし。lessの利用法は、第3回W14-6



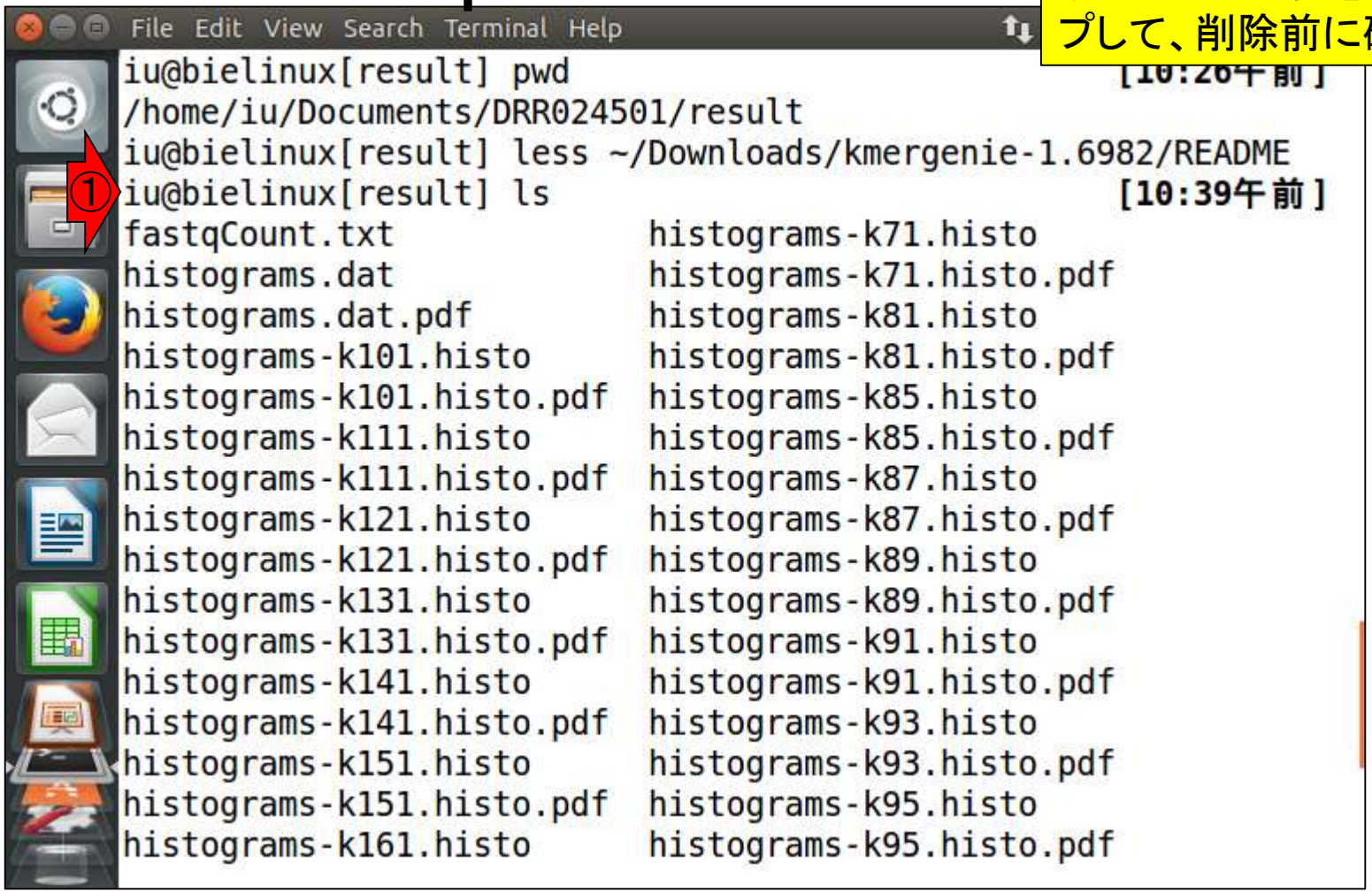
W11-10：paired-end

①複数のファイルに分かれている場合はリストファイルを作成して入力として与えればよい。②入力ファイルの順番(order)やリードの向き(orientation)は気にしなくていいようだ。qでlessから抜ける



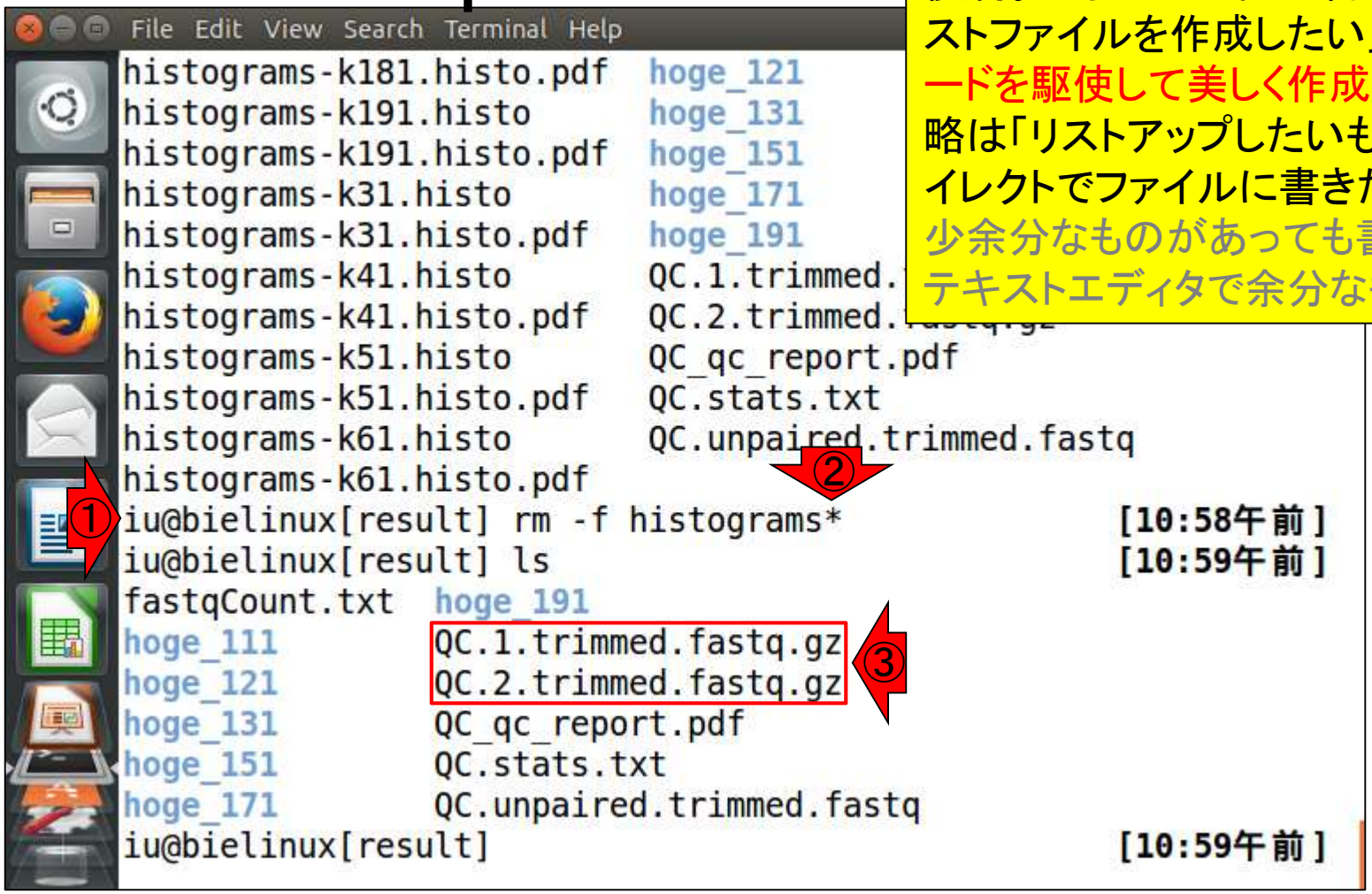
W11-10：paired-end

①lsするとhistograms*のファイルが沢山出て見づらいので削除する。実用上は削除候補ファイル群をlsやタブ補完でリストアップして、削除前に確認する(第4回のW1-2)



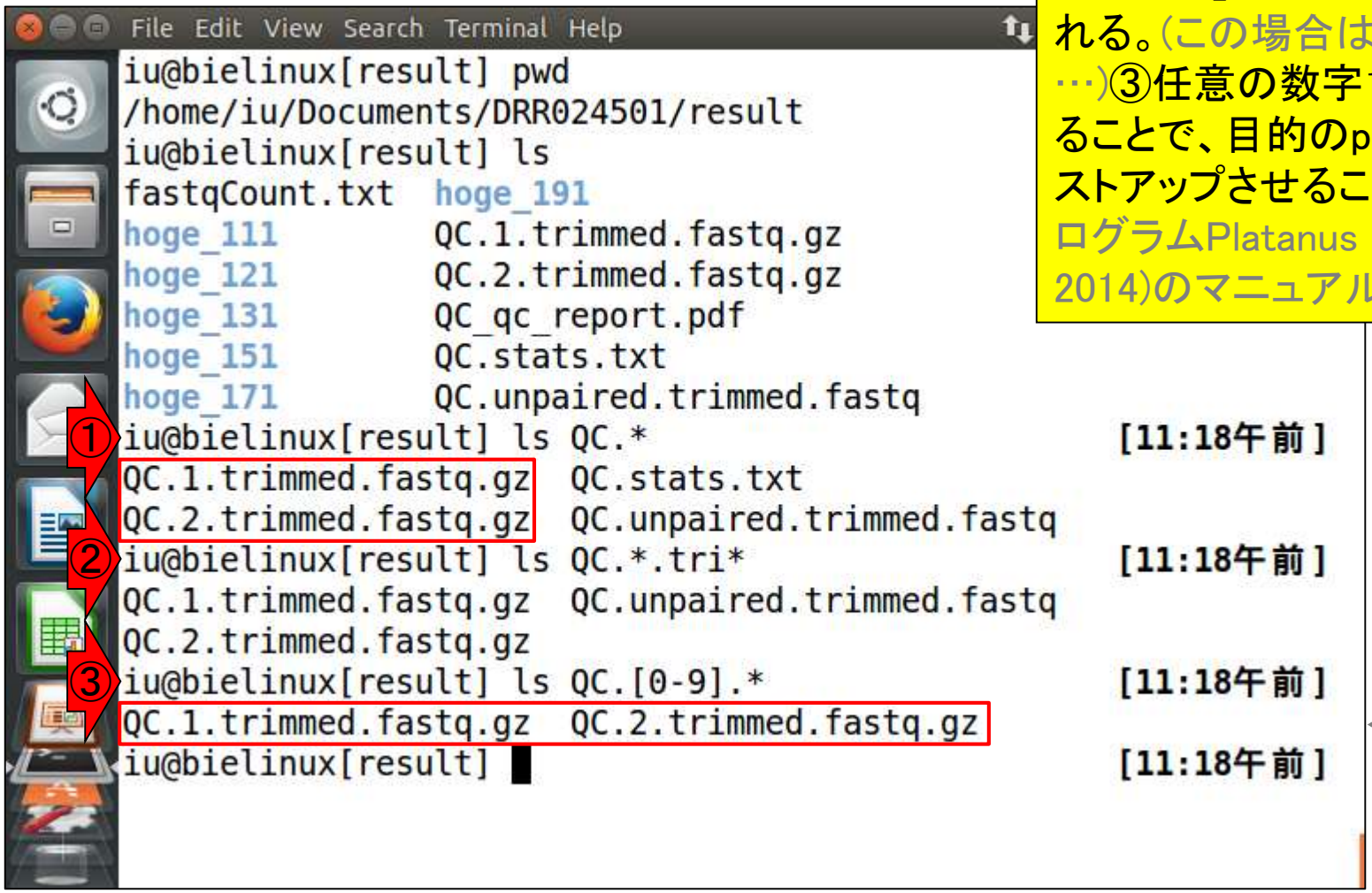
W11-10：paired-end

①rm -fで削除。②*はワイルドカードの1つ(第4回のW1-2)。ワイルドカードは使いこなせるとかなり便利。ちなみに当面の目的は③「paired-endのリストファイルを作成したい」。ここでは、ワイルドカードを駆使して美しく作成するTipsを紹介。基本戦略は「リストアップしたいもののみlsし、それをリダイレクトでファイルに書きだす」です。実際には多少余分なものがあっても書きだした上で、viなどのテキストエディタで余分なものを削除したりします



W11-10：ワイルドカード

①「QC.*」だと、赤枠で示す目的のpaired-endファイル以外のものが表示される。②「QC.*.tri*」でも目的外のファイルが表示される。(この場合はQC.*.gzでもうまくいくが…)③任意の数字1字を表す[0-9]を利用することで、目的のpaired-endファイルのみリストアップさせることができる。アセンブリプログラムPlatanus (ver. 1.2.4; Kajitani et al., 2014)のマニュアルを見るときに役立つ

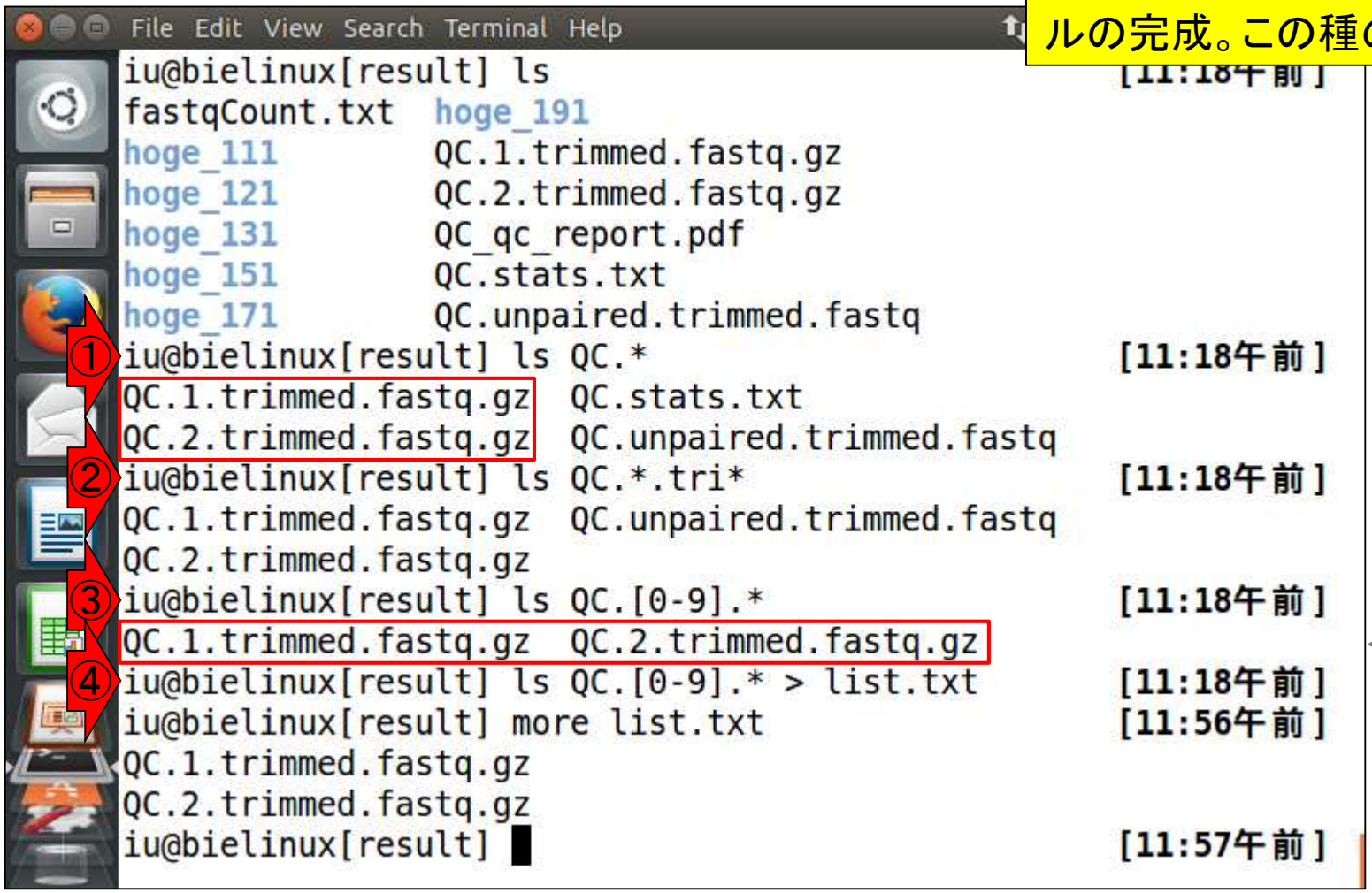


Kajitani et al., *Genome Res.*, **24**: 1384-1395, 2014

Chikhi and Medvedev, *Bioinformatics*, **30**: 31-37, 2014

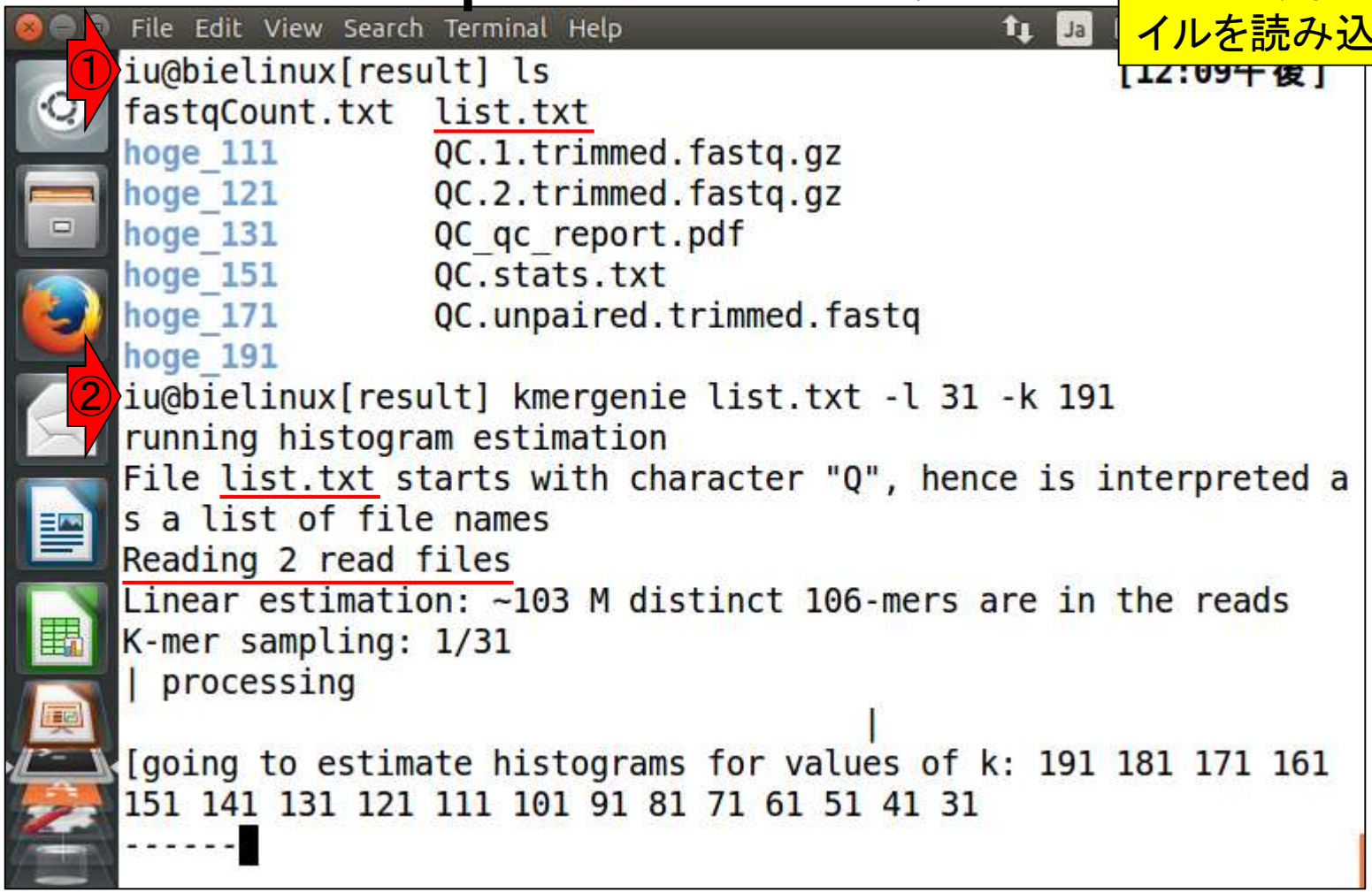
W11-10：ワイルドカード

③でリストアップできることを確認したら、④リダイレクト(>)を利用して任意のファイル名(ここではlist.txt)で保存すれば、リストファイルの完成。この種のテクニックは頻用する



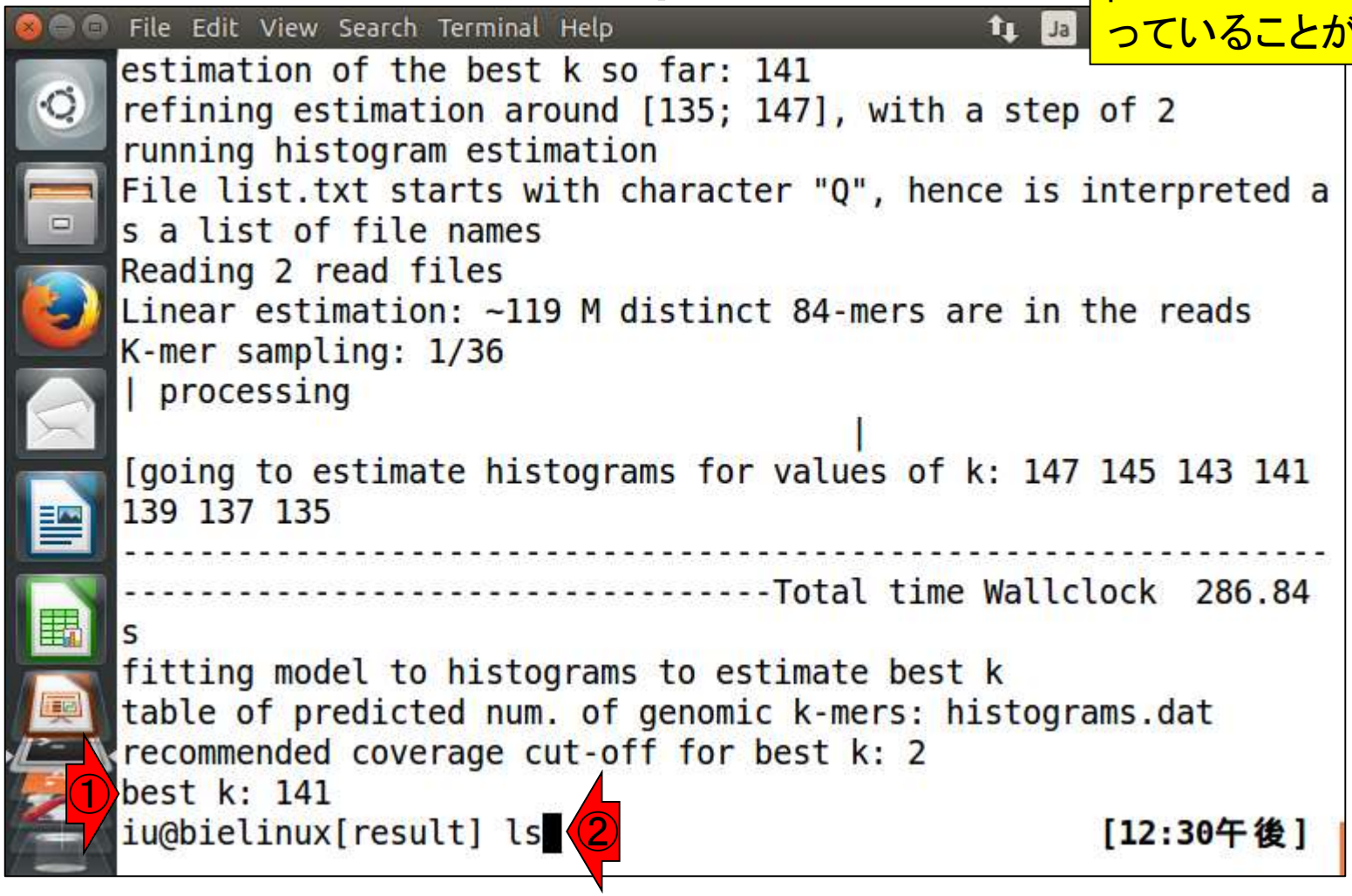
W11-11：paired-end実行

①リストファイルlist.txtがあることを確認して、②kmergenieコマンドを実行。画面は数秒後の状態。うまくリストファイルを読み込めているようだ。約20分



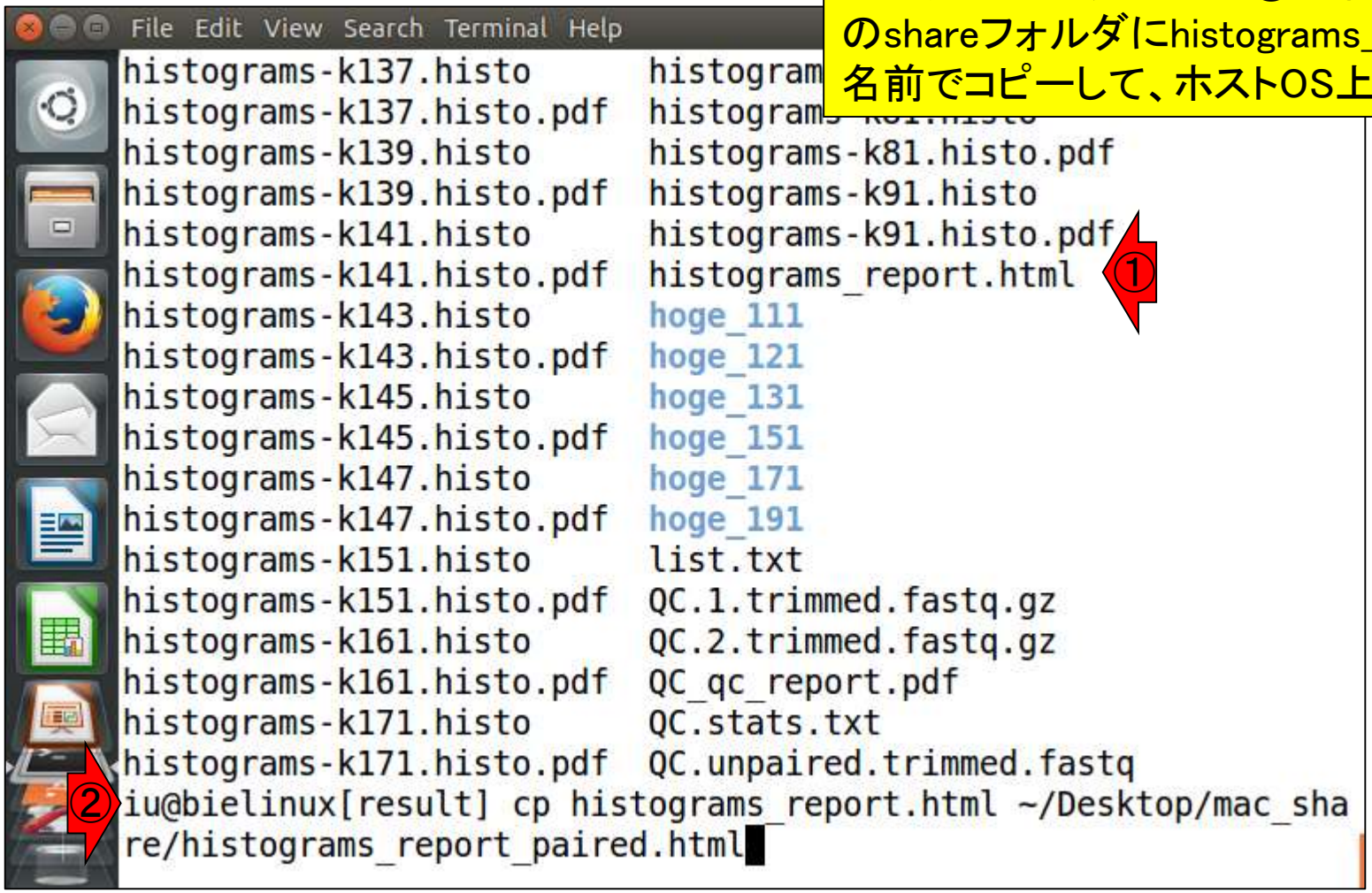
W11-11：無事終了

終了後の状態。forward側のみ(single-end)で実行した結果(k=87)と違って、①paired-endのときはk=141がお勧めとなっていることがわかる。とりあえず②ls



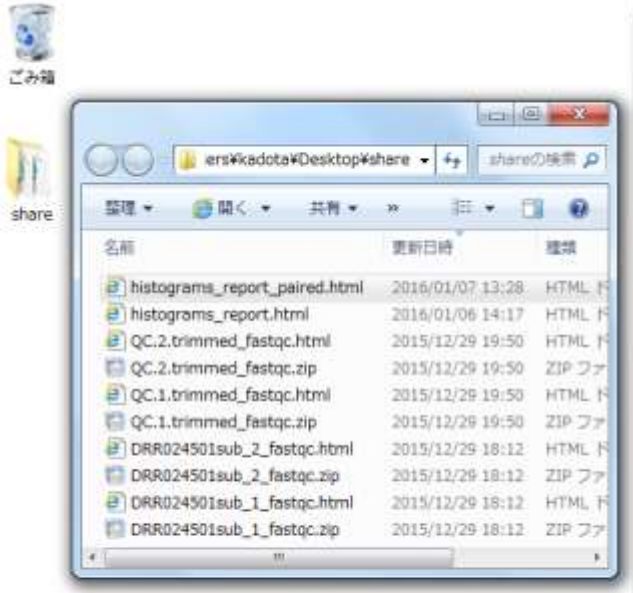
W11-11：確認

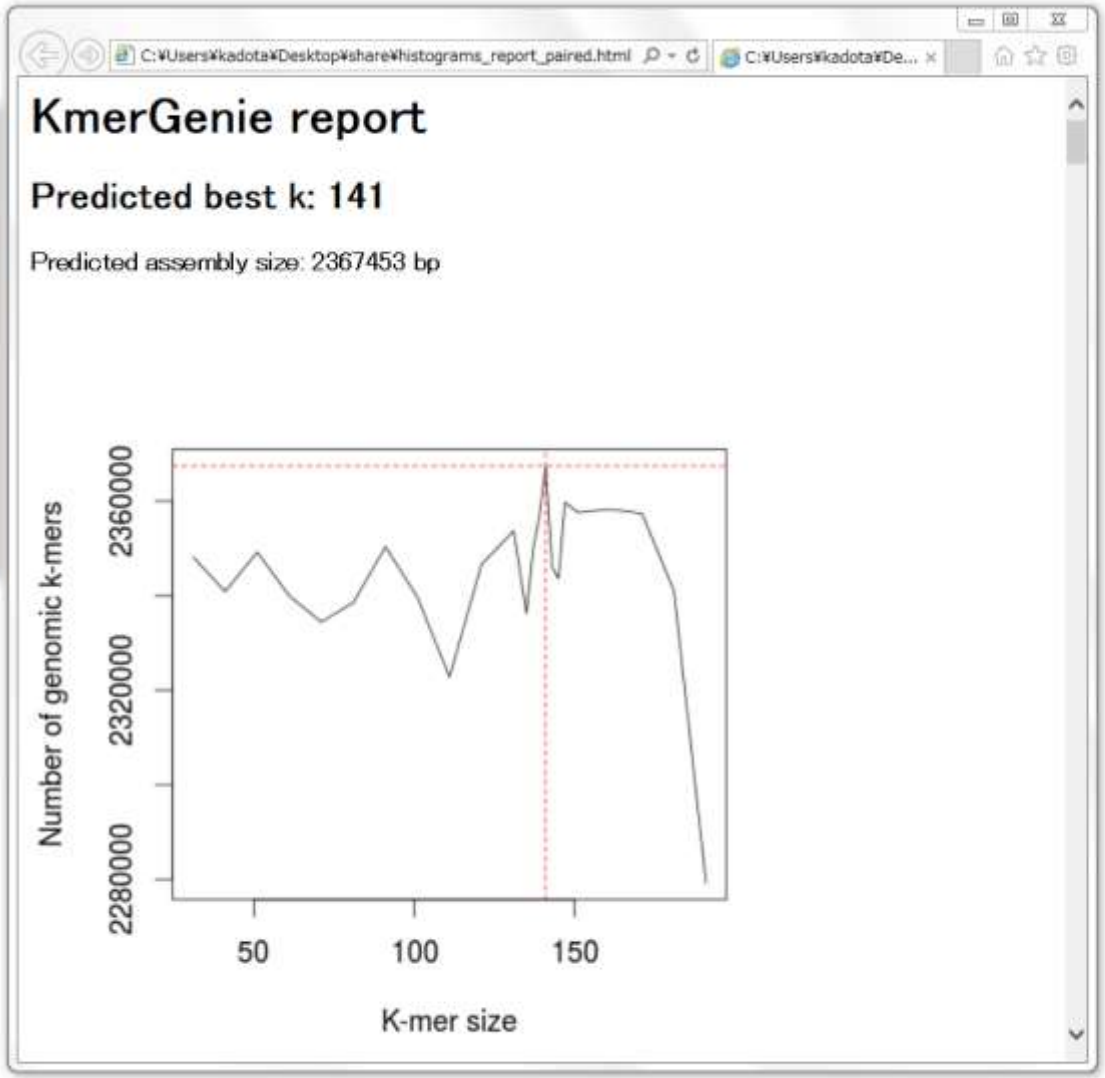
W11-9と同様に画面が一気に流れる。ここでは、(single-endの結果ファイルがhistograms_report.htmlとして既に存在するので)②共有フォルダ経由でホストOSのshareフォルダにhistograms_report_paired.htmlという名前でコピーして、ホストOS上でhtmlファイルを眺める



W11-11：確認

single-end (k=87)とpaired-end (k=141)で推奨k値は大きく異なるものの、ゲノムサイズの推定値はほとんど変わらないことがわかる(single-endは2356713 bp、paired-endは2367453 bp)





Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

第6回原稿PDFのp45

①の部分の話です。②目的は、アセンブリ結果からの短い配列の除外

億バイト）であったことを思い出せば納得できるであろう。尚、このデータの正解は、配列数が3（1 chromosome + 2 plasmids）、2,400,586 bp（約2.4MB）である[4]。*k* 値の選択の重要性がよくわかる例といえよう。

通常、Velvetを実行する場合は複数の異なる *k* 値を用いてアセンブルを行い、それらの結果を眺める［W10］。ここでは、計10個の *k* 値（k=31, 61, 91, 111, 121, 131, 151, 171, 181, 191）で実行した結果を眺め、主に配列数の観点から、*k*=171 周辺の結果が一番よさそうだと解釈する。もちろんこのデータの場合は、「真のゲノムサイズは約2.4MB」だという答えがわかった状態でアセンブリ結果の評価を行っていることになるが、実際には近縁種との比較により妥当と考えられるゲノムサイズを検討する。ここではそのような情報が得られなかったと仮定して「ゲノムサイズ推定」を行い、アセンブリ結果の評価を行う。

ゲノムサイズ推定

ゲノムサイズの推定は、フローサイトメトリー（flow cytometry）という手法を用いて実験的に求めるやり方かる。推奨の *k* 値が大きく異なるのは、paired-end（*k*=141）では single-end（*k*=87）に比べデータ量が単純に2倍になっているからである。実際には paired-end のデータを用いてアセンブリを行うので、paried-end での推奨 *k* 値（=141）の前後である *k*=131-191 あたりを手厚く探索するとよい結果が得られそうだと判断する。

配列長によるフィルタリング ①

比較的マイナーな事柄ではあるが、通常下記に示す3つの理由から、アセンブリ結果から短い配列を除外する：

②

1. MiSeq を含むショートリードの *de novo* アセンブリでは、挿入配列（insertion sequence）やリボソームRNA 遺伝子領域（rDNA）のような、ゲノム中に複数コピーが散在する反復領域（dispersed repeat）の再現は難しい。配列（コンティグ）がこれらの反復領域部分で分断されてしまうからである。アセンブリ結果に含まれる短いコンティグは、これらの反復領域の一部である場合が多く、その後の解析には大きな影響を及

W12-1：ダウンロード

multi-FASTA形式ファイルを入力として、指定した配列長未満の配列を除くPythonプログラム（fastaLengthFilter.py；第6回筆頭著者作）をダウンロード。簡単な自作プログラムは~/binに置き、そこにパスを通して使っている（ヒトもいる）。そうすることで毎回/usr/local/binなどにシンボリックリンクをはる手間が省ける

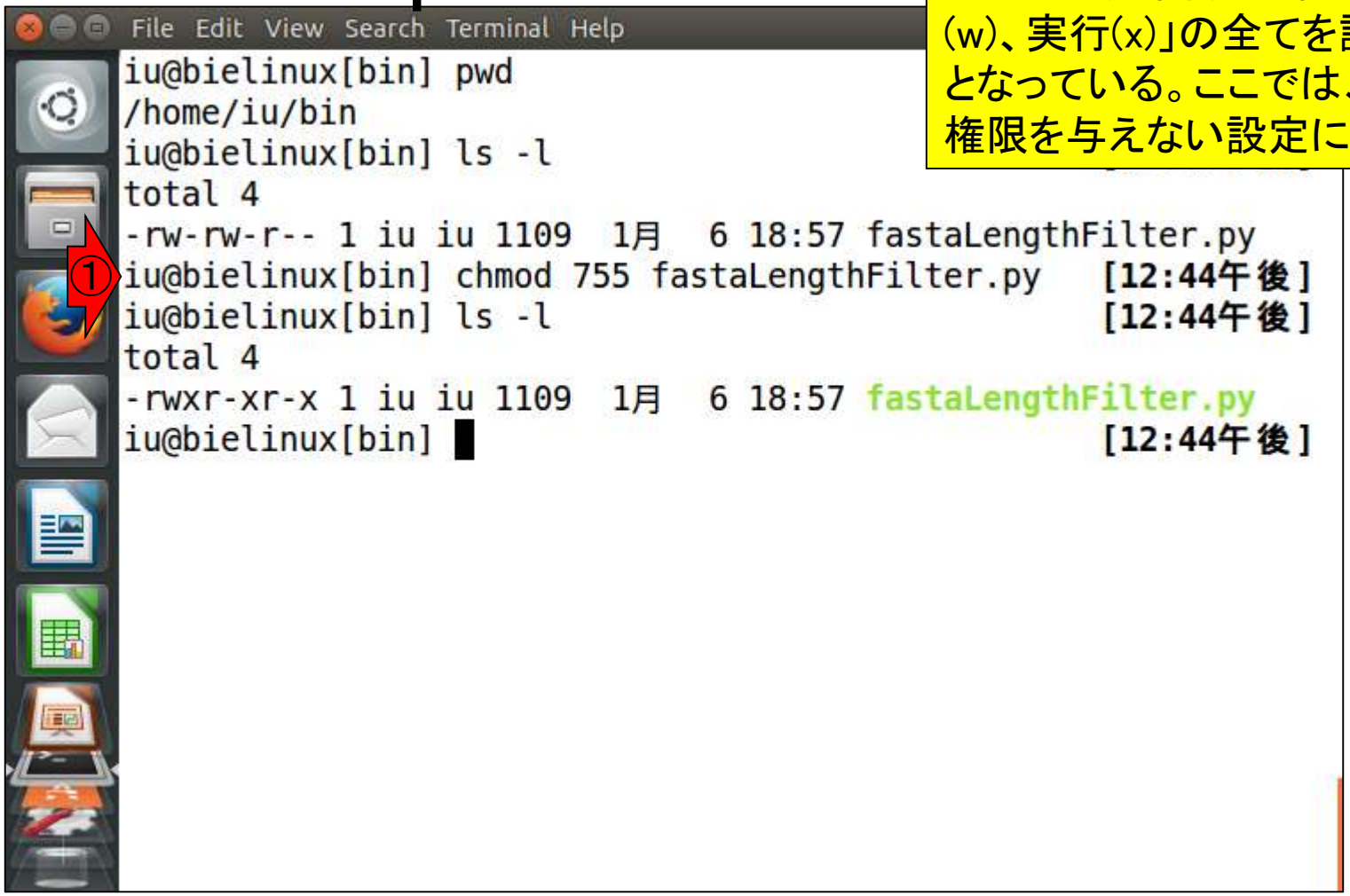
http://www.iu.a.u-tokyo.ac.jp/~kadota/book/fastaLengthFilter.py

```
#!/usr/bin/env python
# coding:utf-8

import math
import sys

class Fasta():

    @staticmethod
    def read(inputfile):
        a = open(inputfile).read()
        entries = a.split(">")[1:]
        for entry in entries:
            lines = entry.split("¥n")
            fasta = Fasta()
            fasta.header = ">" + lines[0]
            fasta.seq = "".join(lines[1:])
            fasta.length = len(fasta.seq)
            yield fasta

def main(fileName, threshold=0):
    seqList = [seq for seq in Fasta.read(fileName) if seq.length >= threshold]
    seqList.sort(key=lambda seq: seq.length, reverse=True)
    digit = int(math.log10(len(seqList))) + 1
    for i, seq in enumerate(seqList):
        print ">sequence" + str(i + 1).zfill(digit)
        print seq.seq

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print
        print "¥tRemove sequences shorter than threshold."
        print "¥tUsage : fastaLengthFilter.py <filename> <threshold>"
        print
        exit()

    fileName, threshold = sys.argv[1:3]
    main(fileName, int(threshold))
```

W12-1:ダウンロード

①ホームディレクトリに移動し、binディレクトリがないことを確認して、②mkdirで作成。③binディレクトリに移動し、④fastaLengthFilter.pyが存在するURLを指定してwget。⑤(実行権限があればそうなるが)ダウンロードしたファイルが緑色になっていないので、⑥実行権限(第4回W3-1)が自分(この場合iu)にもないことを認識する

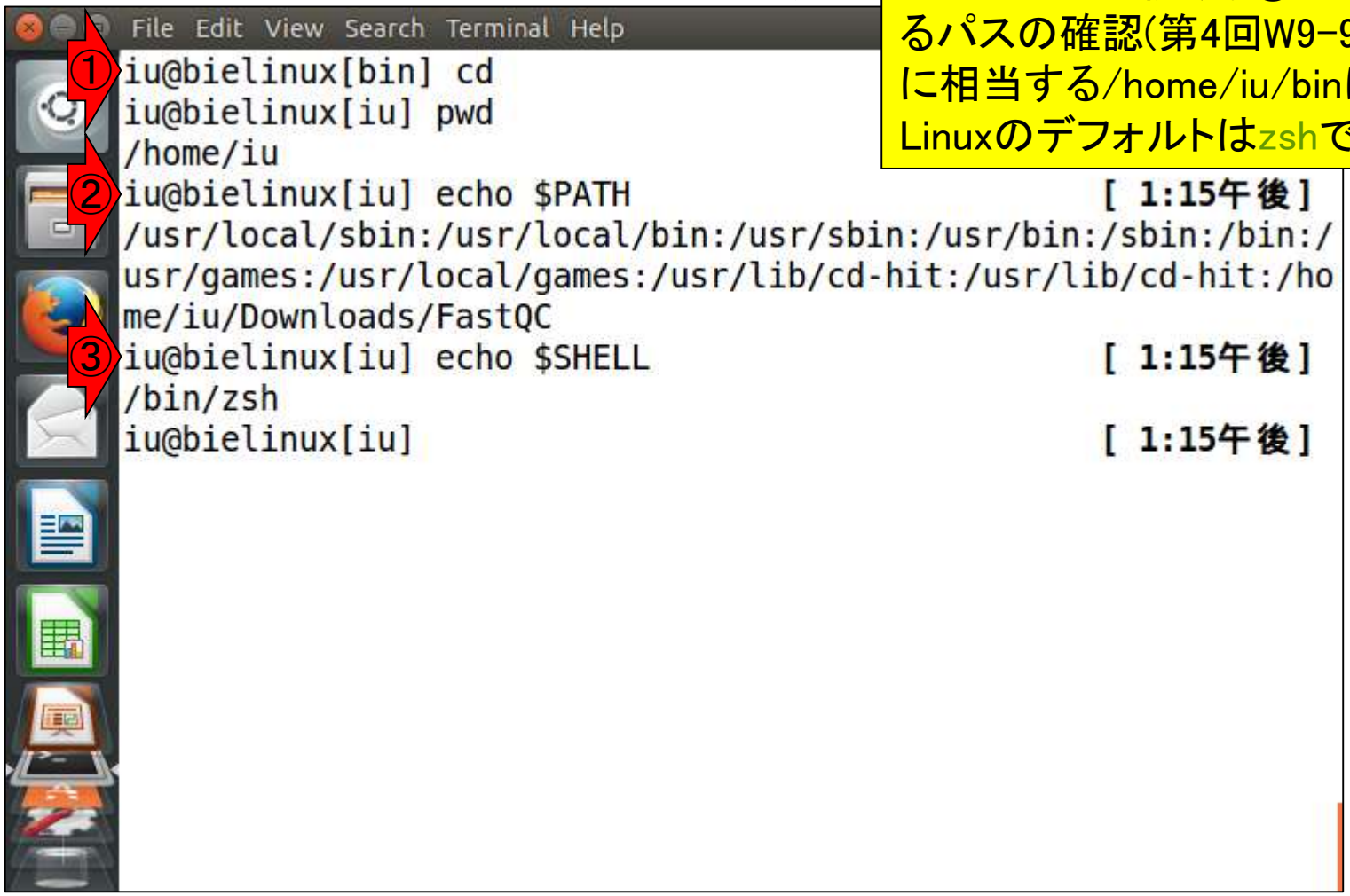
W12-2 : permission



①chmodで実行権限を変更。オプションで「755」とすると「rwxr-xr-x」になる。最初の3文字分は自分の権限に関する部分。「読み込み(r)、書き込み(w)、実行(x)」の全てを許可するという意味でrwxとなっている。ここでは、自分以外には書き込み権限を与えない設定にしている。私はいつも755

W12-3：パスを通す

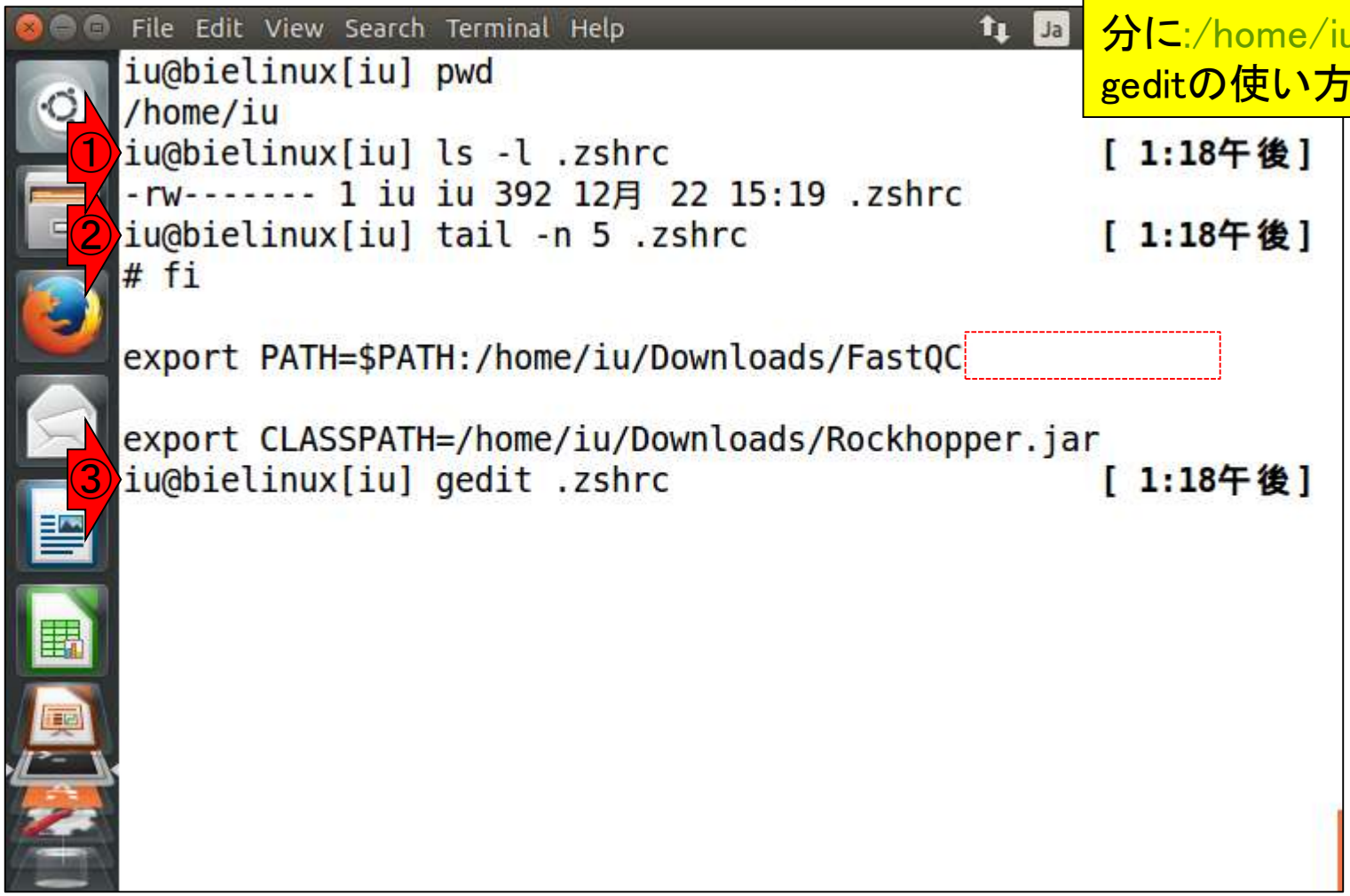
~/binにパスを通すための現状確認。①(後でホームディレクトリの.zshrcファイルをいじるので…)ホームディレクトリに移動。②echo $PATHで現在通っているパスの確認(第4回W9-9)。確かに現状では、~/binに相当する/home/iu/binは存在しない。③Bio-Linuxのデフォルトはzshであることを一応確認



W12-3：パスを通す

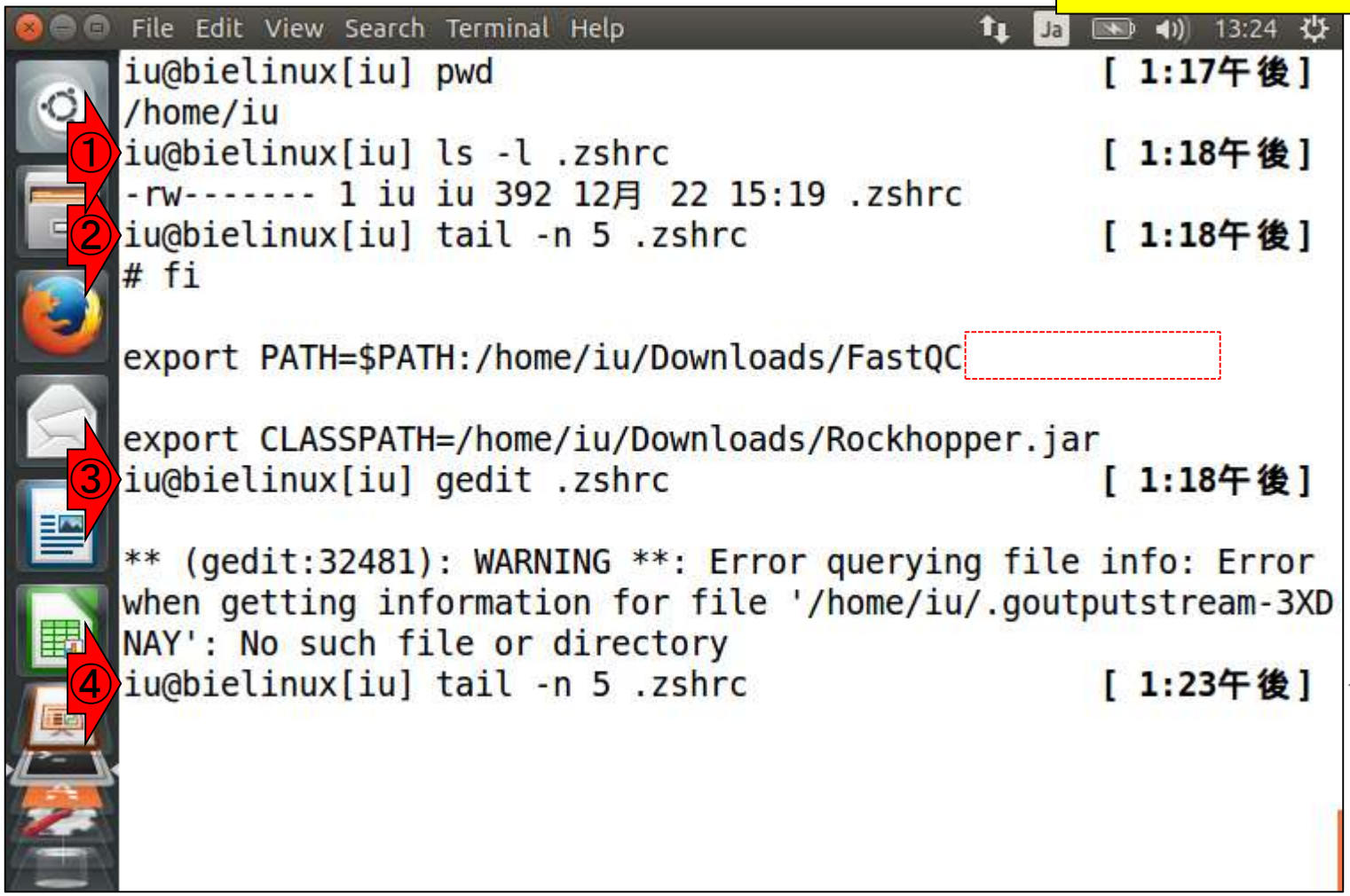
①.zshrcファイルの存在確認と②最後の5行分を表示。③テキストエディタgedit(viやemacsでもよい)を用いて、赤枠部分に:/home/iu/binを追加して保存する。geditの使い方は第4回W10-3にもあり

~/Desktop/backupに.zshrc_orgがあります。消しちゃったヒトはそれで復旧してください



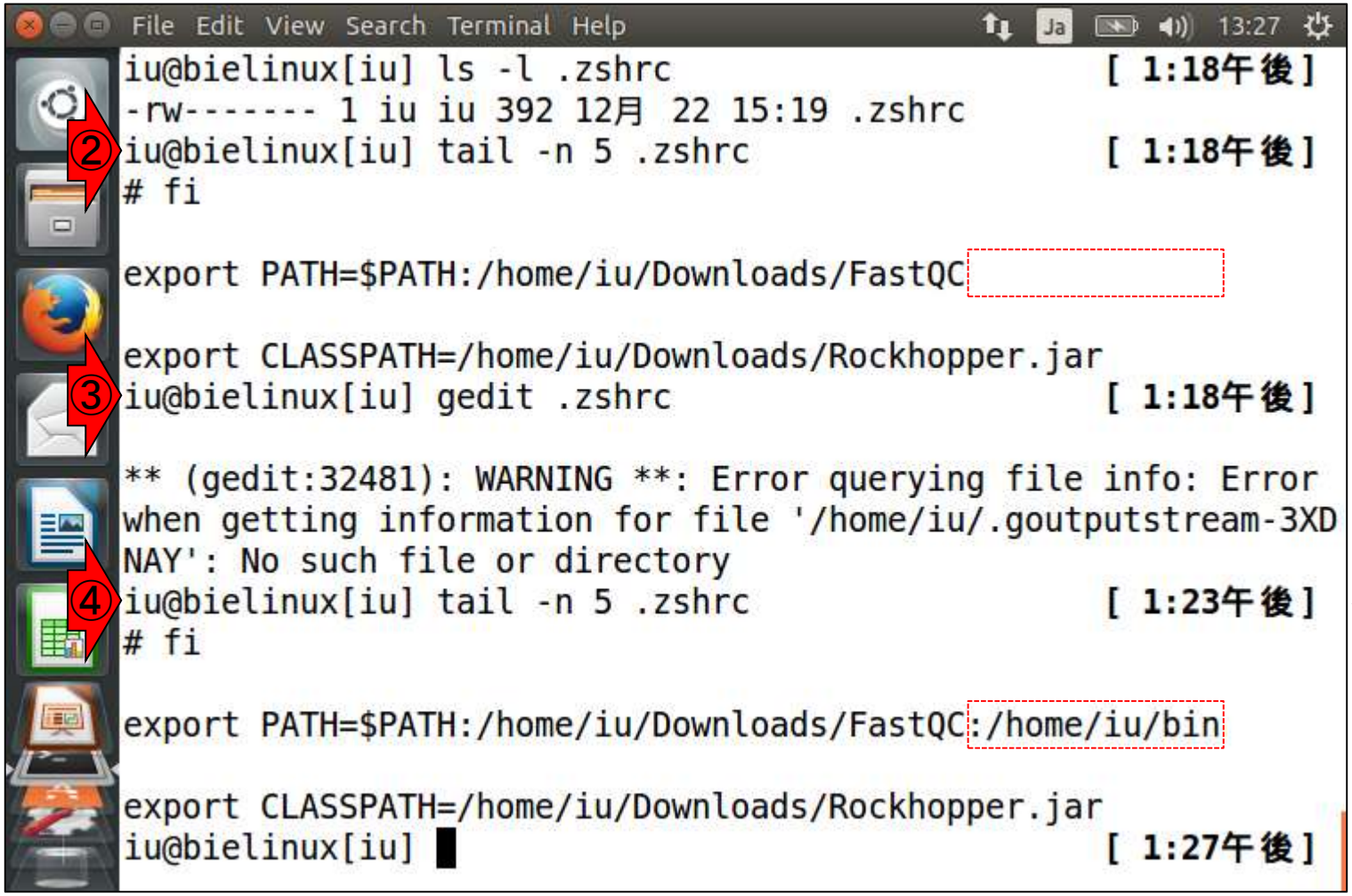
W12-3：パスを通す

赤枠部分に:/home/iu/binを追加して保存し、geditを終了した後の状態。④もう一度.zshrcファイルの最後の5行分を表示



確かに:/home/iu/binが追加されている

W12-3:パスを通す



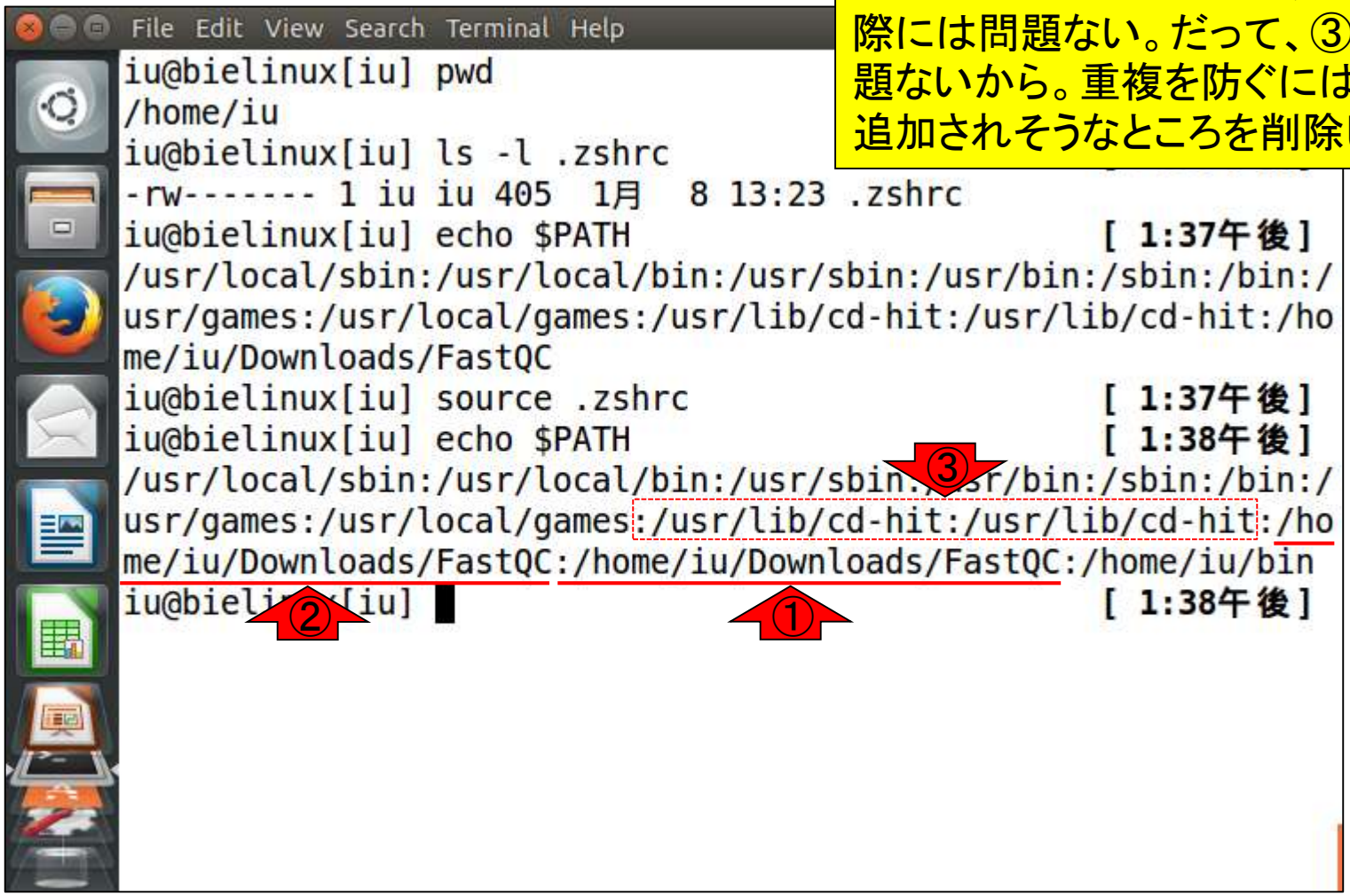
W12-4：確認

①「gedit .zshrc」を実行したこのターミナルは、まだターミナル起動時の環境設定のまま。②sourceコマンドで編集後の環境設定ファイル(.zshrc)を再読み込みしておく。その後もう一度echo $PATHで確認すると、③確かに自分が追加した赤枠のものが見えている。基本的にこれでもうOK

W12-4：確認

参考

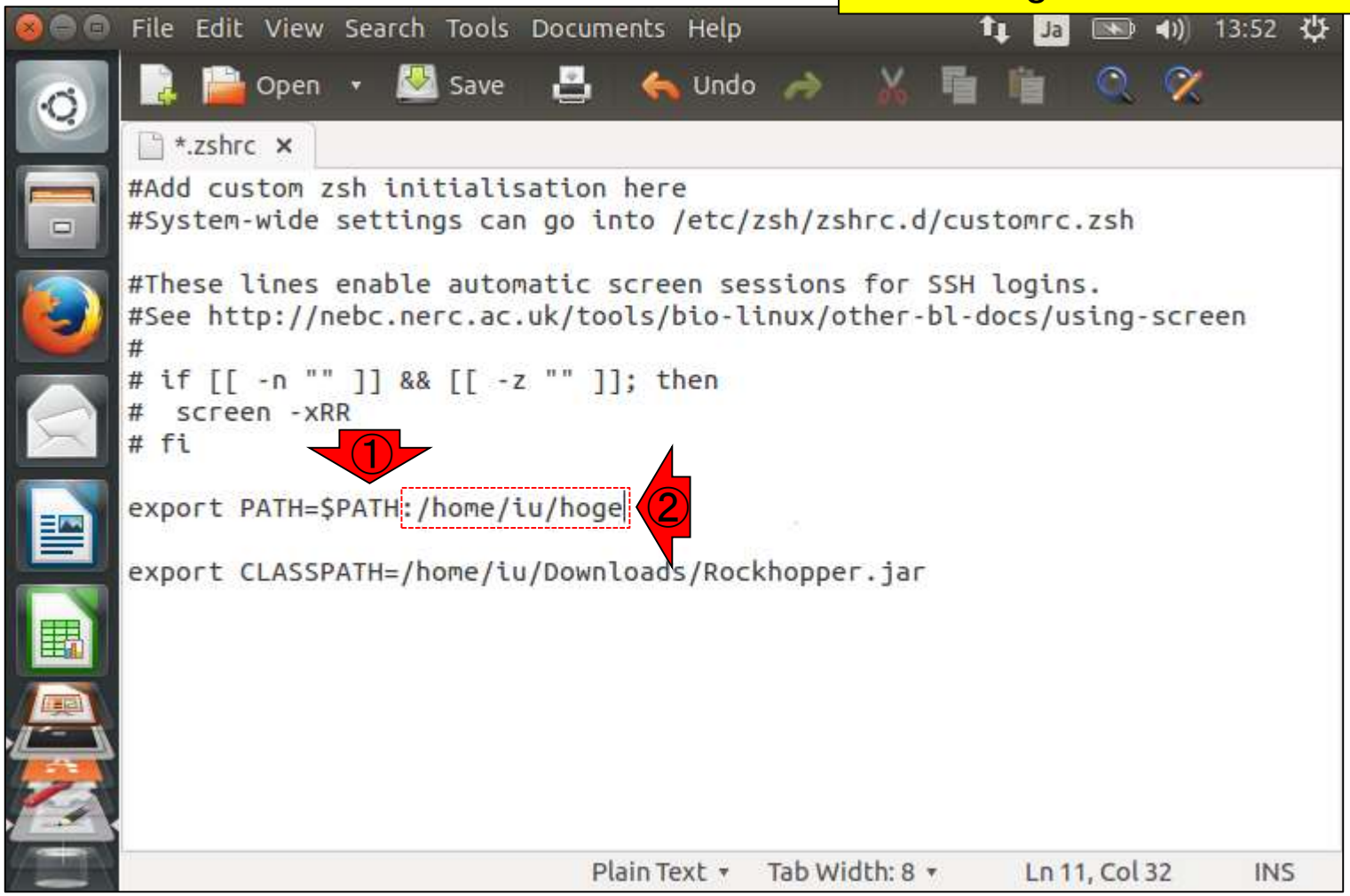
以下はおまけです。よく見ると、①の赤下線部分が重複しており、②左側にあるものと全く同じ。やらかしてしまったのではないかと思うが、重複しているだけで実際には問題ない。だって、③ここにも重複があるが、問題ないから。重複を防ぐには、.zshrc編集時に余分に追加されそうなところを削除しておけばよい。つまり…



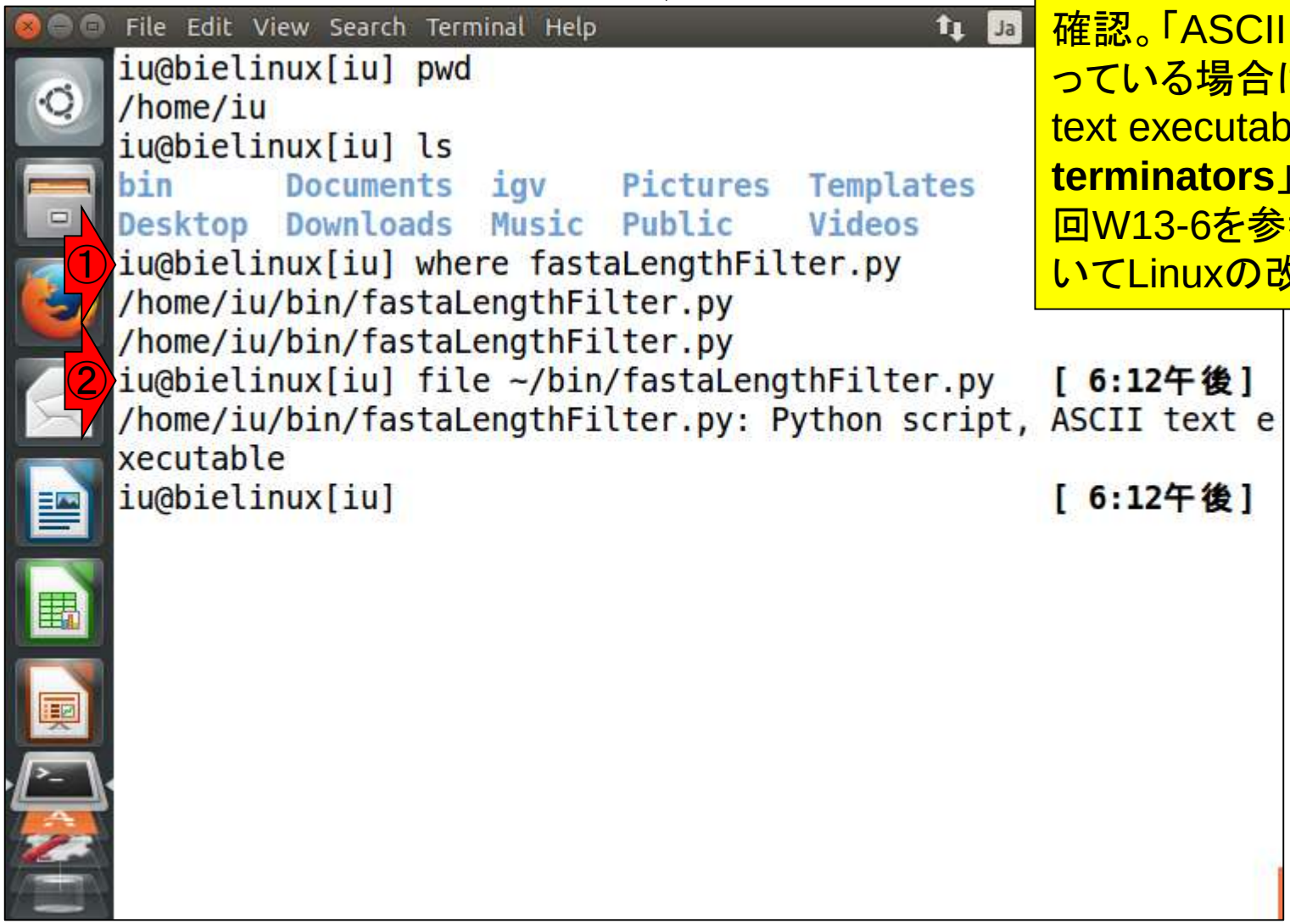
W12-4：確認

参考

例えば~/hogeディレクトリ以下のファイルにパスを通したい場合は、①の「$PATH」の部分のみ残して、②赤枠の~/hogeディレクトリの絶対パスを追加すればよい

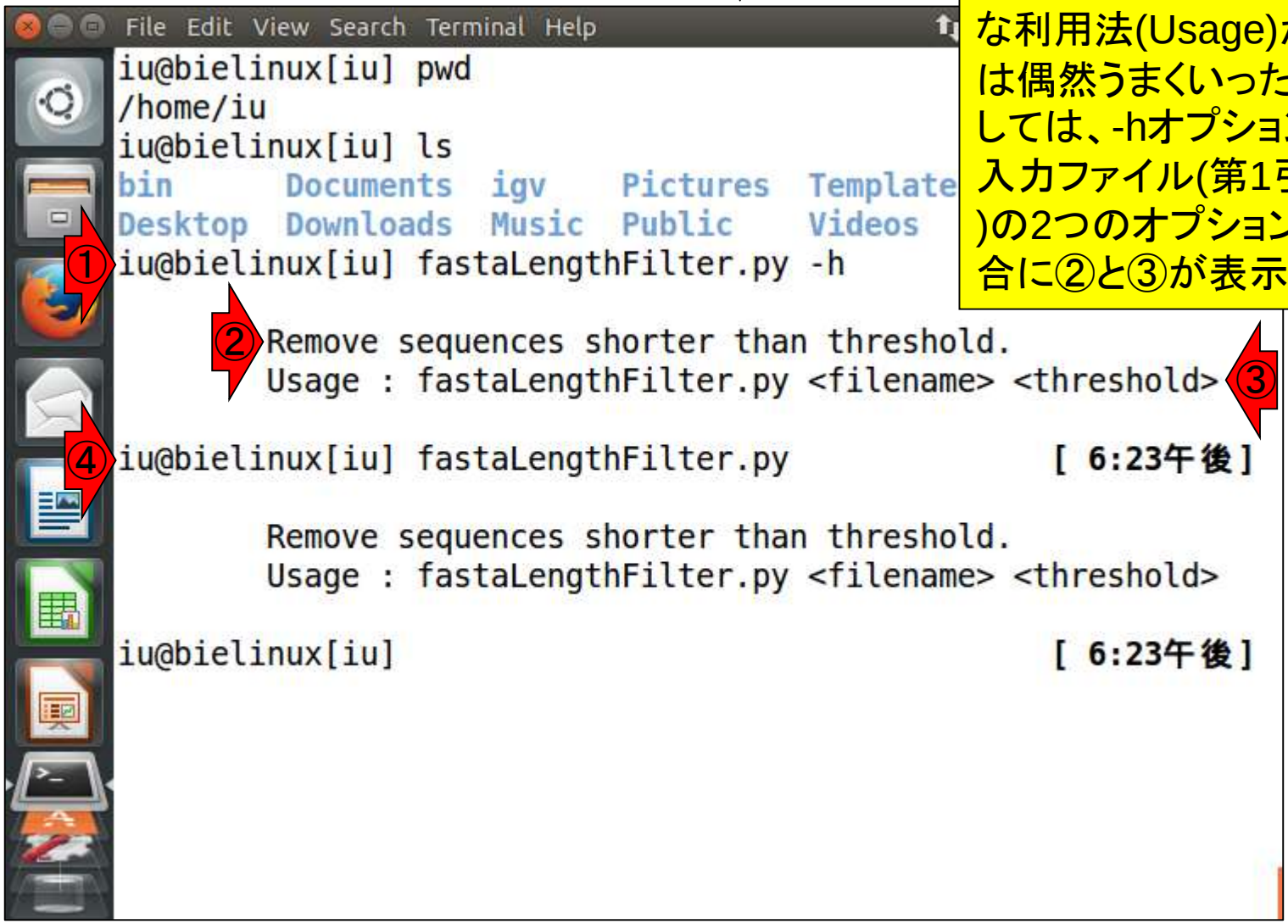


W12-5：パスと改行コード



①whereコマンドでパスが通っていることを念のため確認。②fileコマンドでLinuxの改行コードになっているかどうか確認。「ASCII text executable」で終わっている場合は問題ないが、「ASCII text executable, **with CRLF line terminators**」となっている場合は、第4回W13-6を参考にして、nkfコマンドを用いてLinuxの改行コードに変換しておく

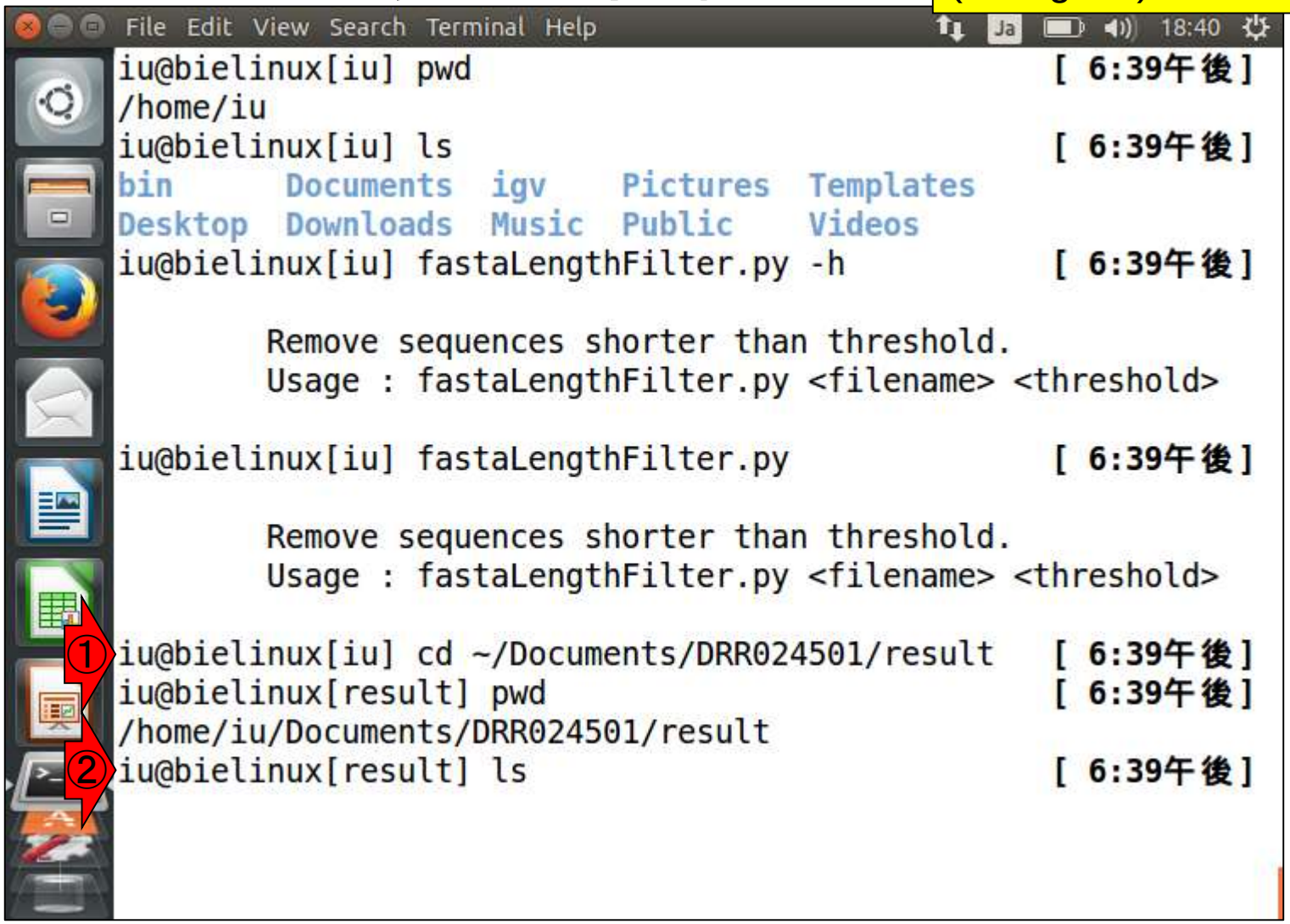
W12-6：ヘルプを表示



①-hをつけてfastaLengthFilter.pyを実行。②このプログラムが何をするものなのかについての簡単なdescription、および③基本的な利用法(Usage)が表示される。④実は①は偶然うまくいっただけ。このプログラムに関しては、-hオプションを受け付けてはおらず、入力ファイル(第1引数)とthreshold(第2引数)の2つのオプションを同時入力しなかった場合に②と③が表示される。つまり④が正解

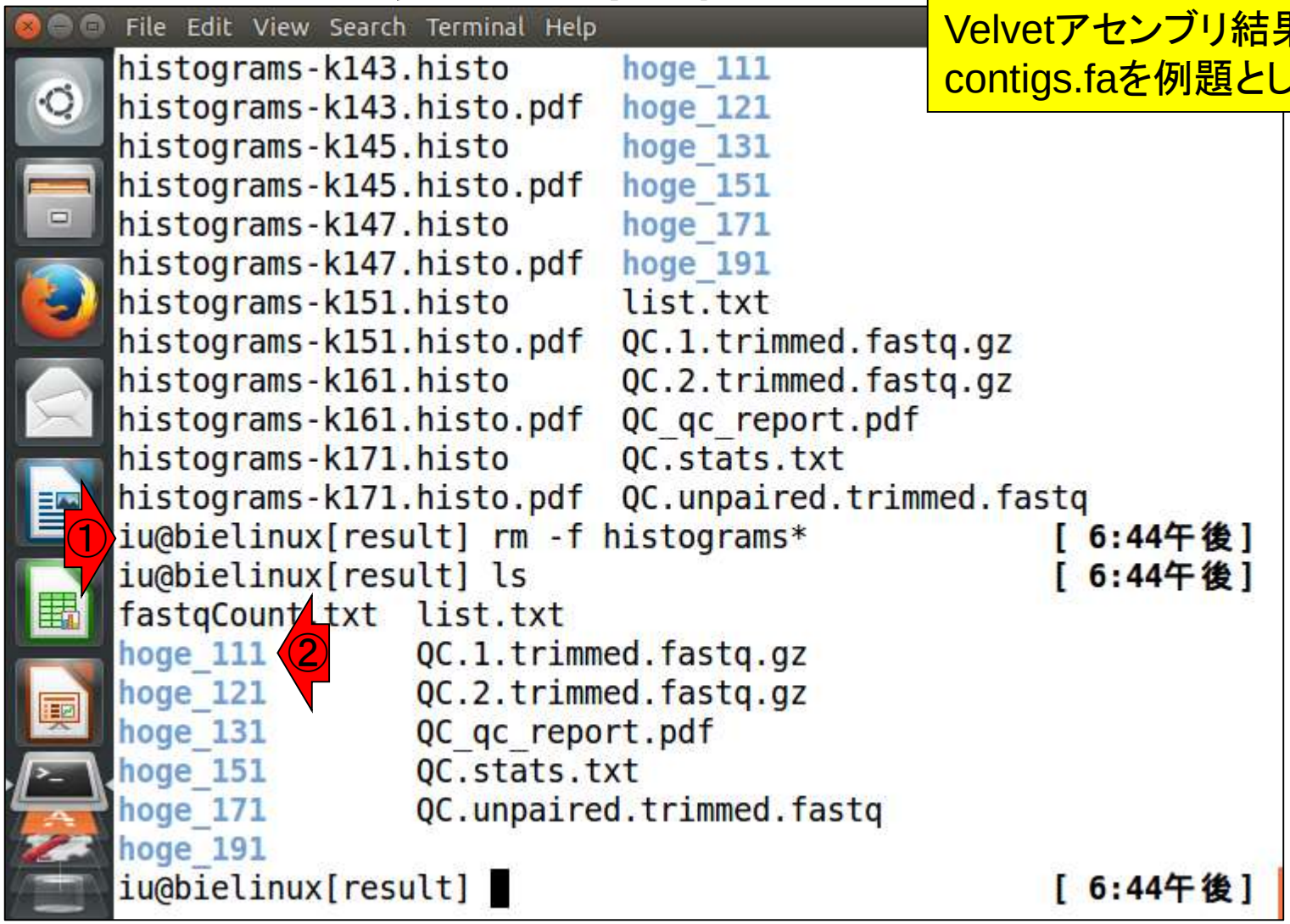
W12-7：実行準備

fastaLengthFilter.pyを実行すべく、①W10-5で作成したVelvetアセンブリ結果ファイル(contigs.fa)**に近い**ディレクトリに移動し、②ls



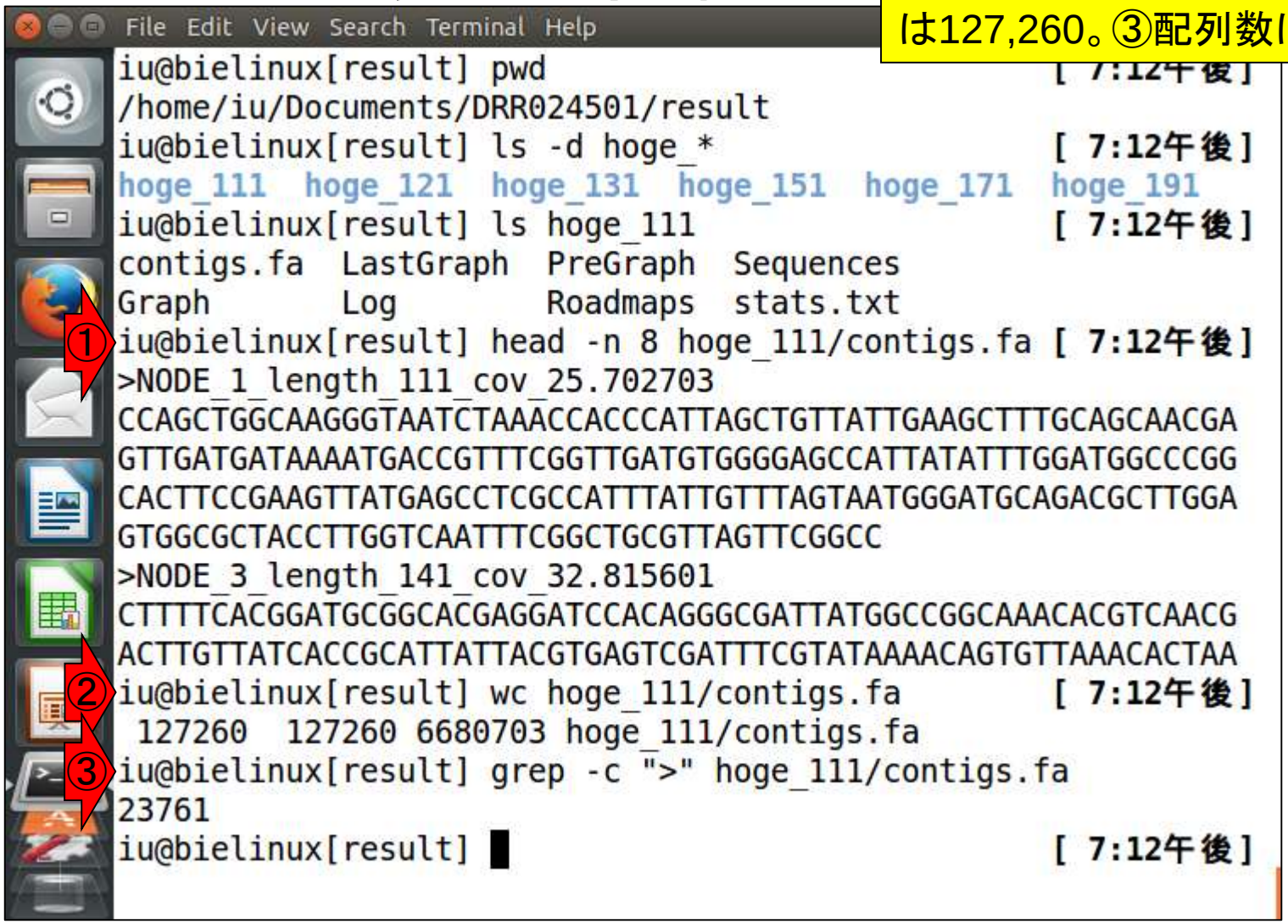
W12-7：実行準備

W11-11で作成したKmerGenie実行結果ファイル群(histograms*)が一気に表示されて見づらいので①削除。とりあえず②k=111で実行したVelvetアセンブリ結果フォルダhoge_111中のcontigs.faを例題としてフィルタリングを行う



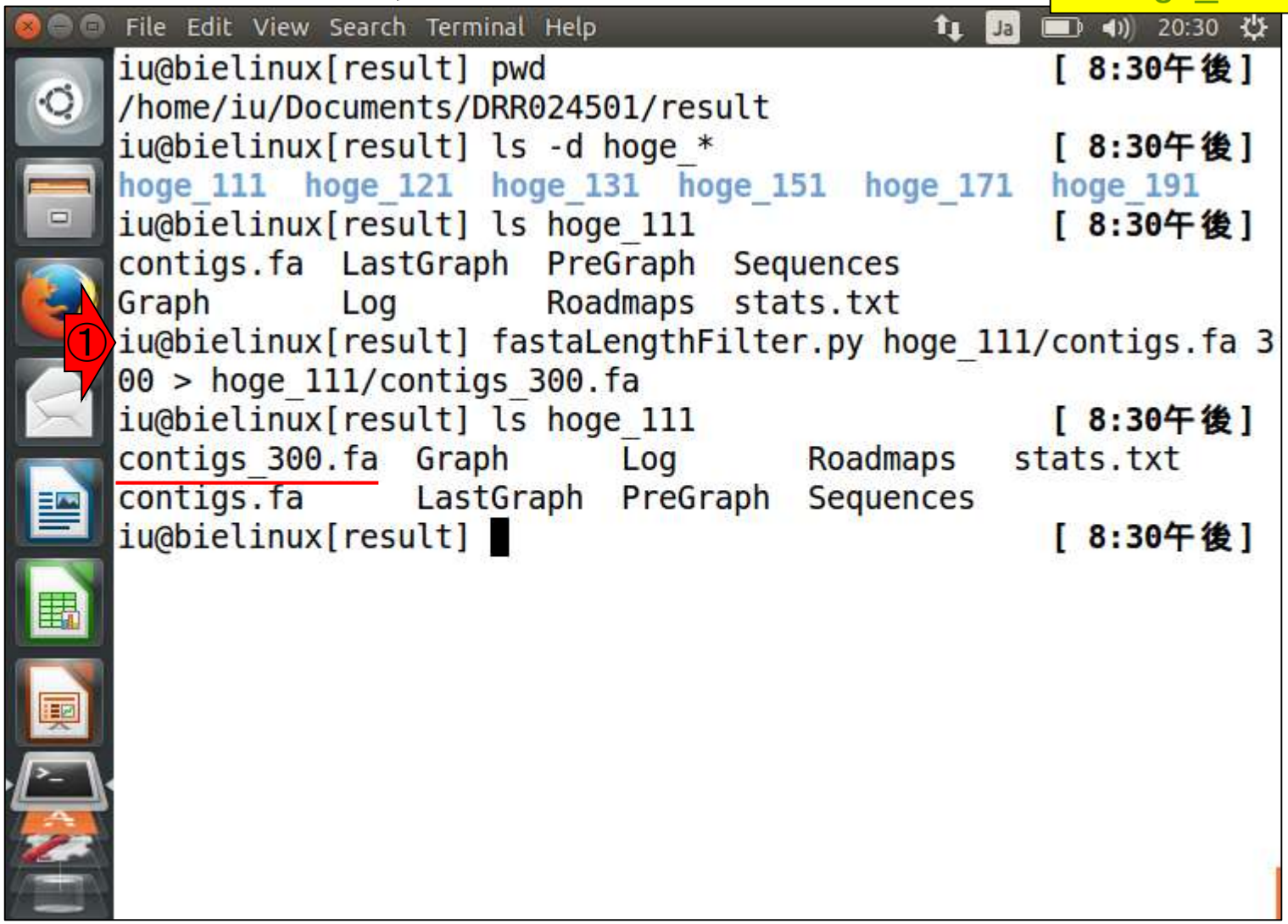
W12-7：実行準備

①hoge_111/contigs.faの最初の8行分を表示。description行は「NODE_...」みたいな感じになっていることがわかる。②wc実行結果を表示。行数は127,260。③配列数は23,761で、W10-6と同じ



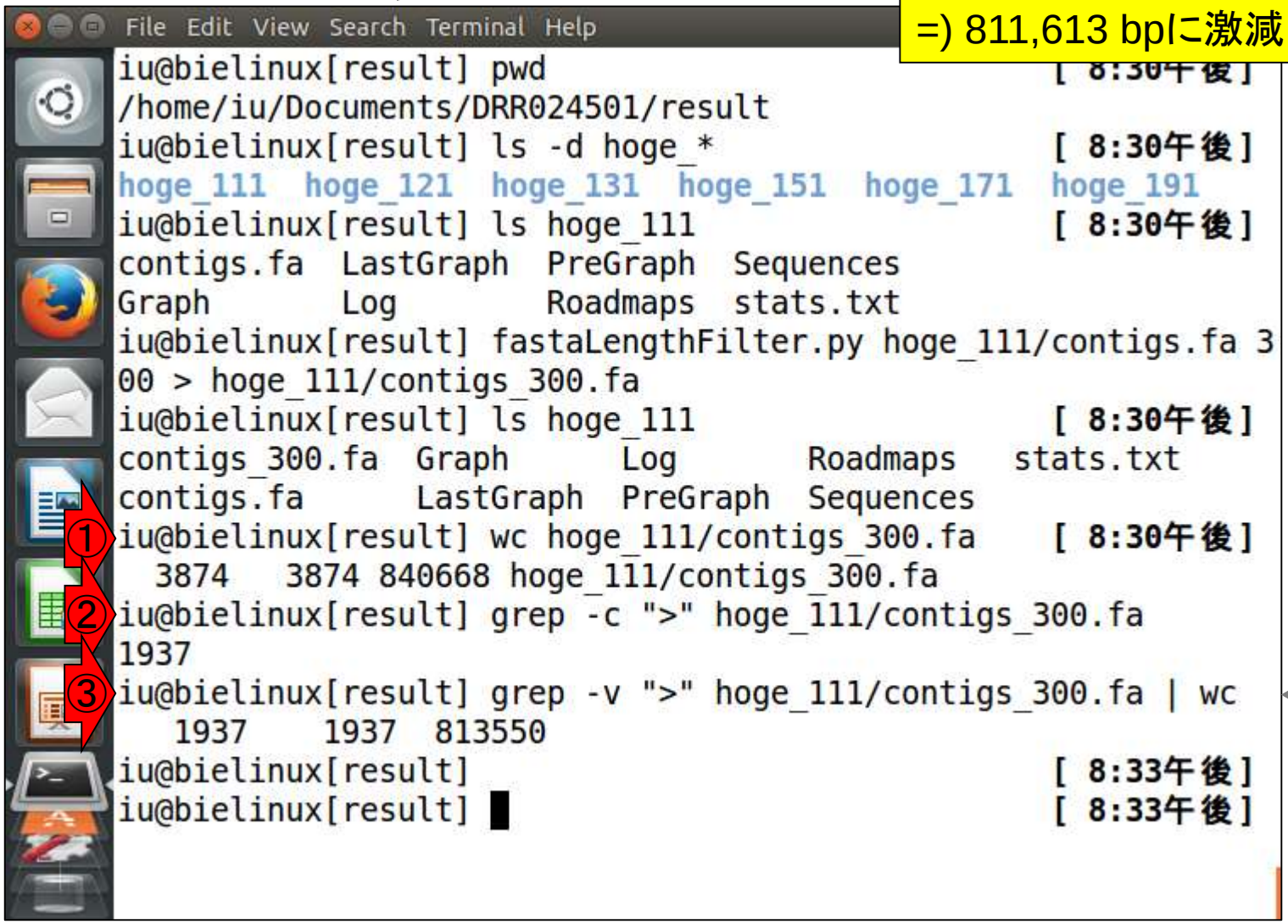
W12-8：実行

①hoge_111/contigs.faを入力として
300 bp以上の配列のみ残した結果を
contigs_300.faというファイル名で出力



W12-8：実行

①行数が127,260行から3,874行に激減している!②配列数も23,761個から1,937個に激減。③総塩基数も5,718,204 bpから(813,550 – 1,937 =) 811,613 bpに激減していることがわかる



W12-8：実行

④の手順でも同じ結果になることを確認しており、プログラムのバグではない

```
File  Edit  View  Search  Terminal  Help                    Ja  20:33
iu@bielinux[result] pwd                                    [ 8:30午後]
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls -d hoge_*                           [ 8:30午後]
hoge_111  hoge_121  hoge_131  hoge_151  hoge_171  hoge_191
iu@bielinux[result] ls hoge_111                            [ 8:30午後]
contigs.fa  LastGraph  PreGraph  Sequences
Graph       Log
iu@bielinux[res
00 > hoge_111/c
iu@bielinux[res
contigs_300.fa
contigs.fa
①iu@bielinux[res
  3874   3874 8
②iu@bielinux[res
1937
③iu@bielinux[result]
   1937    1937  81
iu@bielinux[result]
iu@bielinux[result]                                        [ 8:33午後]
```

(Rで)塩基配列解析

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

(last modified 2015/12/22, since 2011)

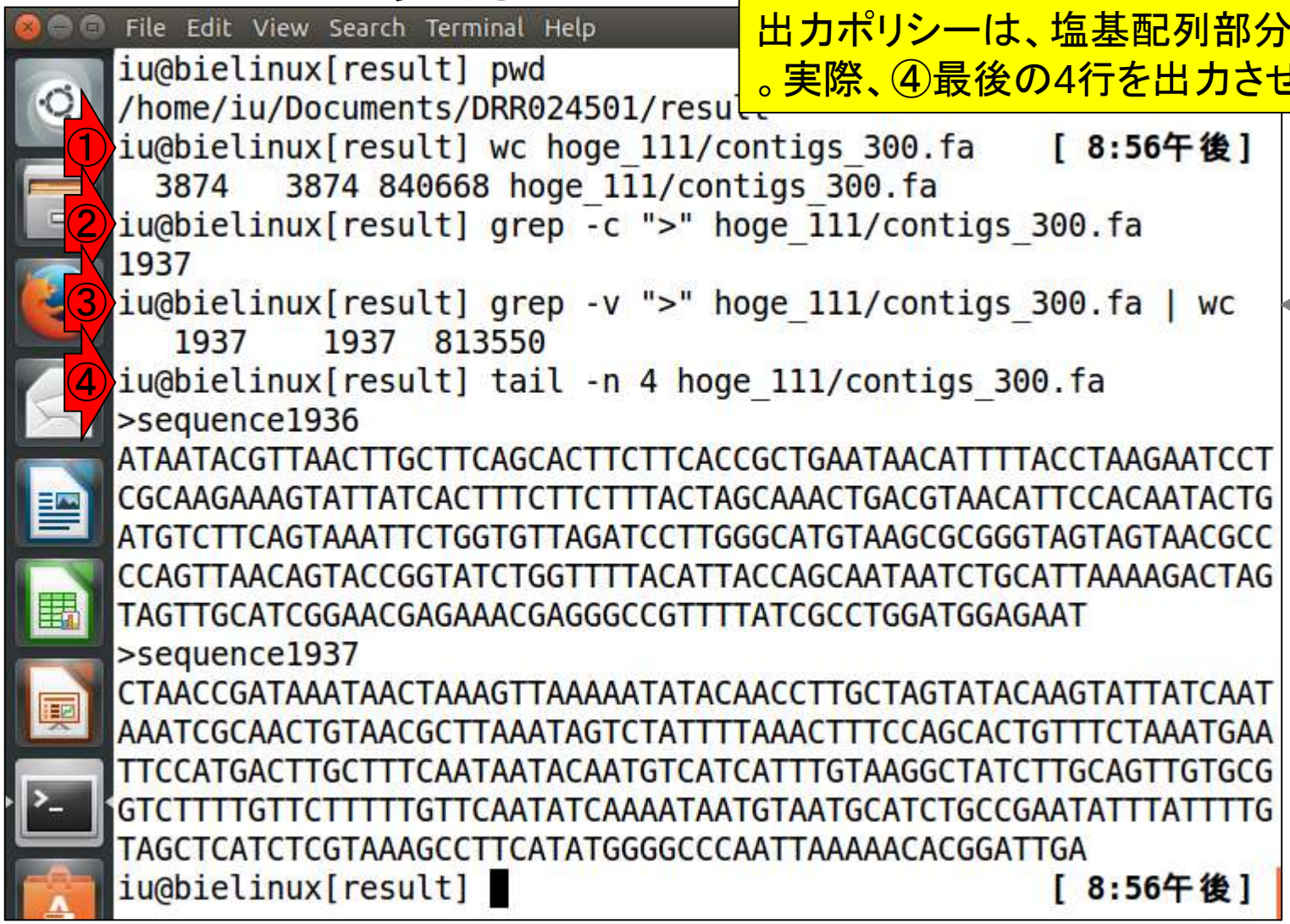
What's new?

- この
- Rと
- (Win
- (201
- 多群

- 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 2015/09/12)
- 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 ④ t 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 (l st modified 2014/08/21)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2015/02/26)
- 前処理 | フィルタリング | 任意のIDを含む配列を抽出 (last modified 2013/06/18)

W12-8：実行

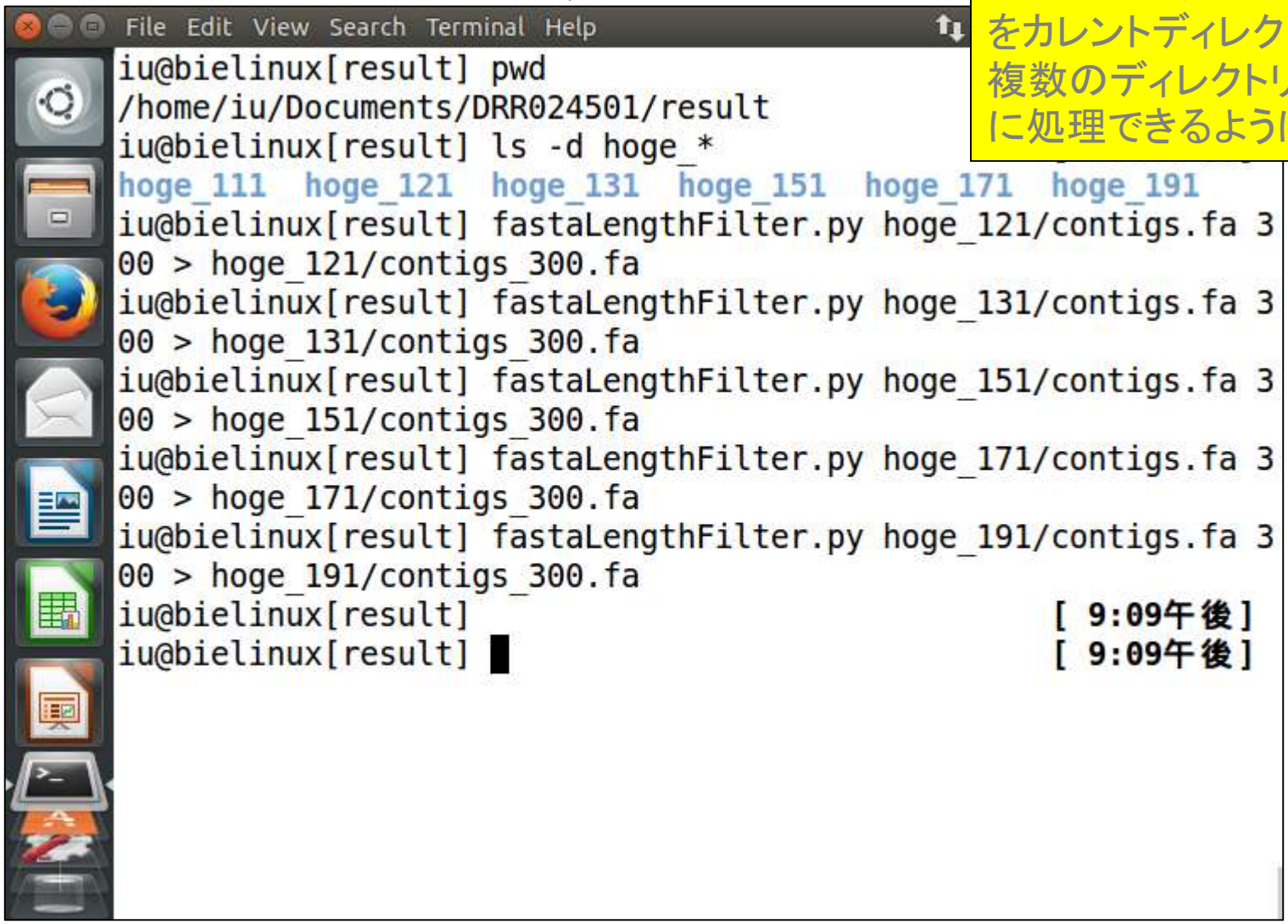
①行数が3,874行、②配列数が1,937個に激減。③description行以外の行数も1,937行、という結果に違和感を覚えるかもしれない。これはfastaLengthFilter.pyの出力ポリシーは、塩基配列部分が**1配列1行**だからである。実際、④最後の4行を出力させると、2配列分表示される



 参考

W12-9：全て実行

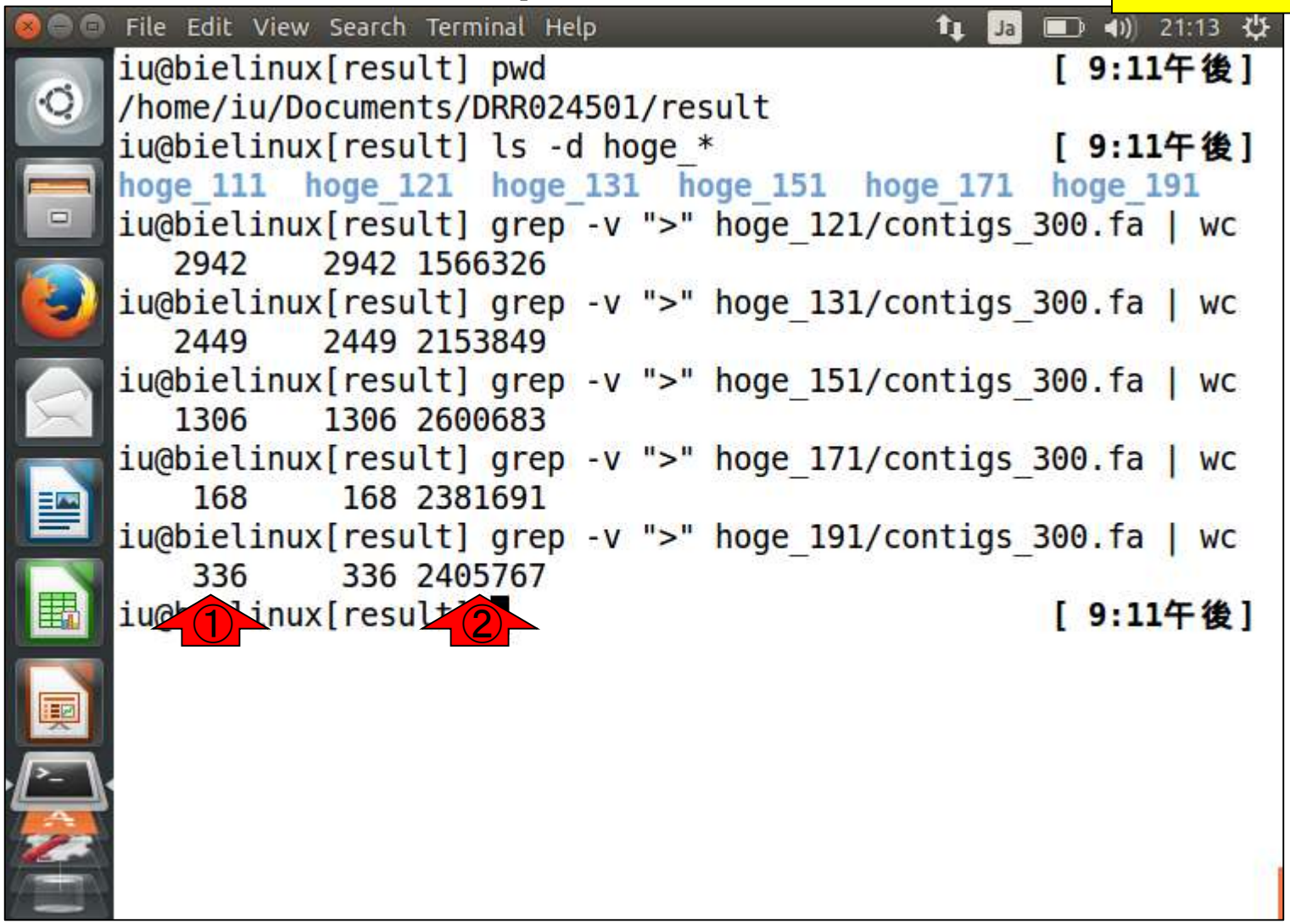
同じ作業を、他のk値のアセンブリ結果に対しても実行。ないヒトはやったつもりでエアーハンズオン。入力ファイルがあるディレクトリをカレントディレクトリにしなかった理由は、複数のディレクトリ上にあるファイルを一気に処理できるようにしたかったからです



 参考

W12-9：結果を確認

フィルタリング結果ファイルに対して、①行数や②ファイルサイズを表示。ないヒトはエアーハンズオン



W12-9：結果を確認

- 全アセンブル結果に対して実行[W12-9]
  300 bp以上の配列のみ残した結果をcontigs_300.faというファイル名で出力。

```
pwd
ls -d hoge_*
fastaLengthFilter.py hoge_121/contigs.fa 300 > hoge_121/contigs_300.fa
fastaLengthFilter.py hoge_131/contigs.fa 300 > hoge_131/contigs_300.fa
fastaLengthFilter.py hoge_151/contigs.fa 300 > hoge_151/contigs_300.fa
fastaLengthFilter.py hoge_171/contigs.fa 300 > hoge_171/contigs_300.fa
fastaLengthFilter.py hoge_191/contigs.fa 300 > hoge_191/contigs_300.fa

pwd
ls -d hoge_*
grep -v ">" hoge_121/contigs_300.fa | wc
grep -v ">" hoge_131/contigs_300.fa | wc
grep -v ">" hoge_151/contigs_300.fa | wc
grep -v ">" hoge_171/contigs_300.fa | wc
grep -v ">" hoge_191/contigs_300.fa | wc
```

実際に行っているのは、赤枠内のコピペ。系統的に記述できている（違いはディレクトリ名部分のみ）ことがわかる。赤枠部分のみからなるファイルは、一般的なシェルスクリプトといえる。私はこういうものを作って一気に系統的な解析を行います

W12-10：これまでのまとめ

(a) フィルタリング前				(b) フィルタリング後		
k-mer	コンティグ数	総塩基数	ウェブ資料	コンティグ数	総塩基数	ウェブ
31	29502	4077679	W10-4			
61	15445	3886574	W10-4			
91	8583	3412266	W10-4			
111	23761	5718204	W10-5			
121	15776	4690144	W10-5	2942	1563384	W12-9
131	8398	3710829	W10-5	2449	2151400	W12-9
151	1306	2599377	W10-5	1306	2599377	W12-9
171	168	2381523	W10-5	168	2381523	W12-9
181	198	2386048	W10-1			
191	336	2405431	W10-5	336	2405431	W12-9

配列長(< 300 bp)によるフィルタリング前は、最大で約5.7MB (k=111でのVelvetアセンブリ結果)というゲノムサイズに達していたが、フィルタリング後は最大でも約2.6MB (k=151の結果)となっていることがわかる

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

DDBJ Pipeline

	(a) フィルタリング前			(b) フィルタリング後		
k-mer	コンティグ数	総塩基数	ウェブ資料	コンティグ数	総塩基数	ウ
31	29502	4077679	W10-4			
61	15445	3886574	W10-4			
91	8583	3412266	W10-4			
111	23761	5718204	W10-5			
121	15776	4690144	W10-5	2942	1563384	W12-9
131	8398	3710829	W10-5	2449	2151400	W12-9
151	1306	2599377	W10-5	1306	2599377	W12-9
171	168	2381523	W10-5	168	2381523	W12-9
181	198	2386048	W10-1			
191	336	2405431	W10-5	336	2405431	W12-9

DDBJ PipelineでVelvetが利用可能である。①実際にk=131で*de novo*アセンブリを行う一連の手順を示し、Bio-Linux上の数値（W10-5やW12-9）と同じ結果が得られて安心するところまでがW13からW18の内容。ダイジェストで紹介

第6回原稿PDFのp47

①以降の話です。②DDBJ Pipelineは、ウェブブラウザ経由で(しかも無料で)遺伝研スパコン上でNGS解析ができる大変ありがたいツールです

DDBJ Pipeline（概要からアカウント作成まで）

①

これまで行ってきた解析結果は、全データの約1/10のリード数からなるサブセットについてのものである。乳酸菌を含むバクテリア程度であれば、オリジナルデータを用いた*de novo*アセンブリも実行可能かもしれない。どの程度のデータ量まで手元のPCで可能か？どの程度メモリを積んだノートPCであればこのデータ量の解析が可能か？といったデータ量やスペックに関するグレーゾーンの議論はここでは行わない。計200万リードからなるpaired-end RNA-seqデータの*de novo* transcriptome assemblyが2GBメモリでできた（第5回のW5-2）こと、本稿で示した合計約60万リードからなるpaired-endデータのVelvetアセンブリがマニュアル通りのやり方でできたことなど、実体験の積み重ねのほうが有意義であろう。

データ量が大きな配列解析を行う手段の1つは、国立遺伝学研究所（以下、遺伝研）が運用するスーパーコンピュータシステム（以下、スパコン）[18]の利用である。本連載で何度か紹介したDDBJ Read Annotation Pipeline（以下、DDBJ Pipeline）[3]は、遺伝研・大量遺伝情報研究室において開発・運用されているNGS解析に特化した遺伝研スパコンを遠隔利用できるクラウドウェブサービスの1つで

②

かには、DDBJ Pipeline
ンポートまたはアップロードで遺伝研スパコンに設置する作業と読み替えればよい。この作業には、3通りのやり方が存在する：

①DRA/ERA/SRAから始まるIDの指定（DRAからのインポート）
②FTP経由でアップロード
③HTTP経由でアップロード

公共DBで公開されているNGSデータを解析したい場合は、①でDRA IDを与えればよい。但し、本稿で取り扱っているDRR024501というIDは、DRR～でありDRA～ではない。つまりDRR024501を与えてもエラーとなるため、DRR024501を頼りに公共DBを眺めて指定可能なDRA ID（この場合はDRA002643）を探す必要がある[**W2; W13-3**]。手元のファイルを解析したい場合は、②FTP経由か③HTTP経由でアップロードする。FTP経由のアップロードは、FTPクライアントソフトウェア（以下、FTPソフト）を利用する[**W13-6**]。WinSCP（Windows用）やCyberduck（Macintosh用）がホストOS上にインストールされていれば、マニュアルの指示通りに設定情報

W13-1：DDBJ pipeline

①しかるべき原著論文を適切に引用するのは、エンドユーザの義務です!

スライドを見るだけ

DDBJ Pipeline（概要からアカウント作成まで）

- 国立遺伝学研究所のスーパーコンピュータシステム
  - 引用文献(たぶんこれが最新論文): Mashima et al., Nucleic Acids Res., 2016
  - 引用文献(web上で指定された公式論文): Ogasawara et al., Nucleic Acids Res., 2013
  - DDBJ Read Annotation Pipeline (DDBJ Pipeline): Nagasaki et al., DNA Res., 2013
  - 遺伝研・大量遺伝情報研究室
- Galaxy: Goecks et al., Genome Biol., 2010
- TRAPLINE: Wolfien et al., BMC Bioinformatics, 2016
- Orione: Cuccuru et al., Bioinformatics, 2014
- 遺伝研スパコンの登録ユーザ情報

①

DDBJ Pipeline（クエリファイルの登録）

- 手元のファイル[W13-4]
  W5-4で作成した解析したい手元のファイルを共有フォルダ(~/Desktop/mac_share)経由でホストOSにコピー。コピーするファイルは、QC.1.trimmed.fastq.gz(57,061,392 bytes; 約55MB)とQC.2.trimmed.fastq.gz(62,989,289 bytes; 約61MB)です。ここではQC.[0-9].*.gzで2つのファイルを指定しています。

```
cd ~/Documents/DRR024501/result

pwd
ls
cp QC.[0-9].*.gz ~/Desktop/mac_share
```


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

W13-1：DDBJ pipeline

①DDBJ Pipeline利用時には、アカウントを作成しておかねばなりません。その際に入力する「氏名・所属・利用目的」は、②遺伝研スパコンのウェブサイト上で公開されるのでご注意ください

スライドを見るだけ

DDBJ Pipeline（概要からアカウント作成まで）

- 国立遺伝学研究所のスーパーコンピュータシステム
  - 引用文献(たぶんこれが最新論文): Mashima et al., Nucleic Acids Res., 2016
  - 引用文献(web上で指定された論文): Ogasawara et al., Nucleic Acids Res., 2013
  - DDBJ Read Annotation Pipeline (DDBJ Pipeline): Nagasaki et al., DNA Res., 2013
  - 遺伝研・大量遺伝情報研究室
- Galaxy: Goecks et al., Genome Biol., 2010
- TRAPLINE: Wolfien et al., BMC Bioinformatics, 2016
- Orione: Cuccuru et al., Bioinformatics, 2014
- 遺伝研スパコンの登録ユーザ情報

①

②

DDBJ Pipeline（クエリファイルの登録）

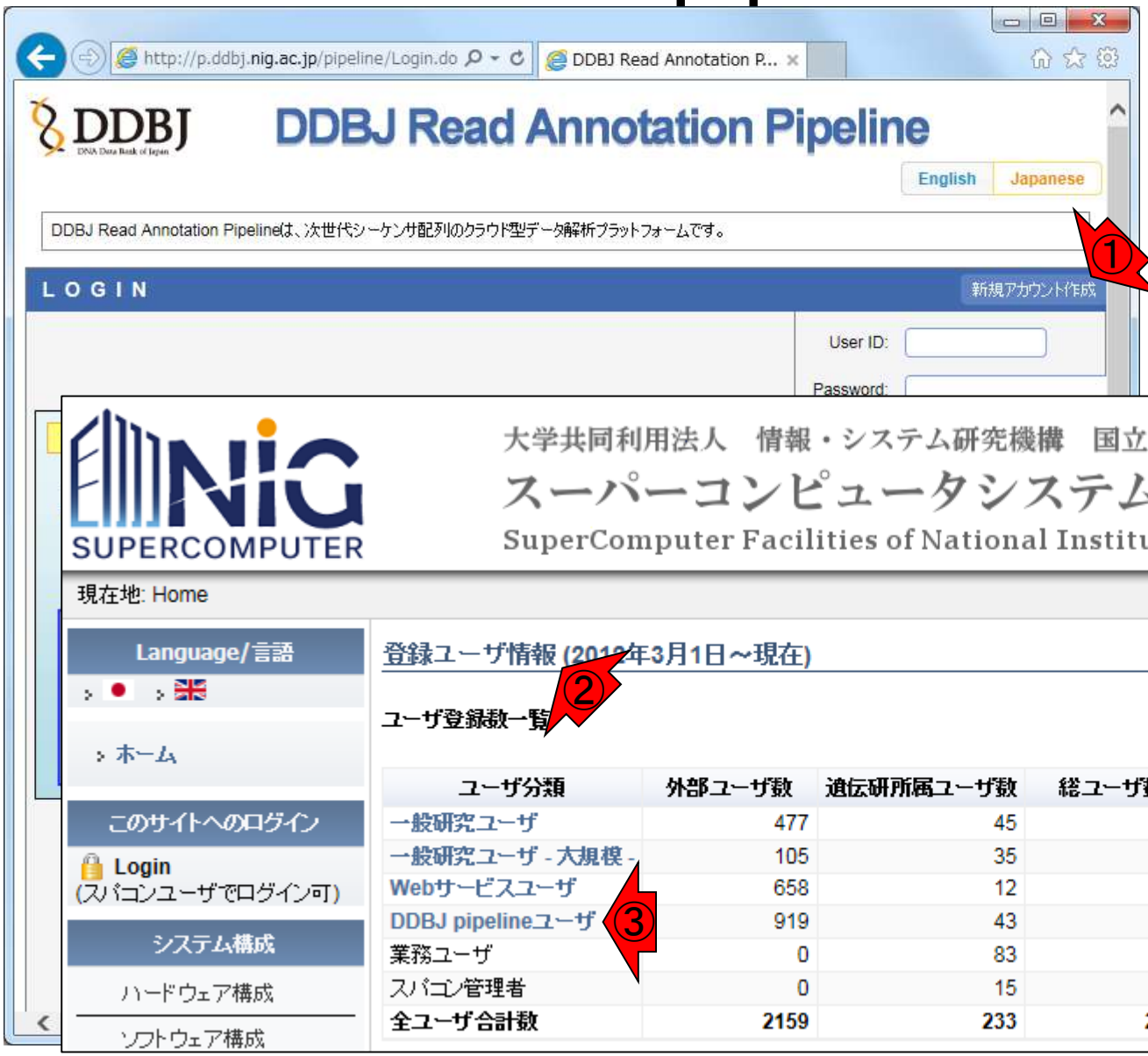
- 手元のファイル[W13-4]
  W5-4で作成した解析したい手元のファイルを共有フォルダ(~/Desktop/mac_share)経由でホストOSにコピー。コピーするファイルは、QC.1.trimmed.fastq.gz(57,061,392 bytes; 約55MB)とQC.2.trimmed.fastq.gz(62,989,289 bytes; 約61MB)です。ここではQC.[0-9].*.gzで2つのファイルを指定しています。

```
cd ~/Documents/DRR024501/result

pwd
ls
cp QC.[0-9].*.gz ~/Desktop/mac_share
```

W13-1：DDBJ pipeline

つまり、①DDBJ Pipelineの新規アカウント作成時の登録情報（氏名・所属・利用目的）は、遺伝研スパコンの②登録ユーザ情報の③のところで公開される。企業の方は、受託解析に使うことはできません。研究開発用途としてお使いください

スライドを見るだけ



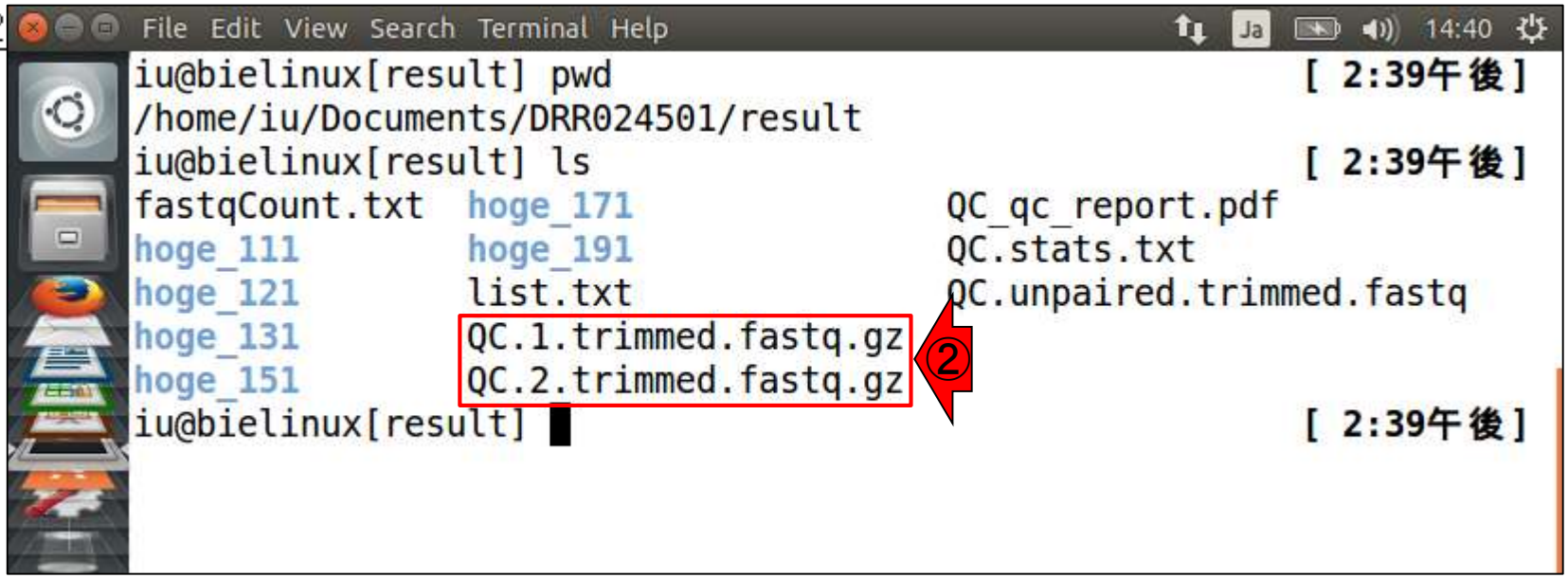
ユーザ分類	外部ユーザ数	遺伝研所属ユーザ数	総ユーザ数
一般研究ユーザ	477	45	522
一般研究ユーザ - 大規模 -	105	35	140
Webサービスユーザ	658	12	670
DDBJ pipelineユーザ	919	43	962
業務ユーザ	0	83	83
スパコン管理者	0	15	15
全ユーザ合計数	**2159**	**233**	**2392**

W13-4：手元のファイル

おさらい。①Bio-Linuxで行ったVelvetの入力ファイルは、②のFaQCs実行結果ファイル。これをDDBJ Pipeline上にFTP経由でアップロードしてVelvetを実行しようとしている

スライドを見るだけ

	(a) フィルタリング前			(b) フィルタリン			
k-mer	コンティグ数	総塩基数	ウェブ資料	コンティグ数	総塩基数		
31	29502	4077679	W10-4				
61	15445	3886574	W10-4				
91	8583	3412266	W10-4				
111	23761	5718204	W10-5				
121	15776	4690144	W10-5	2942	1563384	W12-9	
131	8398	3710829	W10-5	2449	2151400	W12-9	①
151	1306	2599377	W10-5	1306	2599377	W12-9	
171	168	2381523	W10-5	168	2381523	W12-9	
181	198	2386048	W10-1				
191	336	2					



W13-4：ファイル名に注意

連載第2回原稿の①「バイオインフォマティクス分野の常識・非常識」を再掲。非常識なファイル名で実行を試みて「うまくいかないんですけど」的な質問は×

スライドを見るだけ

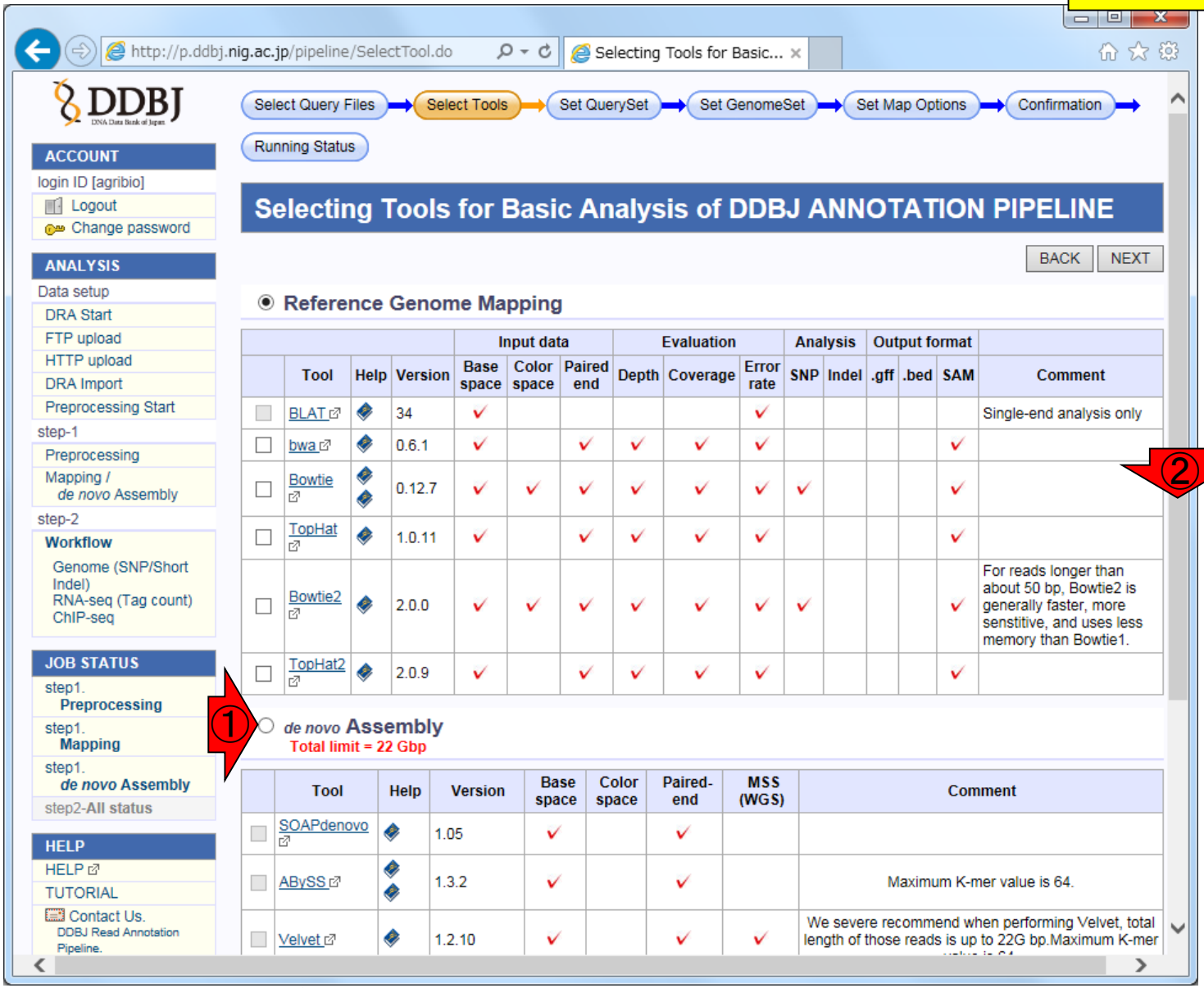
バイオインフォマティクス分野の常識・非常識 ①

コマンドライン環境初心者がよく犯すミスは、適切な場所への半角スペースの入れ忘れである。例えば、「ls -a」と打っているつもりで「ls-a」と打つと、ls と -a の間に半角スペースが適切に挿入されていないため、ls-a というコマンドとして認識される。当然のことながら「そのようなコマンドはない（command not found）」といわれる［ウェブ資料 13］。この程度であれば明確にエラーと認識できるので対処のしようはある。コマンドライン環境では、半角スペースは明確な意味を持つ場合が多い。それゆえ、NGS データ解析に限らず、解析したいファイルを入力として与える際にファイル名の中にスペースを入れるのは、コマンドライン環境中心のバイオインフォマティクスの世界では非常識である。他に「全角文字」や「!, ", #, $ などの英数字以外の文字」も忌避される。一般に多数のファイルが 1 つのディレクトリ内に存在する場合、意味を持たせたファイル名にすることが多い。例えば、group1_rep1.fa, group1_rep2.fa, group2_rep1.fa, group2_rep2.fa のような具合である。この場合、上記のようなブラックリストを眺めるのではなく、使っても大丈夫という経験に基づくホワイトリストを利用するほうが手っ取り早い。例えば、著者らは主に「xxx_yy_zzzz_001.fa」のような英数字とアンダースコア（_）の組合せを利用する。

ファイルの拡張子にも気をつけたほうがよい。上記 FASTA ファイルの拡張子は .fa であったが、.fasta でもよい。拡張子が .fa か .fasta であれば、常識的な範囲で他はなんでもよいという意味合いで＊.fa や＊.fasta という表現もなされる。塩基ごとのクオリティ情報を含む FASTQ 形式ファイル（以下、FASTQ ファイル）の拡張子は、＊.fq または＊.fastq が一般的である。もちろんそれ以外の拡張子でも受け入れてくれる NGS 解析用プログラムは存在するかもしれない。例えば、オプションで FASTQ 形式だということを宣言さえしておけば、実際の拡張子は .txt でも .doc でも .pdf でも受け入れてくれるかもしれない。しかし著者らは、＊.doc や＊.pdf での動作確認（ブラックリスト作成）には関与しない。無難な＊.fq や＊.fastq を素直に利用する。

W15-2：Select Tools

デフォルトはマッピングになっているので、①de novo Assemblyにチェックを入れて、②ページ下部に移動



http://p.ddbj.nig.ac.jp/pipeline/SelectTool.do — Selecting Tools for Basic...

DDBJ

Select Query Files → Select Tools → Set QuerySet → Set GenomeSet → Set Map Options → Confirmation → Running Status

ACCOUNT
login ID [agribio]
Logout
Change password

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

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

HELP
HELP
TUTORIAL
Contact Us.
DDBJ Read Annotation Pipeline.

Selecting Tools for Basic Analysis of DDBJ ANNOTATION PIPELINE

BACK NEXT

Reference Genome Mapping

	Tool	Help	Version	Input data: Base space	Input data: Color space	Input data: Paired end	Evaluation: Depth	Evaluation: Coverage	Evaluation: Error rate	Analysis: SNP	Analysis: Indel	Output format: .gff	Output format: .bed	Output format: SAM	Comment
☐	BLAT		34	✓					✓						Single-end analysis only
☐	bwa		0.6.1	✓		✓	✓	✓	✓					✓	
☐	Bowtie		0.12.7	✓	✓	✓	✓	✓	✓	✓				✓	
☐	TopHat		1.0.11	✓		✓	✓	✓	✓					✓	
☐	Bowtie2		2.0.0	✓	✓	✓	✓	✓	✓	✓				✓	For reads longer than about 50 bp, Bowtie2 is generally faster, more senstitive, and uses less memory than Bowtie1.
☐	TopHat2		2.0.9	✓		✓	✓	✓	✓					✓	

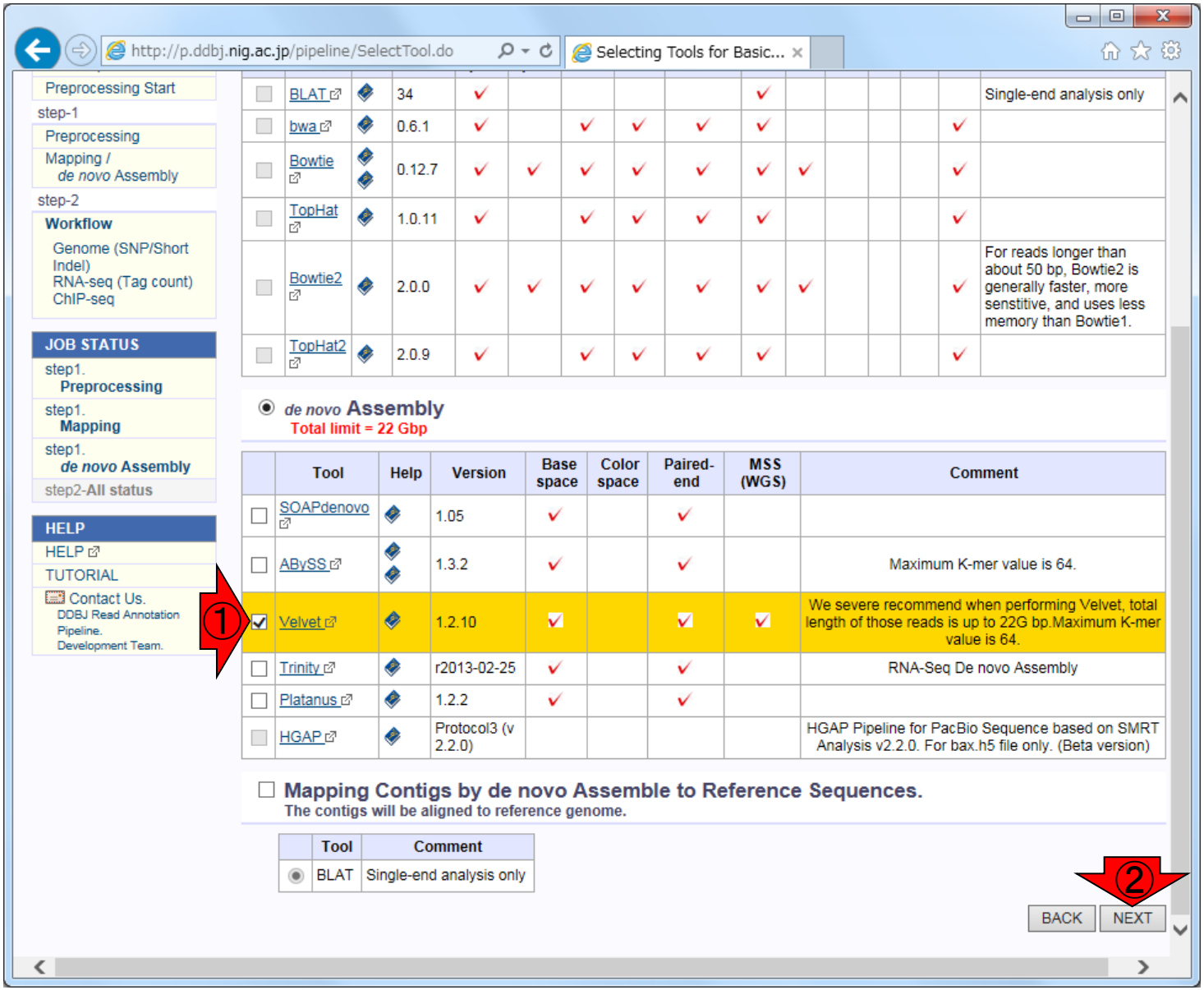
① ○ ***de novo* Assembly**
Total limit = 22 Gbp

	Tool	Help	Version	Base space	Color space	Paired-end	MSS (WGS)	Comment
☐	SOAPdenovo		1.05	✓		✓		
☐	ABySS		1.3.2	✓		✓		Maximum K-mer value is 64.
☐	Velvet		1.2.10	✓		✓	✓	We severe recommend when performing Velvet, total length of those reads is up to 22G bp.Maximum K-mer

② ←

①Velvetにチェックをいれて、②NEXT

W15-2：Select Tools



W15-3：Set QuerySet

①解析したいデータにチェックをいれて、②Set as Pair-End。③NEXT。本来この画面は、複数のクエリファイルを使用する場合に、それらを連結して1つのクエリセットとしてジョブを実行するか、別々のクエリセットとして並列して実行するかを指定する画面である。今回はクエリファイル1つを単独で使用するので、この画面にはあまり意味がない

灰色は読まない

W15-4：Set Ass.Options

W7-3のBio-Linux上での実行経験から、DDBJ Pipelineでは、①k=23がデフォルトなのだろう。W7-8のvelvetg実行時は特にオプションを指定しなかったが、②で示されているようなオプションもあるのだと学ぶ。③長さによる配列のフィルタリング(④デフォルトは100 bp)もやってくれるようだ

W15-4：Set Ass.Options

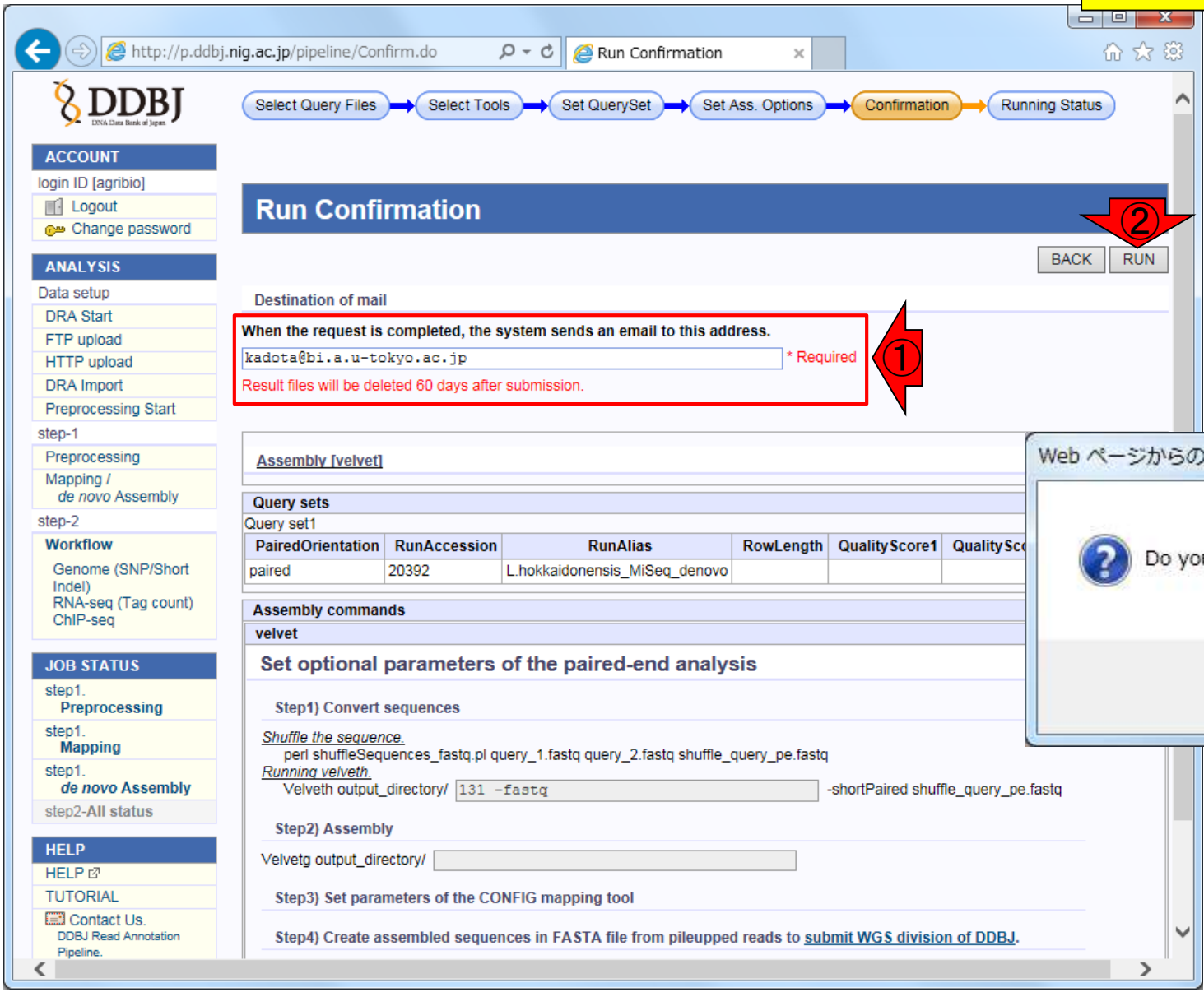
W12-10(スライド72)を眺め、とりあえず
①k=131でBio-Linux上で指定したオプションと同じにして(②～④)実行。⑤NEXT

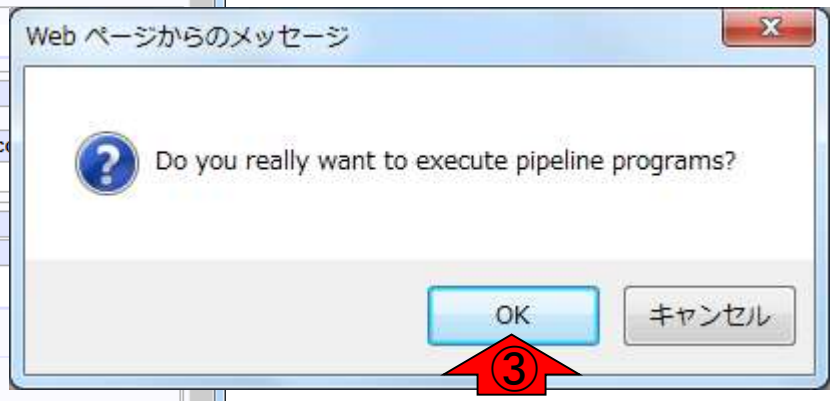
W15-4：Set Ass.Options

①オプションの見栄えはプログラムごとに異なる。これは②Velvetの場合

W15-5 : Confirmation

①アセンブリが終了したら、アカウント作成時に指定したアドレス宛にメールが送られる。②RUN。③OK





W16-1：計算終了

①DDBJ Pipelineから計算終了メールが届いたら、②DDBJ Pipelineに再度ログインして計算結果を眺める

2016/01/15 (金) 20:37

pipeline_team@g.nig.ac.jp

Job finished : DDBJ Read Annotation Pipeline ①

宛先 kadota@bi.a.u-tokyo.ac.jp

ＣＣ pipeline_report@g.nig.ac.jp

このメッセージから余分な改行を削除しました。

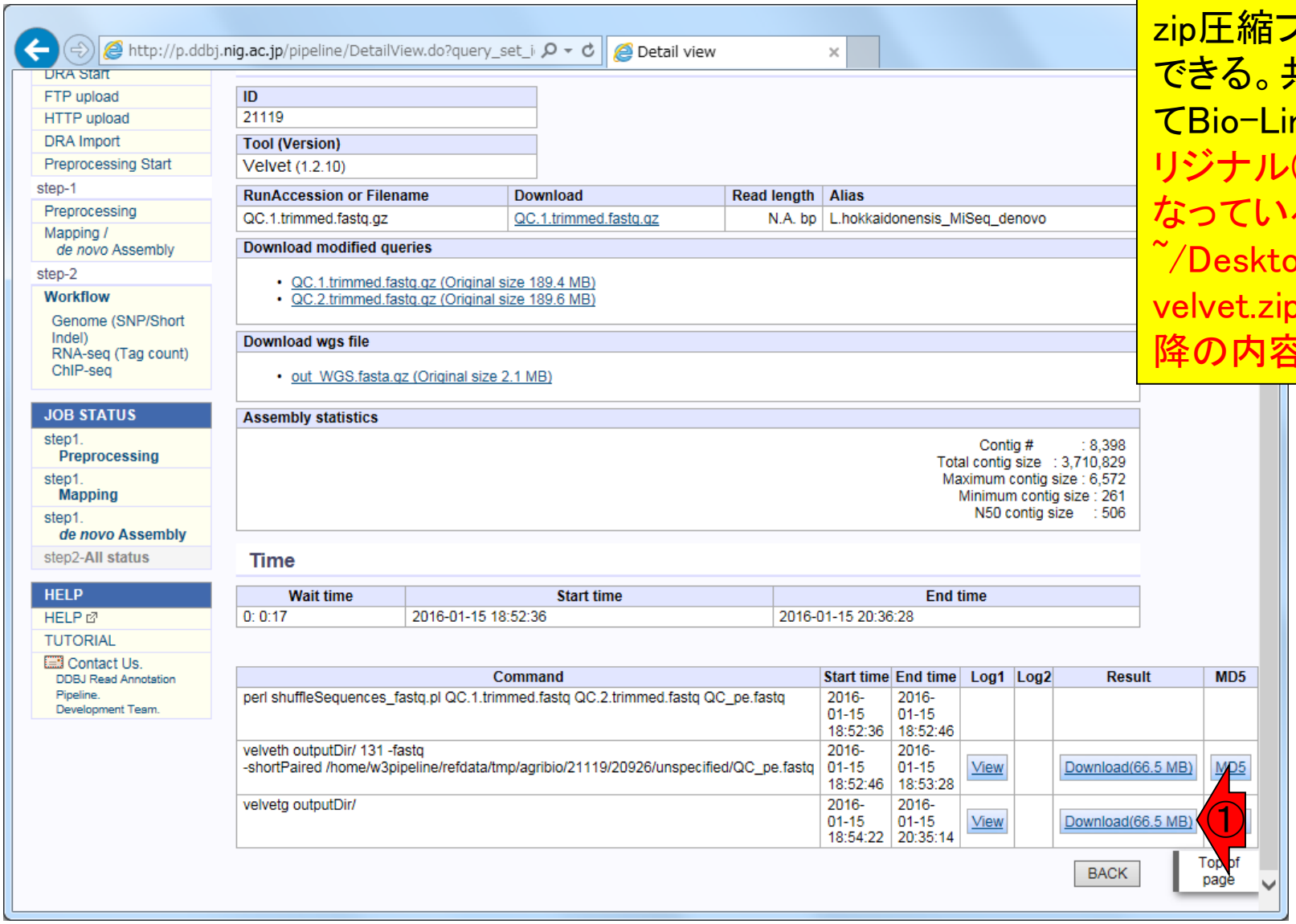
Dear agribio,

Your request to DDBJ pipeline service has finished.
Please visit the web site to obtain analytical results.
Request ID: 21119
URL: https://p.ddbj.nig.ac.jp/ ②

If you have troubles in this service, please write to pipeline_dev@ddbj.nig.ac.jp
Thank you for trying our analytical service.

Regards,
DDBJ

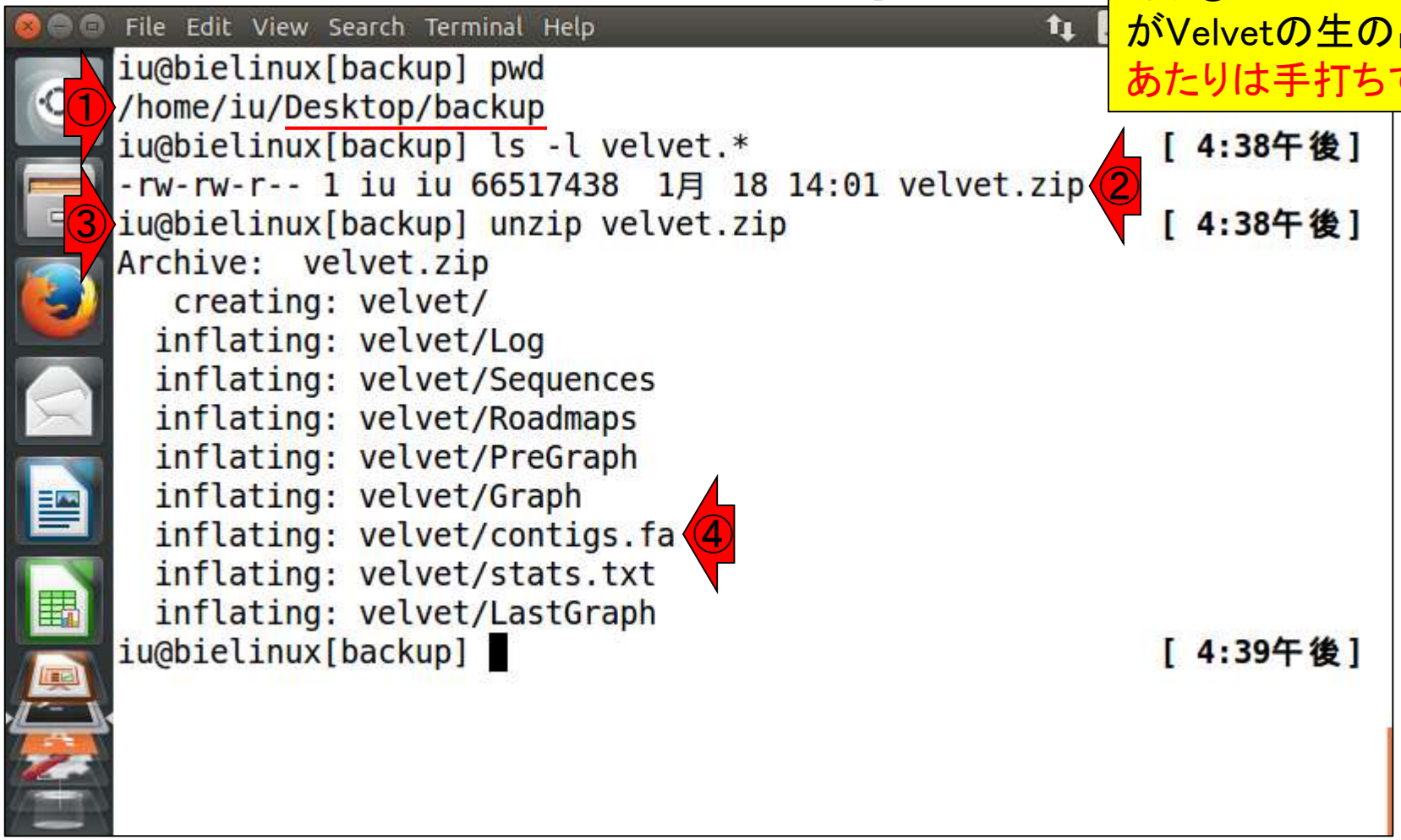
W18-1：フィルタリング前の結果



ページ下部に移動した後の状態。①「Download(66.5MB)」をクリックすると、velvet.zipというzip圧縮ファイルをダウンロードできる。共有フォルダに保存してBio-Linux上で眺める。とオリジナル(第6回ウェブ資料)はなっているが、講習会では、~/Desktop/backup上にあるvelvet.zipを取り扱うので、以降の内容は若干異なる

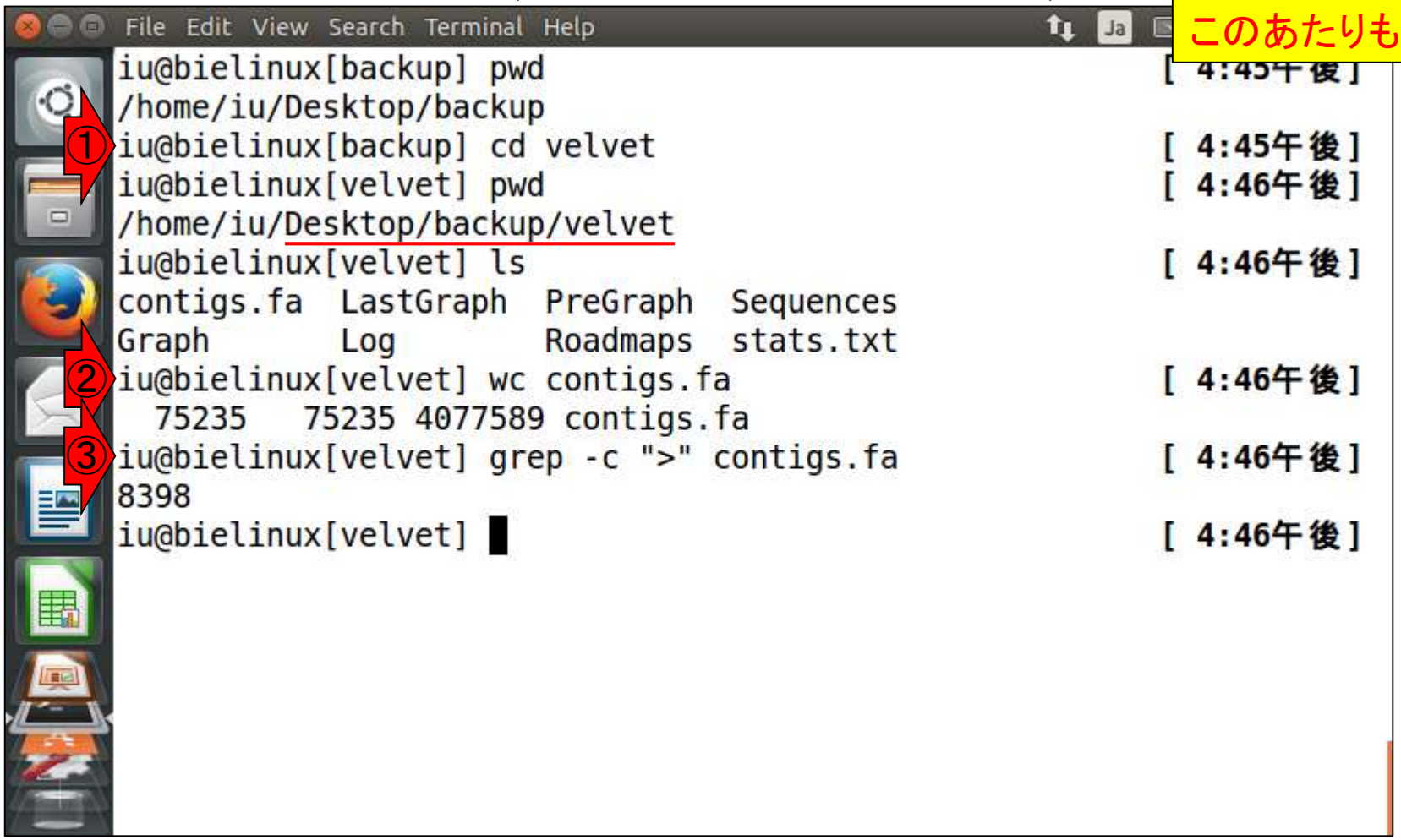
W18-2：Bio-Linuxで解凍

①~/Desktop/backupにある、②velvet.zipの存在確認。③unzipで解凍。④velvetフォルダ中のcontigs.faがVelvetの生の出力ファイル。このあたりは手打ちでやってください



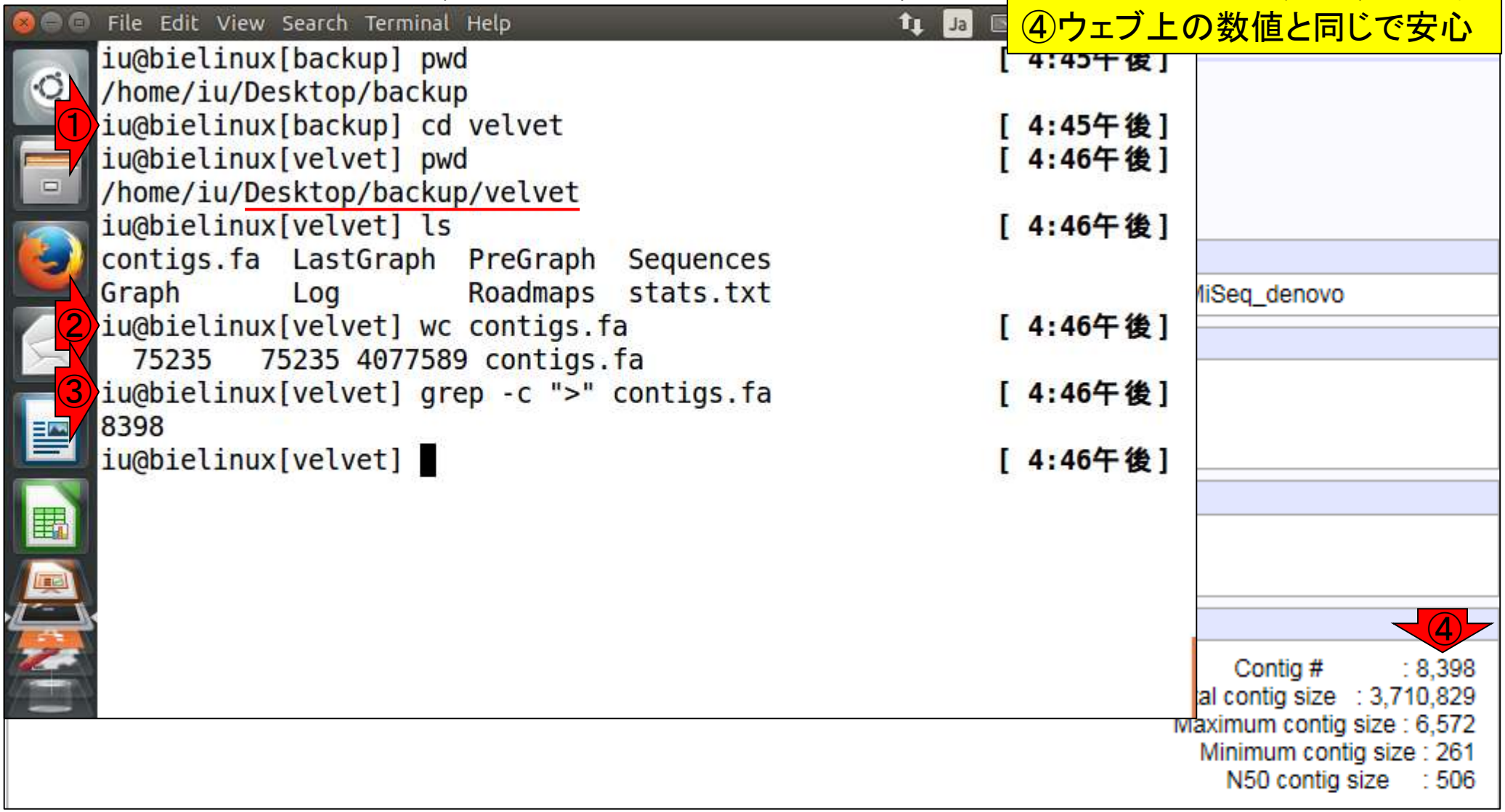
W18-3：行数とコンティグ数

①velvetディレクトリに移動。
contigs.faの②行数は75,235、③配列（コンティグ）数は8,398個。
このあたりも手打ち



W18-3：行数とコンティグ数

①velvetディレクトリに移動。
contigs.faの②行数は75,235、③配列（コンティグ）数は8,398個。
④ウェブ上の数値と同じで安心



W18-4：フィルタリング

DDBJ Pipeline実行結果ファイルを入力として、①指定した配列長閾値**未満**の配列を取り除くPythonプログラムfastaLengthFilter.pyを実行。300 bp**以上**の配列は②2,449個となり、Bio-Linux上で実行した結果と同じ(W12-9)

```
iu@bielinux[~/Desktop/backup/velvet]
iu@bielinux[backup] pwd
/home/iu/Desktop/backup
iu@bielinux[backup] cd velvet                                    [ 4:45午後]
iu@bielinux[velvet] pwd                                          [ 4:46午後]
/home/iu/Desktop/backup/velvet
iu@bielinux[velvet] ls                                           [ 4:46午後]
contigs.fa  LastGraph  PreGraph  Sequences
Graph       Log        Roadmaps  stats.txt
iu@bielinux[velvet] wc contigs.fa                                [ 4:46午後]
  75235   75235 4077589 contigs.fa
iu@bielinux[velvet] grep -c ">" contigs.fa                       [ 4:46午後]
8398
iu@bielinux[velvet] fastaLengthFilter.py contigs.fa 300 > hoge.fa
iu@bielinux[velvet] grep -c ">" hoge.fa                          [ 5:01午後]
2449
iu@bielinux[velvet]                                              [ 5:02午後]
```

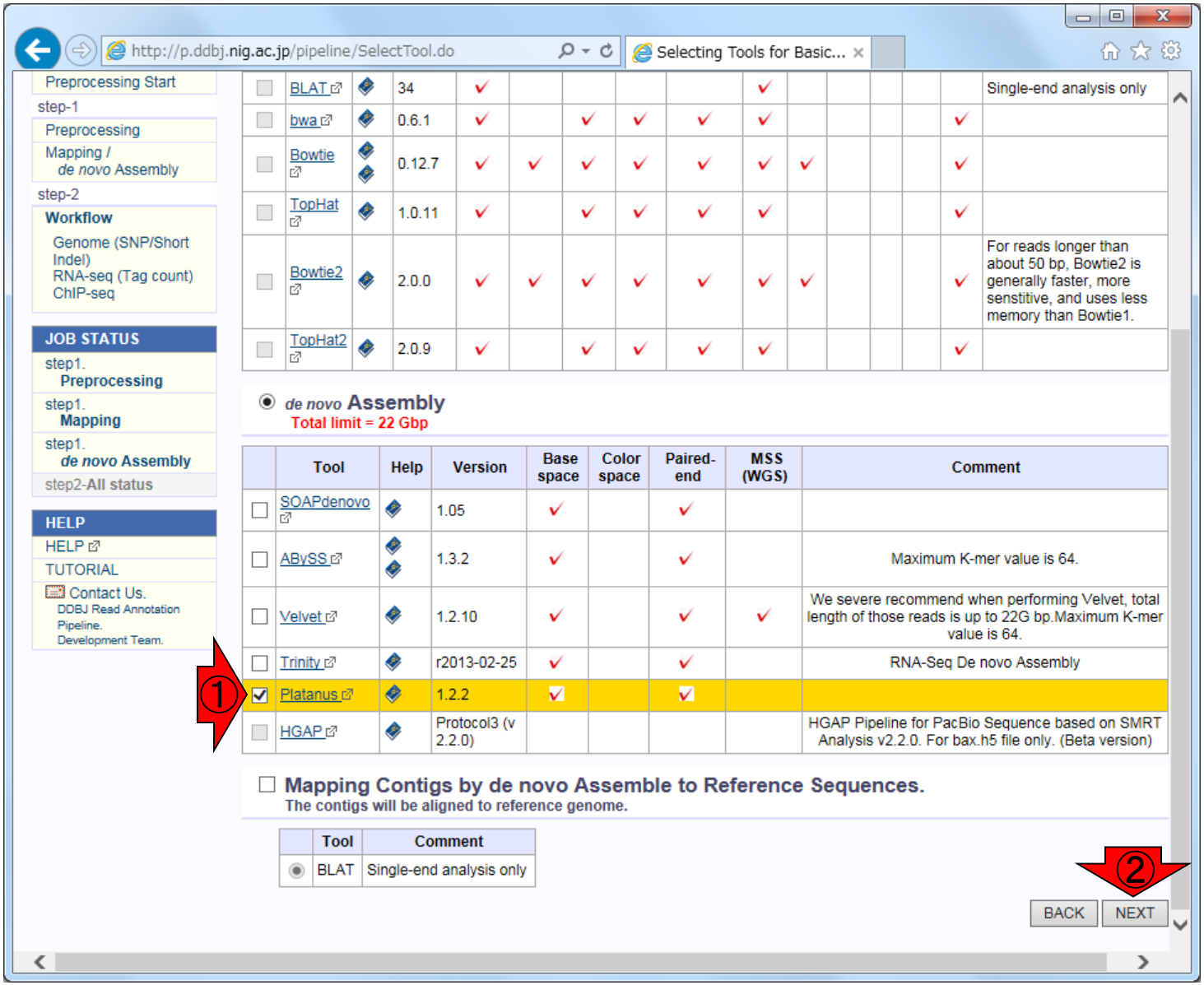
① ②

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

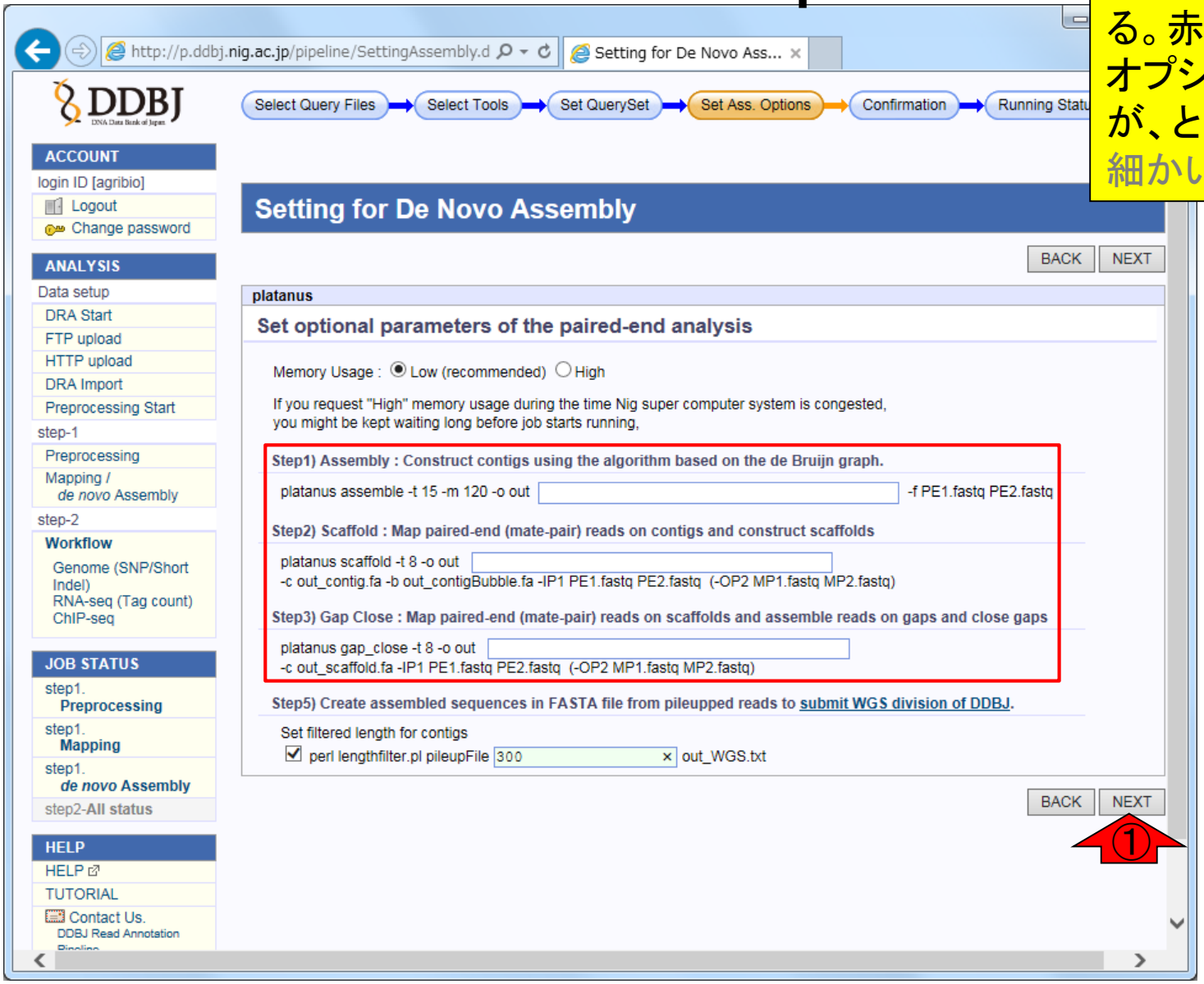
W19-2：Select Tools

DDBJ Pipelineでは、①Platanus（ver. 1.2.2）も実行可能。チェックを入れて、②NEXT



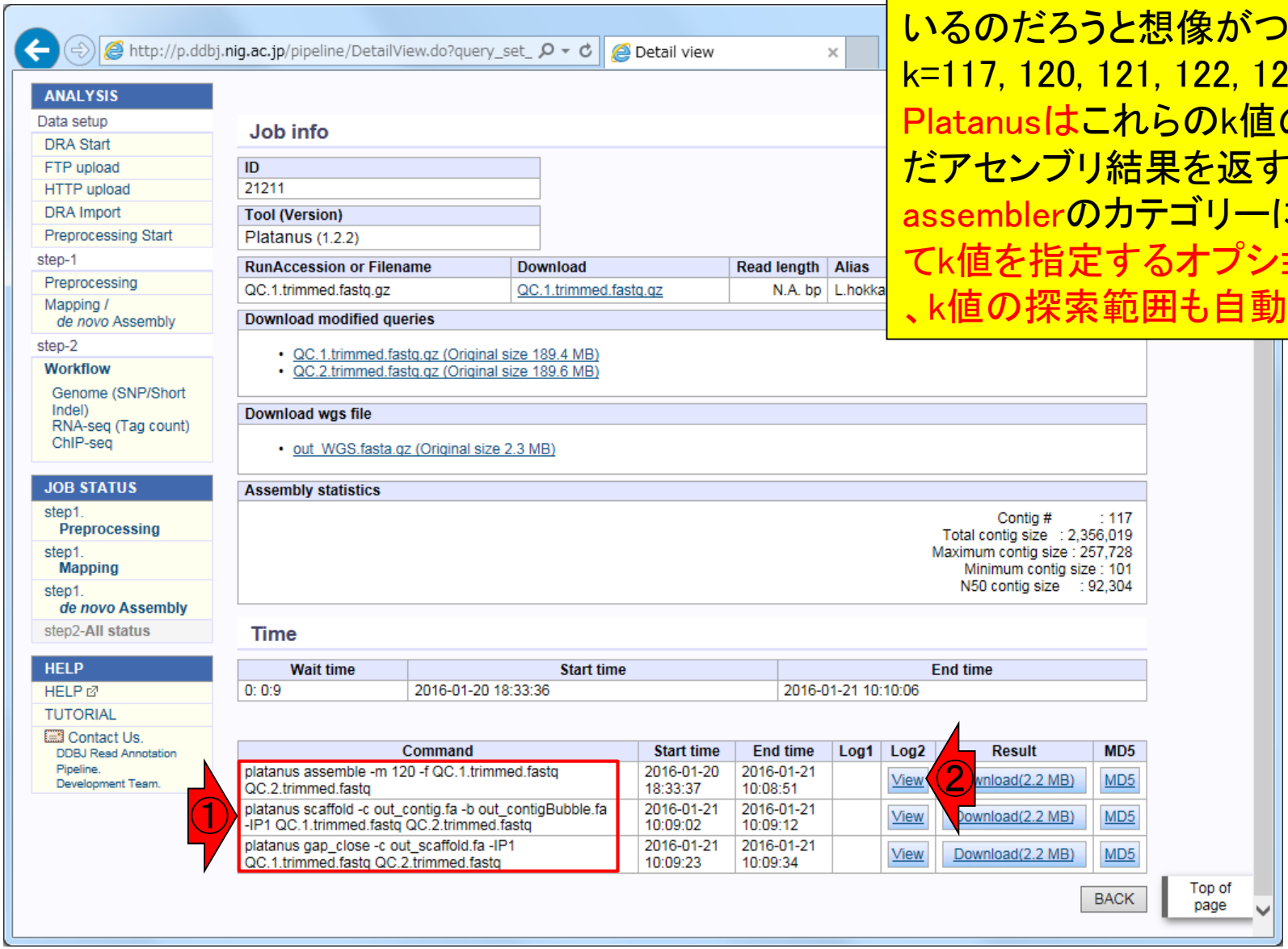
W19-4：Set Ass. Options

乳酸菌ゲノム決定論文中では、「Platanus assembler ver 1.2 with **the default settings**」と書かれている。赤枠のStep1, 2, 3の計3か所にオプションを指定する箇所があるが、とりあえずここは空白として…
細かい点はすっ飛ばして①NEXT



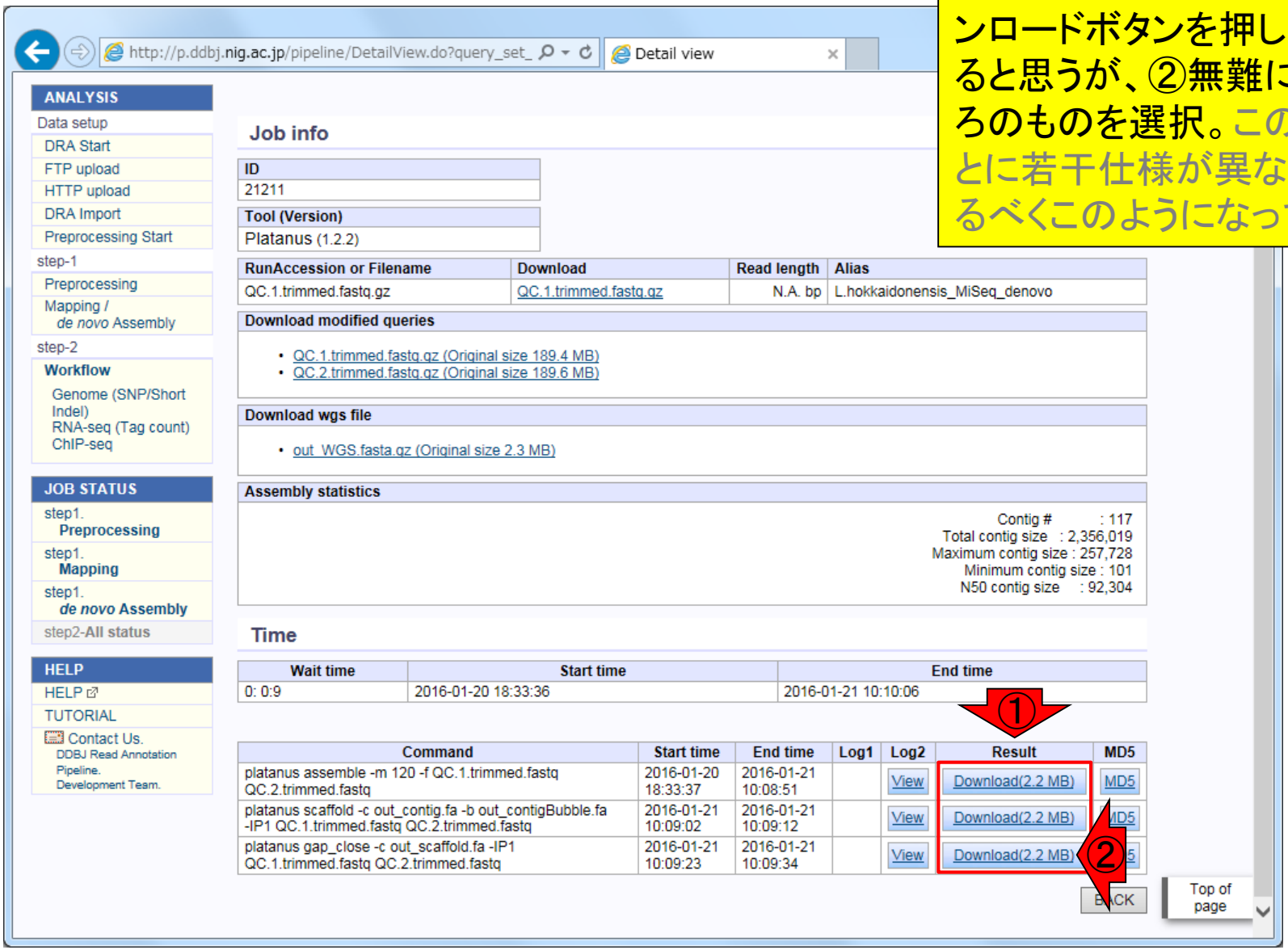
W20-2：結果を眺める

①Platanusは、主に3ステップからなる(W19-4)が、そのコマンドの実体がわかる。②実行ログを眺めると、k=32, 42, 52などをいろいろ調べているのだろうと想像がつく。最後のほうが、k=117, 120, 121, 122, 123で終わっている。Platanusはこれらのk値の結果を全て取り込んだアセンブリ結果を返すmulti-k genome assemblerのカテゴリーに属する。Velvetと違ってk値を指定するオプションが存在しないのは、k値の探索範囲も自動的に決められるから



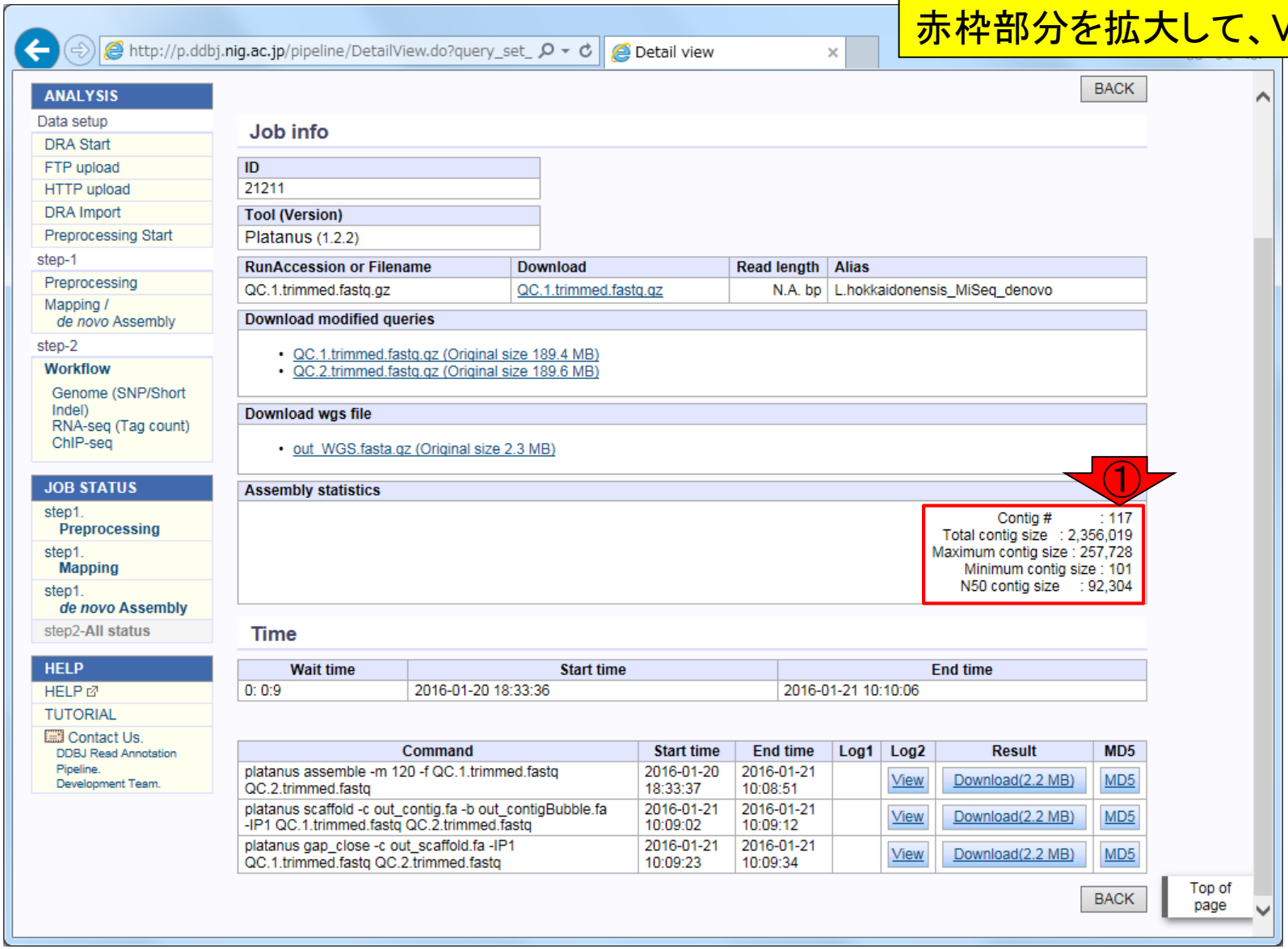
W20-2：結果を眺める

①フィルタリング前のPlatanus実行結果ファイルは、ここからダウンロードできる。（*de novo*アセンブリの場合は）おそらくどのダウンロードボタンを押しても同じものが得られると思うが、②無難に最後のステップのところのものを選択。このあたりはプログラムごとに若干仕様が異なるため、統一的に見せるべくこのようになっているのだろう



W20-2：結果を眺める

①Platanus実行結果の基本情報はここからみられる。パッと見、フィルタリング前の段階ですでにコンティグ数も明らかに少なくよさそう。赤枠部分を拡大して、Velvetの結果と比較



W20-3：Velvetと比較

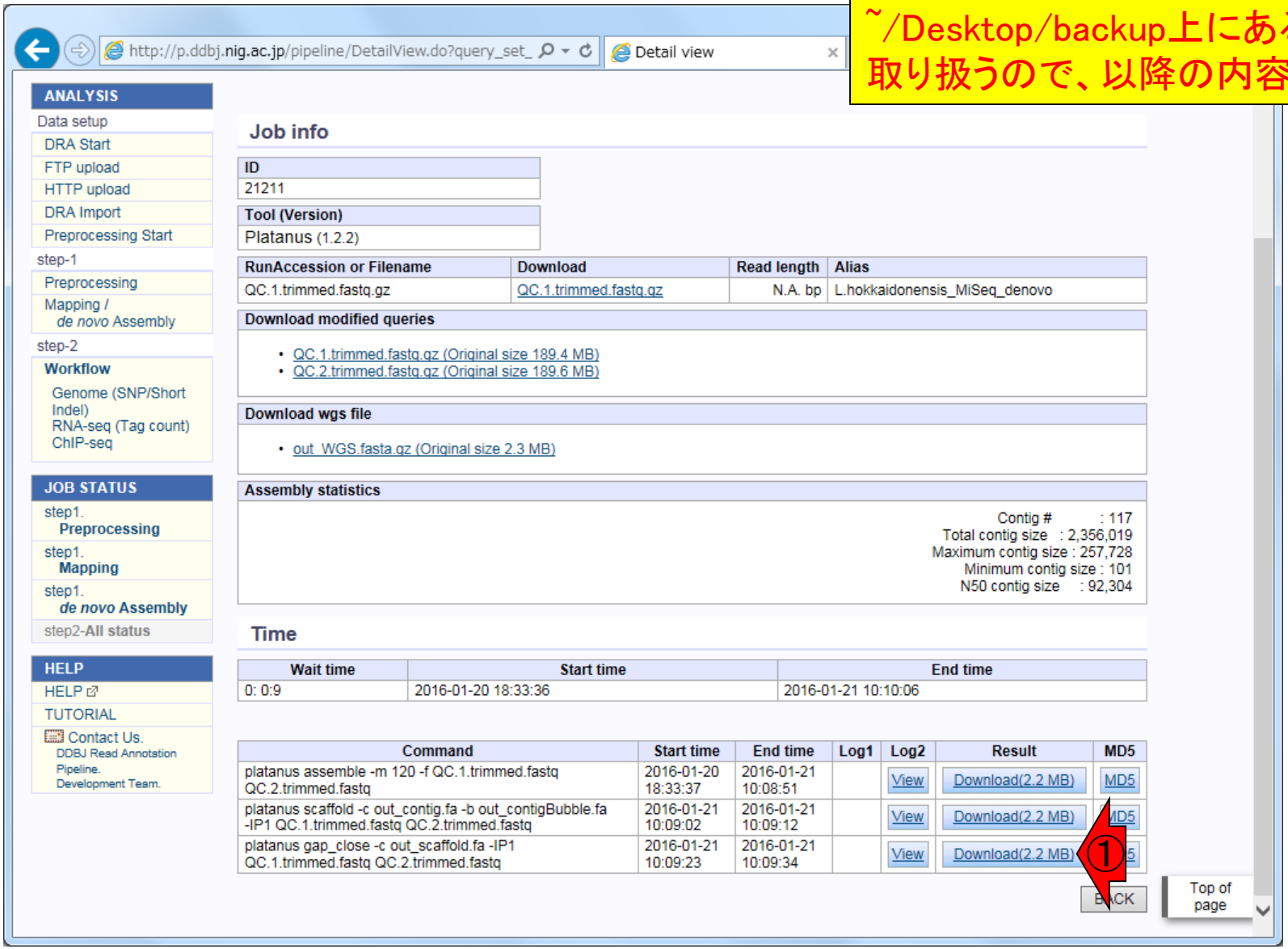
	(a) フィルタリング前			(b) フ		
k-mer	コンティグ数	総塩基数	ウェブ資料	コンティグ数		
31	29502	4077679	W10-4			
61	15445	3886574	W10-4			
91	8583	3412266	W10-4			
111	23761	5718204	W10-5			
121	15776	4690144	W10-5	2942		
131	8398	3710829	W10-5	2449	2151400	W12-9
151	1306	2599377	W10-5	1306	2599377	W12-9
171	168 ①	2381523	W10-5	168	2381523	W12-9
181	198	2386048	W10-1			
191	336	2405431	W10-5	336	2405431	W12-9

左表で調べた範囲のVelvet実行結果(W12-10)よりも、特にk値を意識すらせずに実行したPlatanusの結果のほうが明らかに優れていることがわかる。具体的には、Velvetでの最少配列数は①168個(k=171)であったが、②Platanusはフィルタリング前の状態で117個。そして、③Minimum contig sizeが101 bpであることから、300 bp未満でフィルタリングを行うと、間違いなく配列数が減ると予想する

② Contig # : 117
Total contig size : 2,356,019
Maximum contig size : 257,728
③ Minimum contig size : 101
N50 contig size : 92,304

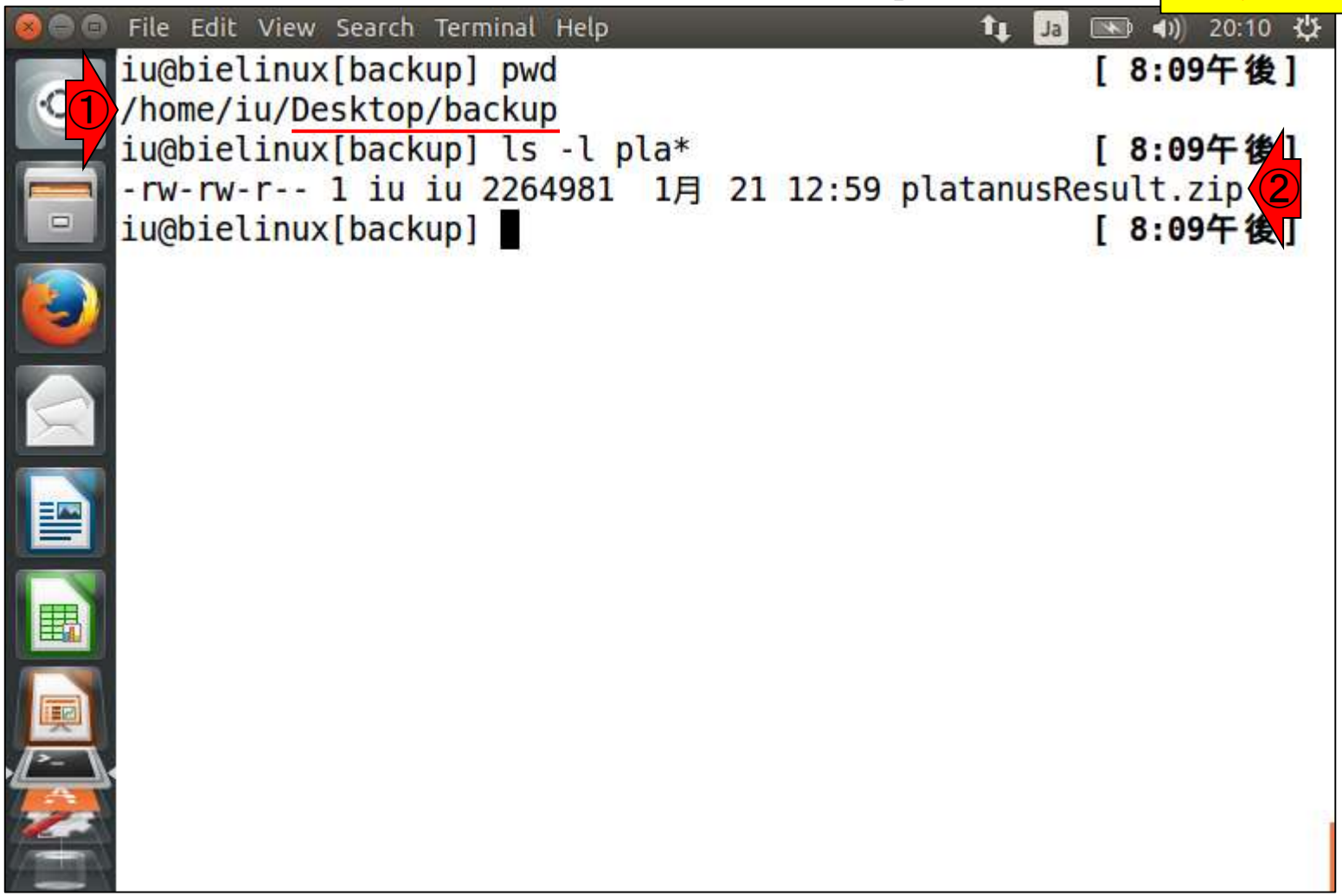
W20-4：ダウンロード

①Platanus実行結果ファイル(platanusResult.zip)を、共有フォルダにダウンロード。とオリジナル(第6回ウェブ資料)はなっているが、講習会では、~/Desktop/backup上にあるplatanusResult.zipを取り扱うので、以降の内容は若干異なる



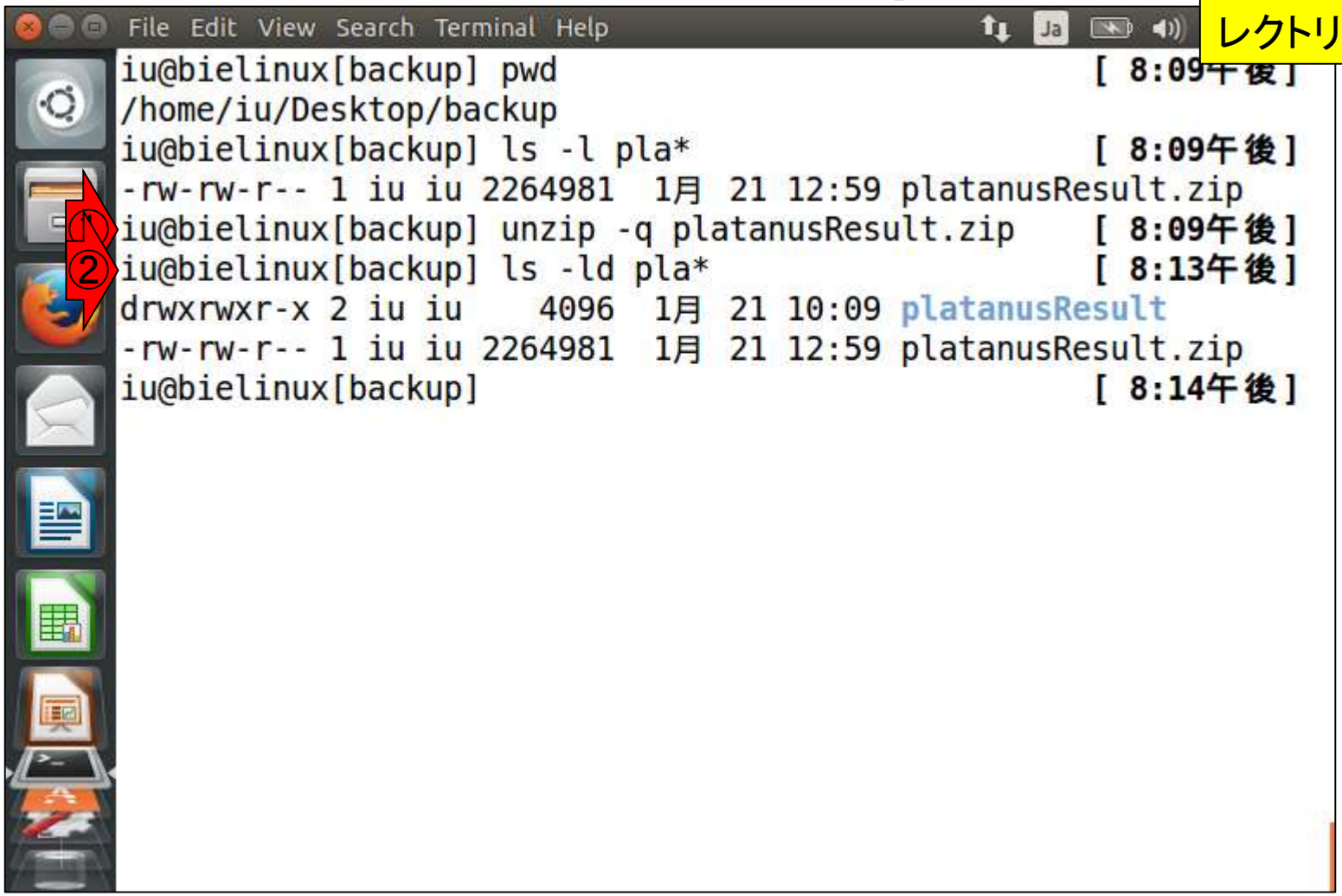
W20-5：Bio-Linuxで解凍

①~/Desktop/backupにある、②platanusResult.zipの存在確認。このあたりは手打ちでやってください



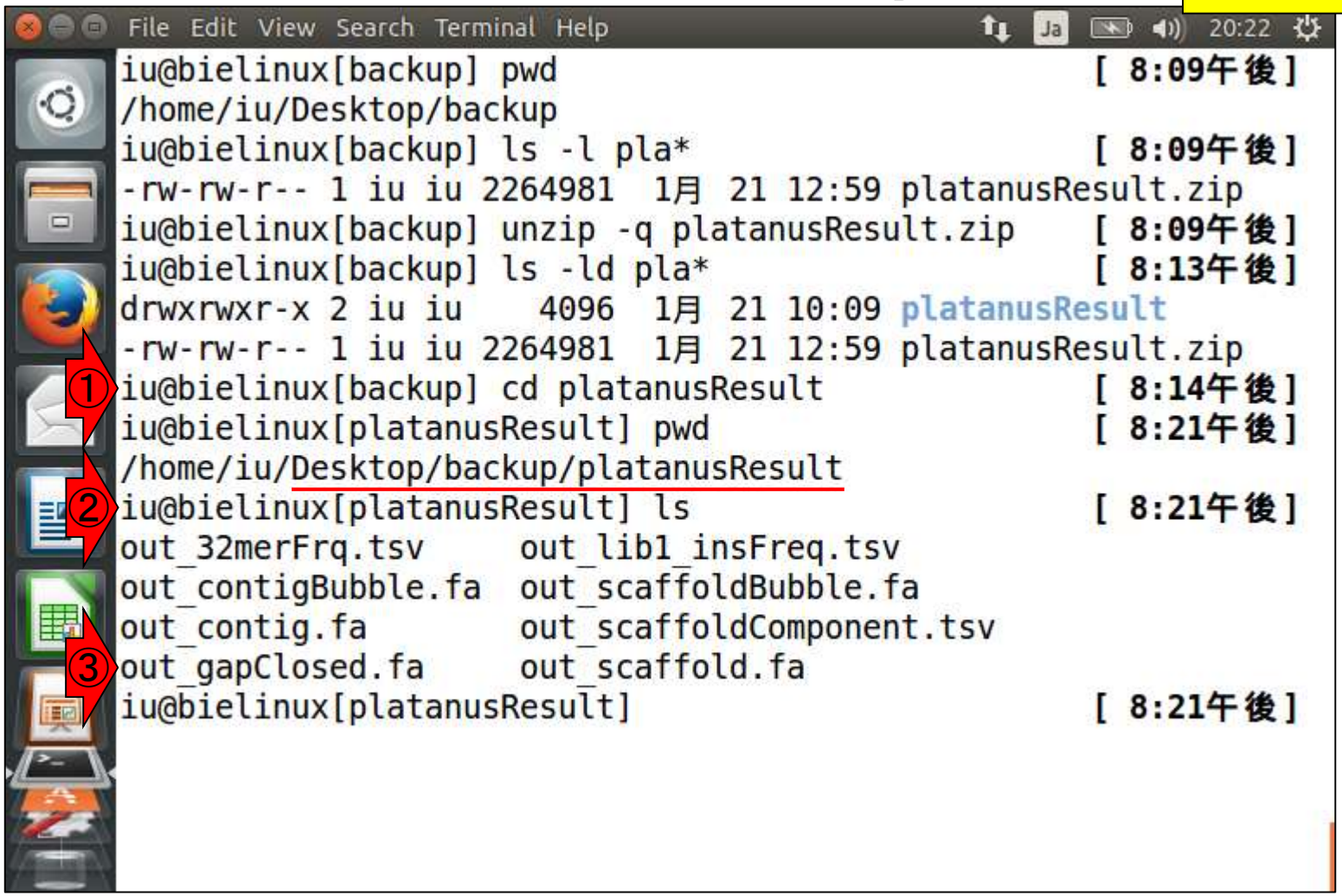
W20-5：Bio-Linuxで解凍

①unzipコマンドで解凍。-qをつけてquietで解凍。②lsで確認。dオプションをつけることで、ディレクトリの中身を表示させない



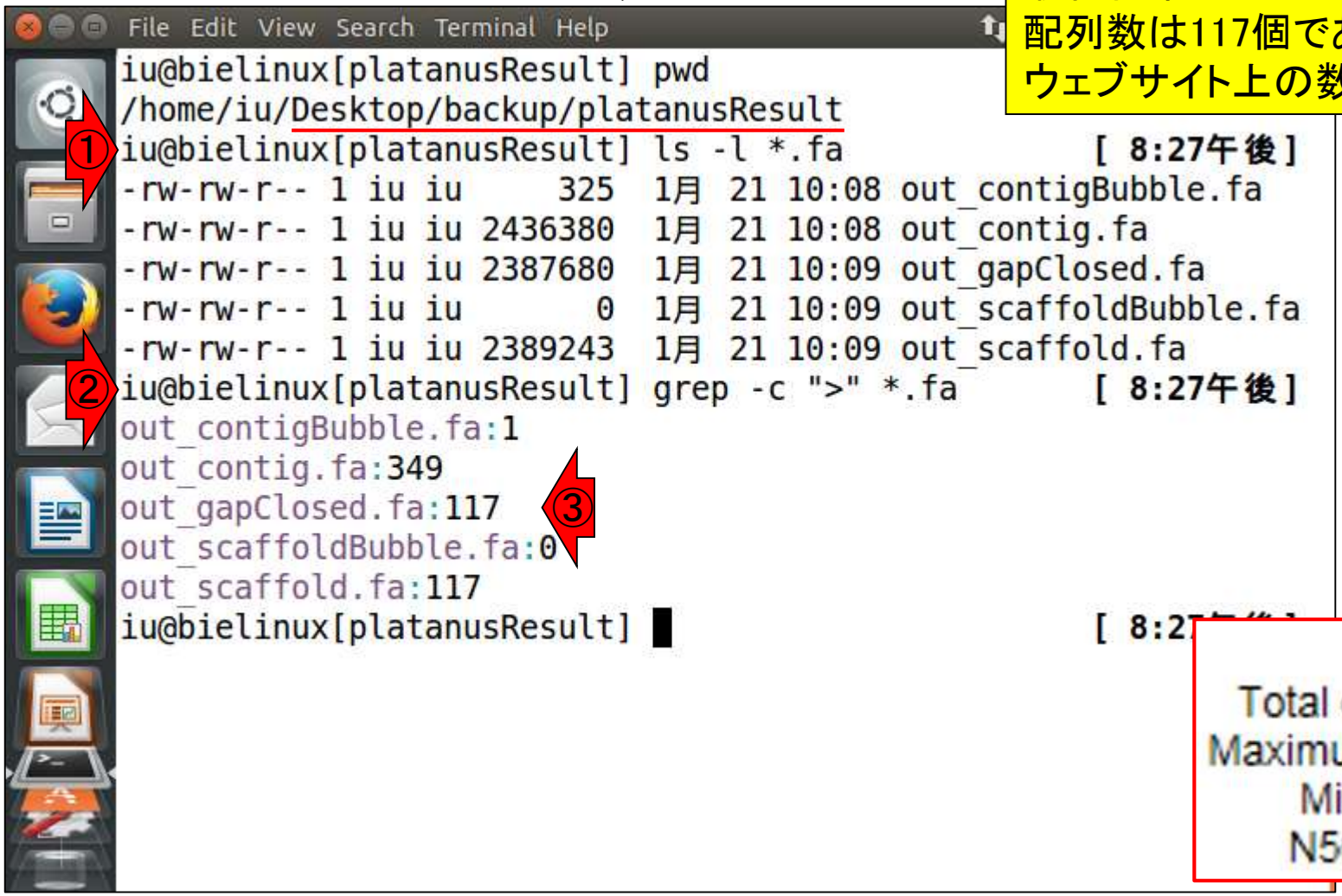
W20-5：Bio-Linuxで解凍

①platanusResultディレクトリに移動し、②ls。Platanusの最終結果ファイルは③out_gapClosed.fa



W20-6：配列数確認

①拡張子が.faのファイルを表示。②*.faの配列数を一気に出力。③確かにPlatanusの最終結果ファイルであるout_gapClosed.faの配列数は117個であり、④DDBJ Pipelineのウェブサイト上の数値と一致する

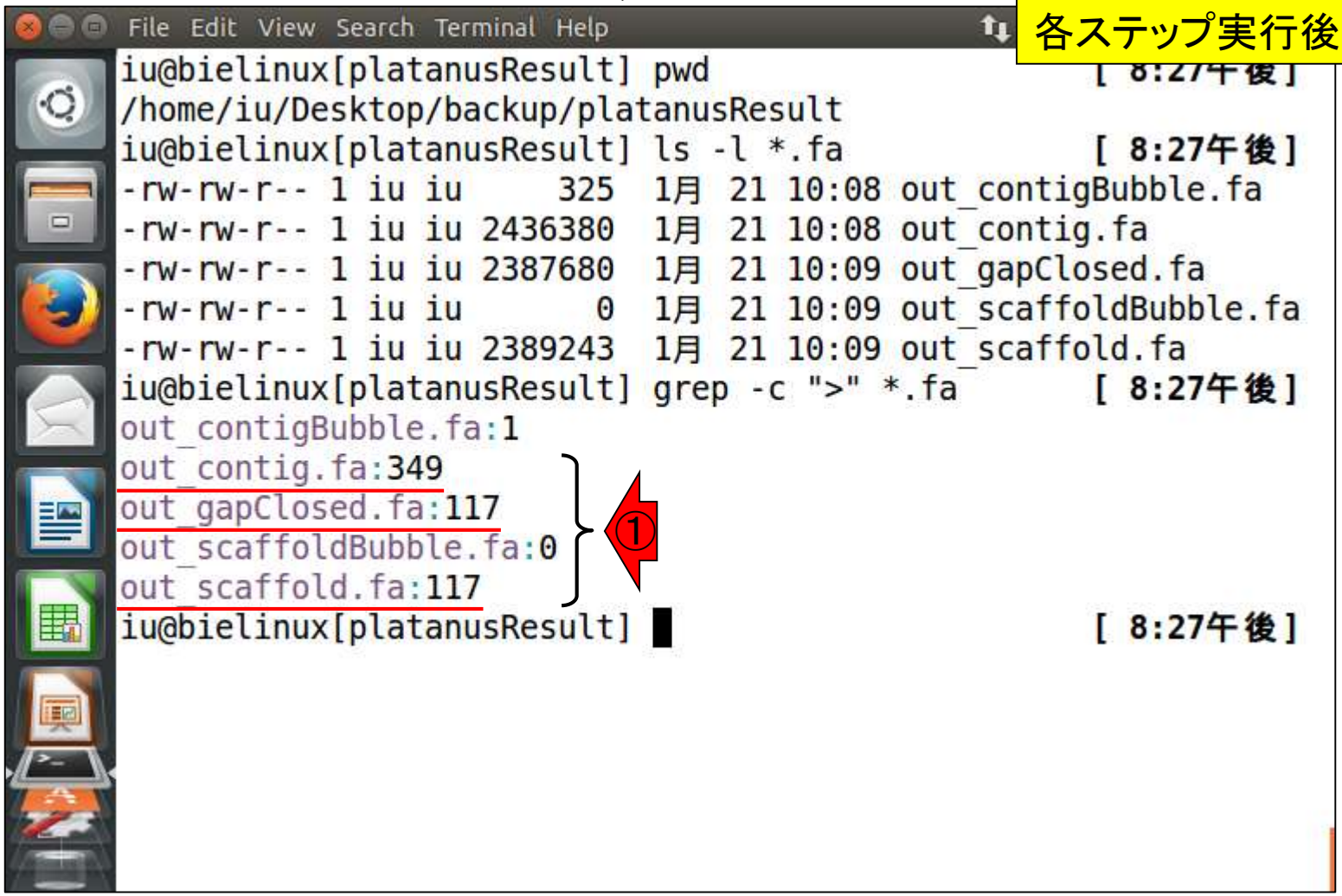


④

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

W20-6：配列数確認

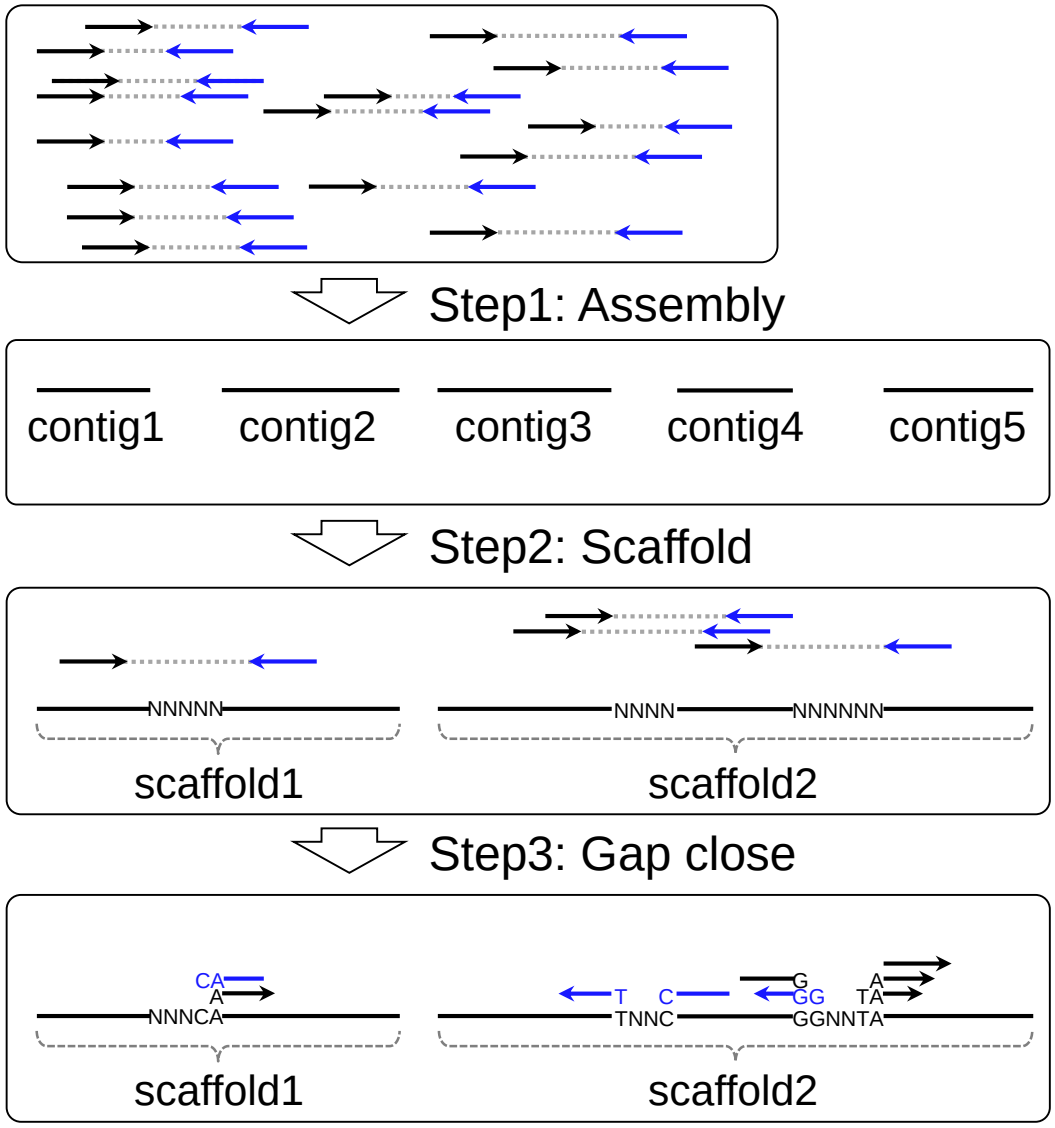
①赤下線の配列数は、2016.07.20のスライド50と同じです。この3つは、Platanusによる3ステップからなる*de novo*アセンブリの、各ステップ実行後の出力ファイルでした



おさらい：*de novo*アセンブリの手順は大まかに3つのステップからなる

*de novo*アセンブリ

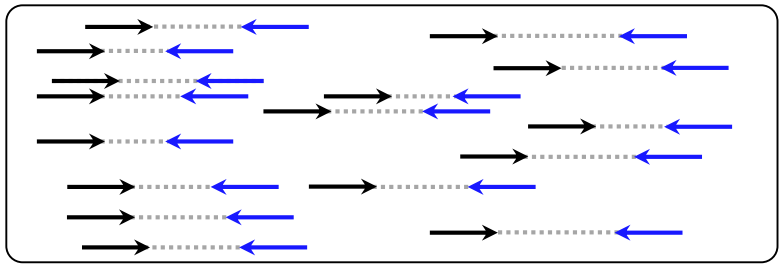
入力：paired-end FASTQファイル



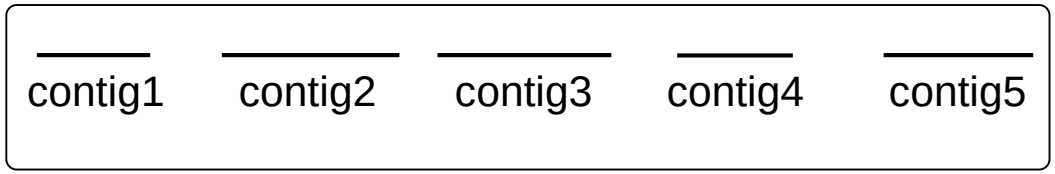
de novoアセンブリ

①②③の3つは、各ステップ実行後のファイルだったことを思い出そう。配列数の変遷が妥当である、という議論を2016.07.20にしました

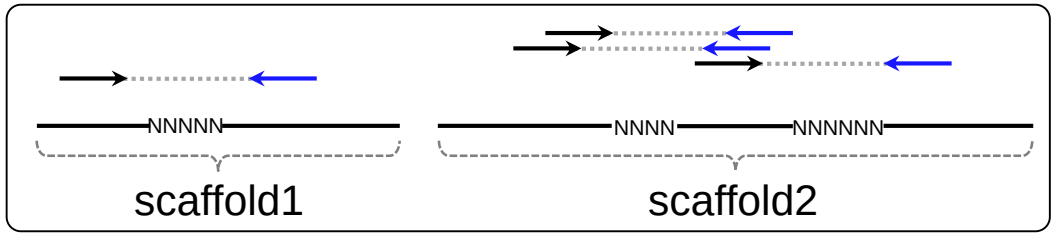
入力：paired-end FASTQファイル



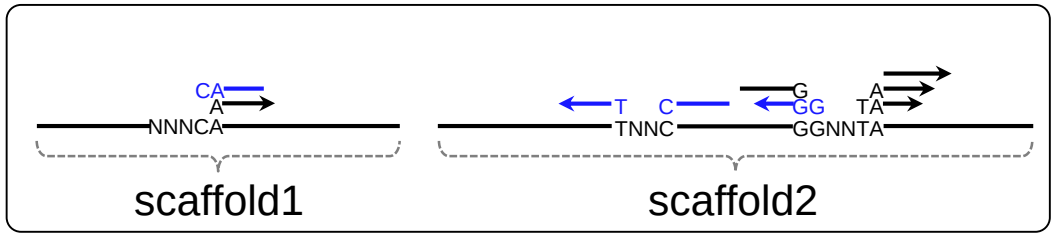
Step1: Assembly → out_contig.fa (349配列) ①



Step2: Scaffold → out_scaffold.fa (117配列) ②



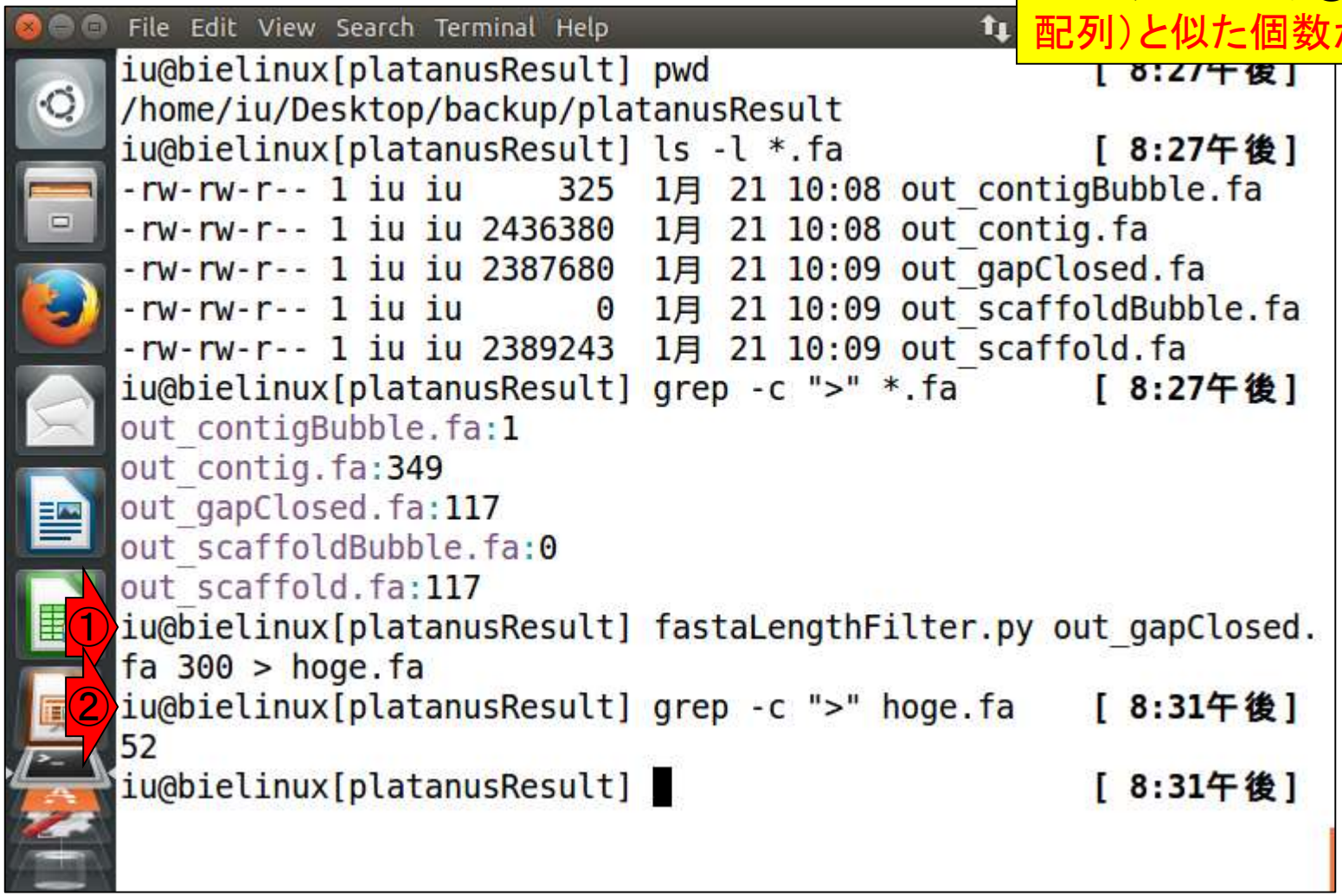
Step3: Gap close → out_gapClosed.fa (117配列) ③



参考

W20-7：フィルタリング

out_gapClosed.faを入力として、①300 bp未満の配列を除去するプログラムを実行。②配列数は52個。③原著論文中の結果（53配列）と似た個数が得られたことがわかる

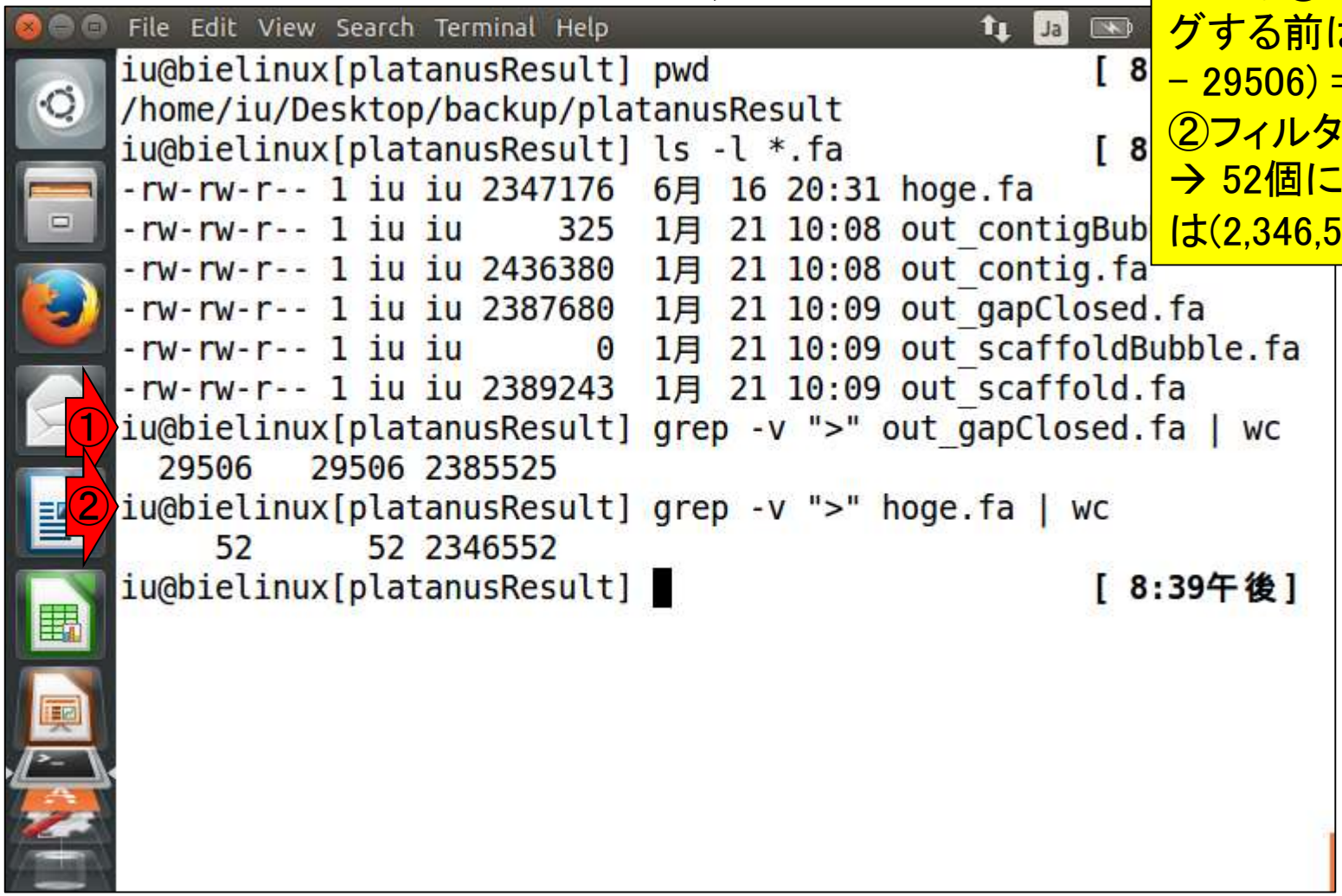


③

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

参考

W20-7：総塩基数



「grep -v “>”」は、”>”を含まない行を出力する(W8-2; W10; W12-8など)。①300 bp未満をフィルタリングする前は総塩基数が(2,385,525 – 29506) = 2,356,019 bpであった。②フィルタリングして配列数が117 → 52個に減ったあとの総塩基数は(2,346,552 – 52) = 2,346,500 bp

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

 参考

W21-2：ACGTカウント2

①赤枠部分がA, C, G, T, Nの出現回数をカウントするRコード。Step1, 2, 3の配列長フィルタリング前の出力結果ファイルを入力としている。②のコピペ実行結果が次のスライド。2016.07.20のスライド30-49で、ホストOSのR上で行った作業と同じことがLinux上でもできることがわかってもらえればそれでよし

スライド120まで省略

- ACGTカウント2[W20-11]

それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが挿
認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を用

②

```
pwd
ls *.fa
R -q
library(Biostrings)
```

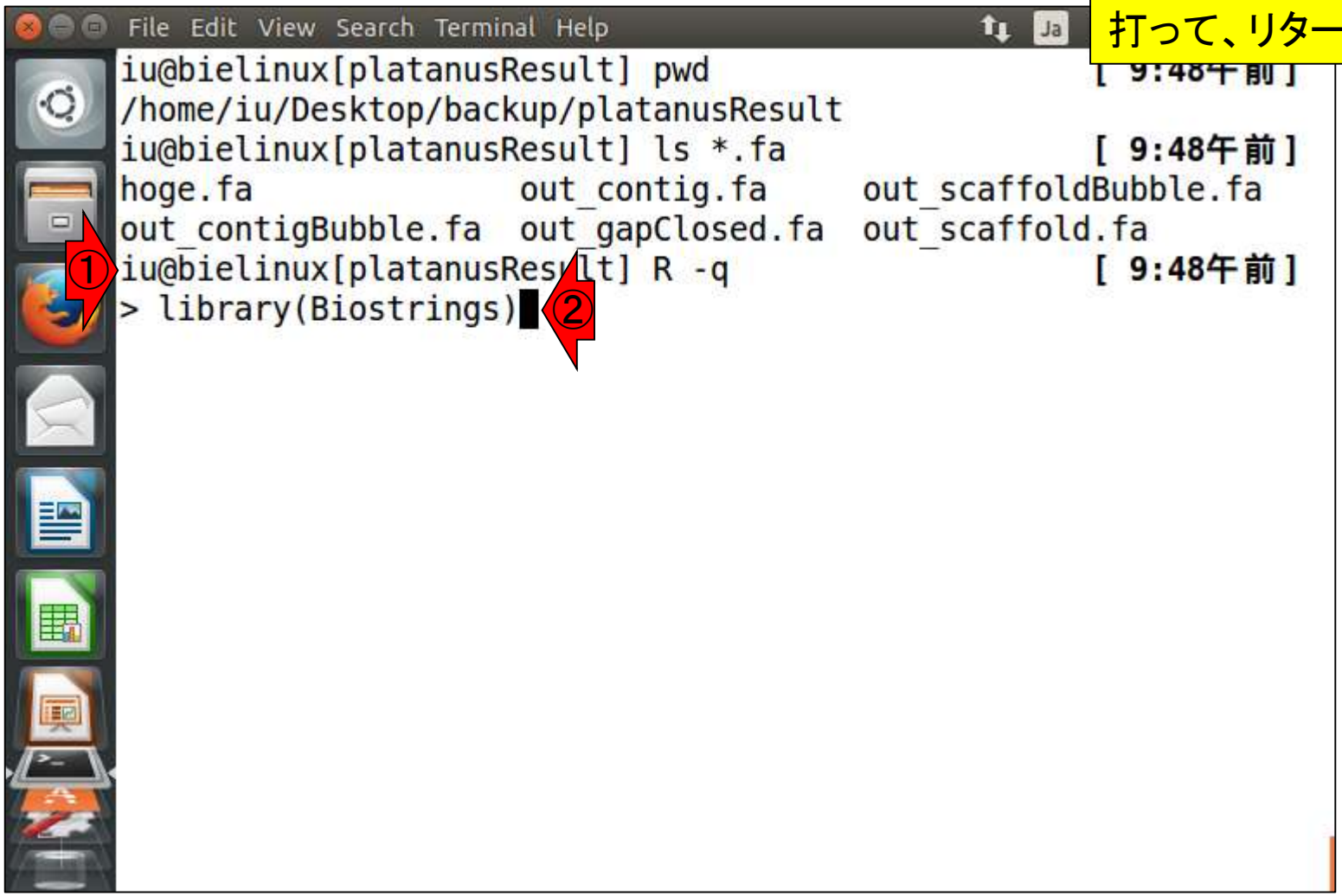
①

```
fasta <- readDNAStringSet("out_contig.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet("out_scaffold.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet("out_gapClosed.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)
```

 参考

W21-2：ACGTカウント2

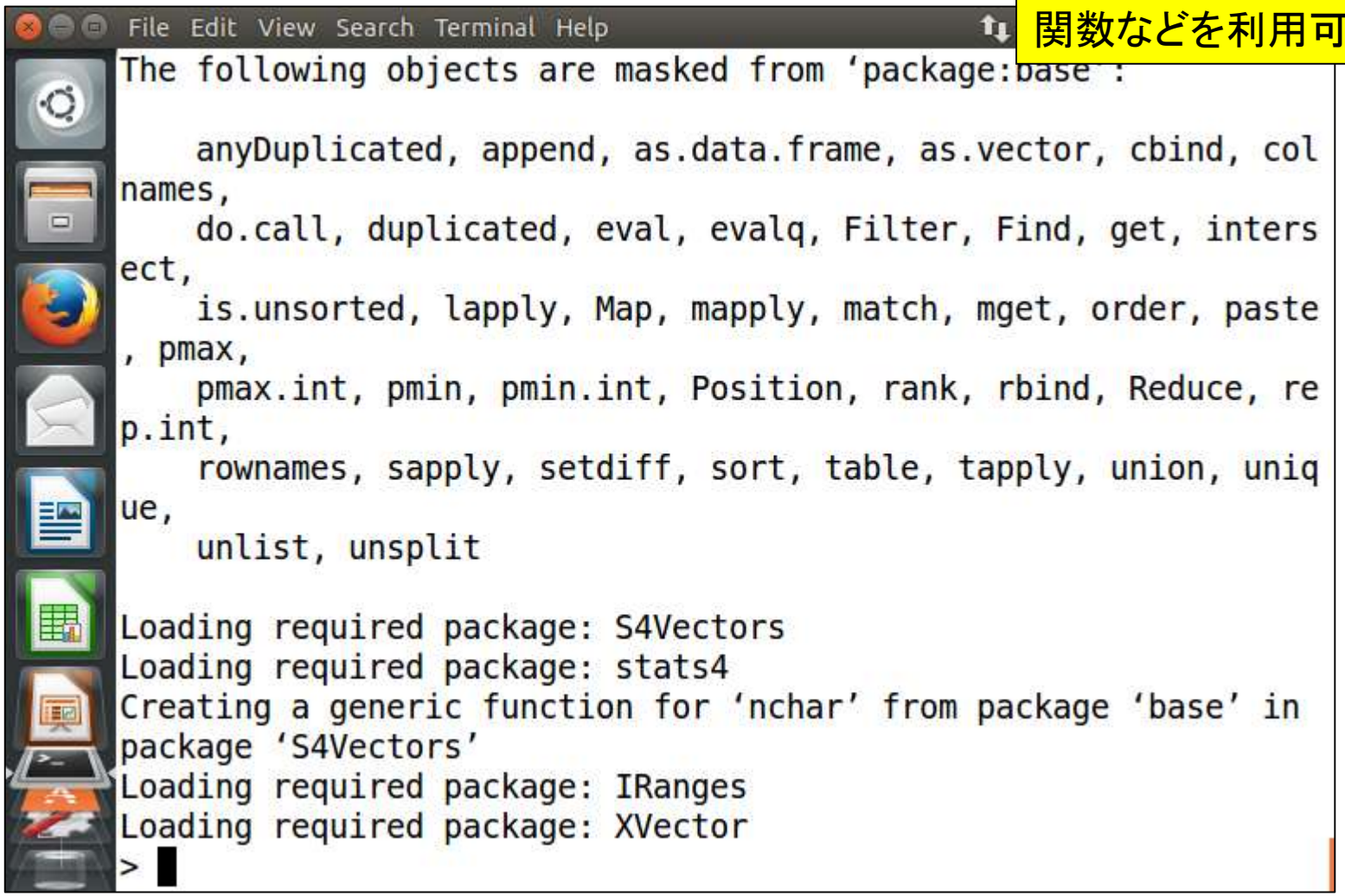
ACGTのカウントを調べたいファイルがあるディレクトリ上で、①Rを起動（第5回W9-8）。②library(Biostrings)と打って、リターンキーを押す



参考

W21-2：ACGTカウント2

エラーなく読み込み完了。これでBiostringsパッケージが提供するACGTの文字列をカウントするalphabetFrequency関数などを利用可能な状態になった



参考

まずは、①PlatanusのStep1実行結果ファイル(out_contig.fa)を入力として、「A, C, G, T, Nの出現回数をカウントするRコード」をコピペ実行

W21-2：ACGTカウント2

- ACGTカウント2[W20-11]
  それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが挿入されているのかなどを確認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を用いた。連載第5回のW9-8。

```
pwd
ls *.fa
R -q
library(Biostrings)

fasta <- readDNAStringSet("out_contig.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet("out_scaffold.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet("out_gapClosed.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)
```

①

 参考

W21-2：ACGTカウント2

コピペ実行結果。①の実行結果部分が、A, C, G, T, Nの塩基ごとの出現回数。②Nは1つもないことがわかる

- ACGTカウント2[W20-11]

  それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが挿入されているのかなどを確認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を用いた。連載第5回のW9-8。

```
pwd
ls *.fa
R -q
library(Biostrings)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)
```

①

File Edit View Search Terminal Help 09:56

```
    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, re
p.int,
    rownames, sapply, setdiff, sort, table, tapply, union, uniq
ue,
    unlist, unsplit

Loading required package: S4Vectors
Loading required package: stats4
Creating a generic function for ‘nchar’ from package ‘base’ in
package ‘S4Vectors’
Loading required package: IRanges
Loading required package: XVector
> fasta <- readDNAStringSet("out_contig.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
739836 469377 446806 742714      0
> sum(hoge)
[1] 2398733
>
```

①

②

参考

W21-2：ACGTカウント

①の実行結果部分は、総塩基数を計算しているところ。②alphabetFrequency関数実行によって得られた結果をhogeというものに格納している。このhogeを入力として総和を計算するsum関数を適用して全塩基数とみなしている

- ACGTカウント 2[W20-11]

それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが
認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を

```
pwd
ls *.fa
R -q
library(Biostrings)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)
```

①

File Edit View Search Terminal Help 09:56

```
    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, re
p.int,
    rownames, sapply, setdiff, sort, table, tapply, union, uniq
ue,
    unlist, unsplit

Loading required package: S4Vectors
Loading required package: stats4
Creating a generic function for ‘nchar’ from package ‘base’ in
package ‘S4Vectors’
Loading required package: IRanges
Loading required package: XVector
> fasta <- readDNAStringSet("out_contig.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
739836 469377 446806 742714      0
> sum(hoge)
[1] 2398733
>
```

②

①

参考

W21-2：ACGTカウント2

①Step2実行結果ファイル(out_scaffold.fa)を入力として実行した結果。Scaffoldingによってコンティグ間が未知塩基Nで埋められたギャップで表現されるため、②Nが存在（491個）するのは妥当

- ACGTカウント2[W20-11]

それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが挿入さ
認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を用いた

```
pwd
ls *.fa
R -q
library(Biostrings)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)
```

```
File  Edit  View  Search  Terminal  Help                    Ja  10:01
Creating a generic function for 'nchar' from package 'base' in
package 'S4Vectors'
Loading required package: IRanges
Loading required package: XVector
> fasta <- readDNAStringSet("out_contig.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
739836 469377 446806 742714      0
> sum(hoge)
[1] 2398733
> fasta <- readDNAStringSet("out_scaffold.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
729635 458119 440510 727306    491
> sum(hoge)
[1] 2356061
>
```

①

②

参考

W21-2：ACGTカウ

①Step3実行結果ファイル(out_gapClosed.fa)を入力として実行した結果。Gap closingのおかげで、この場合は②Nが0個になっている。③総塩基数2,356,019 bpという数値は、W20-7のwcコマンドから得られる結果と同じ。同じ目的を達成する上でも様々な手段がある

- ACGTカウント2[W20-11]

それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffol
認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC co

```
pwd
ls *.fa
R -q
library(Biostrings)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSe
hoge <- alphabetFrequenc
obj <- is.element(colnam
colSums(hoge)[obj]
sum(hoge)
```

File Edit View Search Terminal Help 10:03

```
     A      C      G      T      N
739836 469377 446806 742714      0
> sum(hoge)
[1] 2398733
> fasta <- readDNAStringSet("out_scaffold.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
729635 458119 440510 727306    491
> sum(hoge)
[1] 2356061
> fasta <- readDNAStringSet("out_gapClosed.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
729772 458211 440585 727451      0
> sum(hoge)
[1] 2356019
>
```

①

②

③

①Rの終了（第5回W9-8）

W21-2：Rの終了

- ACGTカウント2[W20-11]
  それぞれのA, C, G, T, Nのカウント数を調べて、本当にscaffold時にNが挿入されているのかなどを確認したい。テンプレートとしては、「解析 | 一般 | GC含量(GC contents)」を用いた。連載第5回のW9-8。

```
R -q
library(Biostrings)

fasta <- readDNAStringSet
hoge <- alphabetFrequency
obj <- is.element(colname
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet
hoge <- alphabetFrequency
obj <- is.element(colname
colSums(hoge)[obj]
sum(hoge)

fasta <- readDNAStringSet
hoge <- alphabetFrequency
obj <- is.element(colname
colSums(hoge)[obj]
sum(hoge)

q(save="no")
```

①

File Edit View Search Terminal Help　Ja　10:06

```
739836 469377 446806 742714      0
> sum(hoge)
[1] 2398733
> fasta <- readDNAStringSet("out_scaffold.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
729635 458119 440510 727306    491
> sum(hoge)
[1] 2356061
> fasta <- readDNAStringSet("out_gapClosed.fa", format="fasta")
> hoge <- alphabetFrequency(fasta)
> obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
> colSums(hoge)[obj]
     A      C      G      T      N
729772 458211 440585 727451      0
> sum(hoge)
[1] 2356019
> q(save="no")
iu@bielinux[platanusResult]                          [10:06午前]
```

①

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

おさらい

①乳酸菌ゲノム配列決定論文は、2種類のNGS機器から得られたデータを併用している。
第6回はIllumina MiSeqデータ(DRR024501)を、
連載第7回はPacBioデータを取り扱っている

- PacBio RS IIデータ(DRR024500)
  - DRR024500は登録内容に問題があったことが判明し消滅
  - 4セル分のデータ。DRR054113-054116に差し替えられている
  - セルあたり約15万リード。4セル分なので約60万リード
- Illumina MiSeqデータ(DRR024501)
  - paired-endゲノムデータ
  - リード長は、forward側とreverse側共に250 bp
  - オリジナルは2,971,310リード。最初の300,000リードを解析
  - forward側(DRR024501sub_1.fastq.gz)
  - reverse側(DRR024501sub_2.fastq.gz)
- FaQCs実行結果(第6回W5-4)
  - 300,000リード → 297,633リード (W5-2)
  - forward側(QC.1.trimmed.fastq.gz)
  - reverse側(QC.2.trimmed.fastq.gz)
  - ファイルの場所:~/Documents/DRR024501/result

①

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

• 書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ

NGS連載第7回は

②このあたりからスタート。ロングリードの代表格であるPacBioデータとその周辺の話。オリジナルの第7回ウェブ資料PDFと若干順番を入れ替えながら、まずは事実の羅列から話を進めます

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モ
(last modified 2016/06/03, since 2011)

What's new?

- このウェブページ
  リーソフトRと必要
  法(Windows2015.
  あります。(2015/0

- 書籍 | 日本乳酸菌学会誌 | 第5回アセンブル、マッピング、そしてQC (last modified 20
- 書籍 | 日本乳酸菌学会誌 | 第6回ゲノムアセンブリ (last m ified 2016/06/17) NEW
- 書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ modified 2016/05/12)
- イントロ | 一般 | ランダムに行を抽出 (last modified 2014/0 17)
- イントロ | 一般 | 任意の文字列を行の最初に挿入 (last modified 2014/07/17)
- イントロ | 一般 | 任意の
- イントロ | 一般 | ランダ
- イントロ | 一般 | 任意の

①

書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ

日本乳酸菌学会誌の第7回分です。Linuxコマンドのリンク先は主に日経BP社様です。

- 原稿PDF
- ウェブ資料PDF
  - Windows用(2016.0 版: 約20M用)
  - Macintosh用

②

PacBioのファイル形式とデータ解析の概要

- W1-1: PacificBiosciencesの YouTubeサイト
  - Introduction to SMRT Sequencing
  - Single Molecule Real Time Sequencing
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - DRR024500: Tanizawa et al., BMC Genomics, 2015
- W2-3: DRR024501 -> DRP002401 -> DRX022185
- W2-4: DRR024501 -> DRA002643
- W2-5: PacBioデータ概観
  - DRR054113
  - DRR054114
  - DRR054115
  - DRR054116
- W2-6: SMRT Portal(PacBio提供のHGAPを含む解析ソフトウェア群)の場所

Linuxコマンド

- apt-cache (パッケージ調査)

おさらい（NGS連載第3回）

①の第3回で行った議論は、
基本的にIlluminaデータの話。
PacBioデータには通用しない

全部読んで説明

- 3大公共DB（NGS分野）の提供ファイル形式
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2圧縮)
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式

  } 第3回W20からW21

- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FASTQ提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

  } ①

W2-7 : bax.h5ファイル

まず、①PacBioの生データは、sraでもFASTQでもなく、bax.h5という形式のファイル（正確にはPacBio RS IIという機器が出力するファイル形式）。②サイズも巨大。具体的には、1セル分のみでも(747MB + 766MB + 901MB) = 2,414MB (約2.4GB)に達する。③これはDRR054113の1セル分のデータだが、1ファイル/セルではなく、3つに分割されている点も最初は戸惑うポイント

PacBioのファイル形式とデータ解析の概要

- W1-1: PacificBiosciencesの YouTubeサイト
  - Introduction to SMRT Sequencing
  - Single Molecule Real Time Sequencing
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - DRR024500: Tanizawa et al., BMC Genomics, 2015
- W2-3: DRR024501 -> DRP002401 -> DRX022185
- W2-4: DRR024501 -> DRA002643
- W2-5: PacBioデータ概観
  - DRR054113
  - DRR054114
  - DRR054115
  - DRR054116
- W2-6: SMRT Portal(PacBio提供のHGAPを含む解析ソフトウェア群)の場所
  - PacBio -> DevNet -> SMRT Analysis
  - SMRT Analysis 2.3までは、HGAPを実行するためにはbax.h5ファイルが必須。
  - SMRT Analysis 3.0からは、BAMファイルが入力フォーマットになる。但しここでのBAMファイルは、マッピングデータではなく、シークエンス生データ。
  - PacBio RSIIの後継機であるSequelの出力ファイル形式はBAM。
  - PacBioのファイル形式の説明についてはこちら (http://pacbiofileformats.readthedocs[illegible]3.0/)。
- W2-7: DRR054113のbax.h5ファイル (下記3ファイル合わせてDRR054113に相当)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5 (747 MB; 784,301,199 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5 (766 MB; 803,938,042 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5 (901 MB; 945,597,712 bytes)

①

②

③

おさらい

- PacBio RS IIデータ(DRR024500)
  - DRR024500は登録内容に問題があったことが判明し消滅
  - 4セル分のデータ。DRR054113-054116に差し替えられている ②
  - セルあたり約15万リード。4セル分なので約60万リード
- Illumina MiSeqデータ(DRR024501)
  - paired-endゲノムデータ
  - リード長は、forward側とreverse側共に250 bp
  - オリジナルは2,971,310リード。最初の300,000リードを解析
  - forward側(DRR024501sub_1.fastq.gz)
  - reverse側(DRR024501sub_2.fastq.gz)
- FaQCs実行結果(第6回W5-4)
  - 300,000リード → 297,633リード (W5-2)
  - forward側(QC.1.trimmed.fastq.gz)
  - reverse側(QC.2.trimmed.fastq.gz)
  - ファイルの場所:~/Documents/DRR024501/result

①乳酸菌ゲノム配列決定論文は、②PacBio RS IIで4セル分のデータを取得し、*de novo*アセンブリを行った。DRA上では、セルごとにDRR054113-054116という計4つのIDで登録されている。その内の1つであるDRR054113について解説しました

①

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

W2-7 : bax.h5ファイル

PacBioのファイル形式とデータ解析の概要

- W1-1: PacificBiosciencesの YouTubeサイト
  - Introduction to SMRT Sequencing
  - Single Molecule Real Time Sequencing
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - DRR024500: Tanizawa et al., BMC Genomics, 2015
- W2-3: DRR024501 -> DRP002401 -> DRX022185
- W2-4: DRR024501 -> DRA002643
- W2-5: PacBioデータ概観
  - DRR054113
  - DRR054114
  - DRR054115
  - DRR054116
- W2-6: SMRT Portal(PacBio提供のHGAPを含む解析ソフトウェア群)の場所
  - PacBio -> DevNet -> SMRT Analysis
  - SMRT Analysis 2.3までは、HGAPを実行するためにはbax.h5ファイルが必須。
  - SMRT Analysis 3.0からは、BAMファイルが入力フォーマットになる。但しここでのBAMファイルは、マッピングデータではなく、シークエンス生データ。
  - PacBio RSIIの後継機であるSequelの出力ファイル形式はBAM。
  - PacBioのファイル形式の説明についてはこちら (http://pacbiofileformats.readthedocs.io/en/3.0/)。
- W2-7: DRR054113のbax.h5ファイル (下記3ファイル合わせてDRR054113に相当)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5 (747 MB; 784,301,199 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5 (766 MB; 803,938,042 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5 (901 MB; 945,597,712 bytes)

① HGAPはbax.h5ファイルのみしか受け付けない。

DDBJ Pipeline上では、PacBio用の*de novo*アセンブリ用プログラムHGAPを利用可能。しかし、① HGAPはbax.h5ファイルのみしか受け付けない。HGAPを内包するPacBio提供のSMRT Analysisシステムのインストールは超高難易度(個人の感想です)。しかも数百GBメモリ搭載マシンでないと動かせない(DDBJ Pipeline上でDRR054113のみを入力としてHGAPを実行しても120GB程度のメモリを要する)。共同研究などで他人に頼むなど以外の場合、通常はDDBJ Pipeline上でのHGAP実行一択

おさらい（NGS連載第3回）

- 3大公共DB（NGS分野）の提供ファイル形式
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2圧縮)
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式

①公共DB上では、bax.h5ファイルは公開されていません。sraやFASTQファイルが手元にあっても実質的には(DDBJ PipelineでHGAPを実行できないので)無意味。しかも②(クオリティ値がPacBioよりも高い)Illuminaデータの場合は、sraからFASTQへの変換時にリード数が若干減る程度だが…

- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FASTQ提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

連載第7回W3

- 3大公共DB（NGS分野）の提供ファイル形式
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2圧縮)
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式

②

①PacBioデータの場合は相当減る。具体的には、②例えばDRA上で同じDRR054113をダウンロードしても、sraファイルは163,482リード（実際には若干異なる?!）であるのに対し、FASTQファイルだと915リードという結果になる。これは減ったリード数ではなく、生き残ったリード数

- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FASTQ提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも~~若干~~少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

①

連載第7回W3

- 3大公共DB（NGS分野）の提供ファイル形式
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2圧
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式
- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FASTQ提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

①これはおそらく、fastq-dump実行時に指定しているオプションの閾値が、Illuminaデータを合理的にフィルタリングするための条件のままで統一的にPacBioデータに対しても適用しているためであろう（推測の域を出ないが、論理的に考えればそのはず）

①

この後の展開は…

- 3大公共DB（NGS分野）の提供ファイル形
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2
  - EMBL-EBI ENA (ENA)：FASTQ形 ①gzip圧縮)
  - NCBI SRA (SRA)：sra形式
- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FAS ②提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

①DRA上でダウンロードしたDRR054113のFASTQファイルが915リードになるのをFastQCで確認します(第7回W3-2)。次に、(第4回W4-5, W13-5, W14, W15でも紹介した)apt-getを用いたプログラムインストールの応用編(第7回W4)として、②SRA Toolkitを解説

念のため説明

- 3大公共DB（NGS分野）の提供ファイル形
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式
- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FAS①提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

①SRA Toolkitは、第7回のオリジナルウェブ資料(W5)ではFASTQファイルを生成してFastQCで確認する用途として利用しています。しかし前述のように、PacBioは入力がbax.h5ファイルなので「PacBioデータの性質を確認できてよかったね」の域を出ません。そしてIlluminaの場合は最初からFASTQファイルを使えばよいので、sraファイルを取り扱う意義は見出せません(第3回原稿PDFのp36左下)

念のため説明

- 3大公共DB（NGS分野）の提供ファイル形
  - DDBJ SRA (DRA)：sra形式とFASTQ形式(bzip2
  - EMBL-EBI ENA (ENA)：FASTQ形式(gzip圧縮)
  - NCBI SRA (SRA)：sra形式
- 大元はsra形式なので…
  - FASTQファイルはsraファイルから作成する（sra → FASTQ）
  - SRA Toolkit中のfastq-dumpで「sra → FASTQ」の変換を行う
  - FAS①提供サイト（DRAとENA）は、fastq-dump実行時にクオリティフィルタリングなどを行っている。このため、FASTQファイル中のリード数は、元のsraファイル中のリード数よりも若干少なくなる(第3回W24)。実際、SRR616268のsraファイルは135,073,834リード、FASTQファイルは134,755,996リードであった(第3回W24-3)。

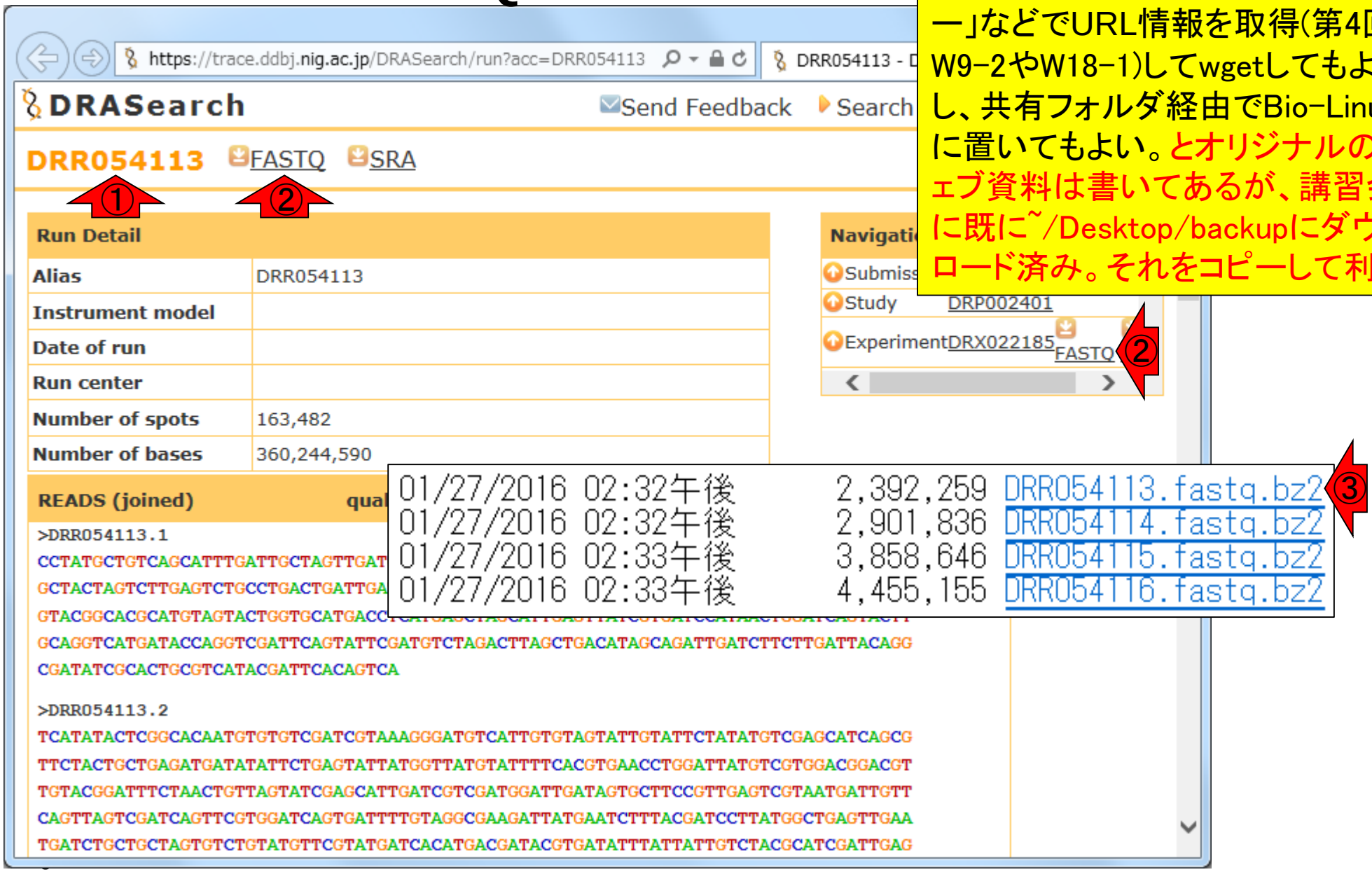
①SRA Toolkitは、あくまでもapt-getを用いたプログラムインストール(応用編; 第7回W4)の例題という位置づけ。apt-get(およびapt-cache)を使いこなして効率的にインストールを行うスキルがあれば、*de novo* transcriptome assemblyプログラムの1つであるTrinityのインストールも行えます(2016.08.04)

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

W3-1：FASTQダウンロード

①DRR054113、②FASTQ、③bzip2圧縮FASTQファイルをダウンロード。右クリックで「ショートカットのコピー」などでURL情報を取得(第4回W9-2やW18-1)してwgetしてもよいし、共有フォルダ経由でBio-Linux上に置いてもよい。とオリジナルのウェブ資料は書いてあるが、講習会用に既に~/Desktop/backupにダウンロード済み。それをコピーして利用



W3-1：FASTQダウンロード

①今はこのあたりの話をしている。②「W3-1」という情報を頼りに探す。③赤枠をそのまま実行すればwgetは実行されないが、大量同時アクセスでDDBJに迷惑をかけないように、~/Desktop/backup上にあるファイルのコピーにしてください

PacBioデータの概観1

①

②

W3-1: DRR0[illegible]13のFASTQファイルをダウンロード
wget実行時に-qをつけて途中経過を非表示。URL情報取得については、連載第4回W9-2

```
cd ~/Documents
pwd
mkdir DRR054113
cd DRR054113
pwd
#wget -cq ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/DRA002/DRA002643/DRX022185/DRR054113.fastq.bz2
cp ~/Desktop/backup/DRR054113.fastq.bz2 .
ls -l
ls -lh
```

③

- W3-2: FastQC (ver. 0.11.4)実行
  FastQC ver. 0.11.4は、第4回W9-2でインストールし、fastqc2というコマンドでパスを通している。

```
cd ~/Documents/DRR054113

pwd
ls -l
fastqc2 -v
fastqc2 -q DRR054113.fastq.bz2 --outdir=/home/iu/Desktop/mac_share
ls -l ~/Desktop/mac_share
```

- W3-3: 結果を眺める
  DRR054113_fastqc.html
- W3-4: 配列長分布
  DRR054113_fastqc.html

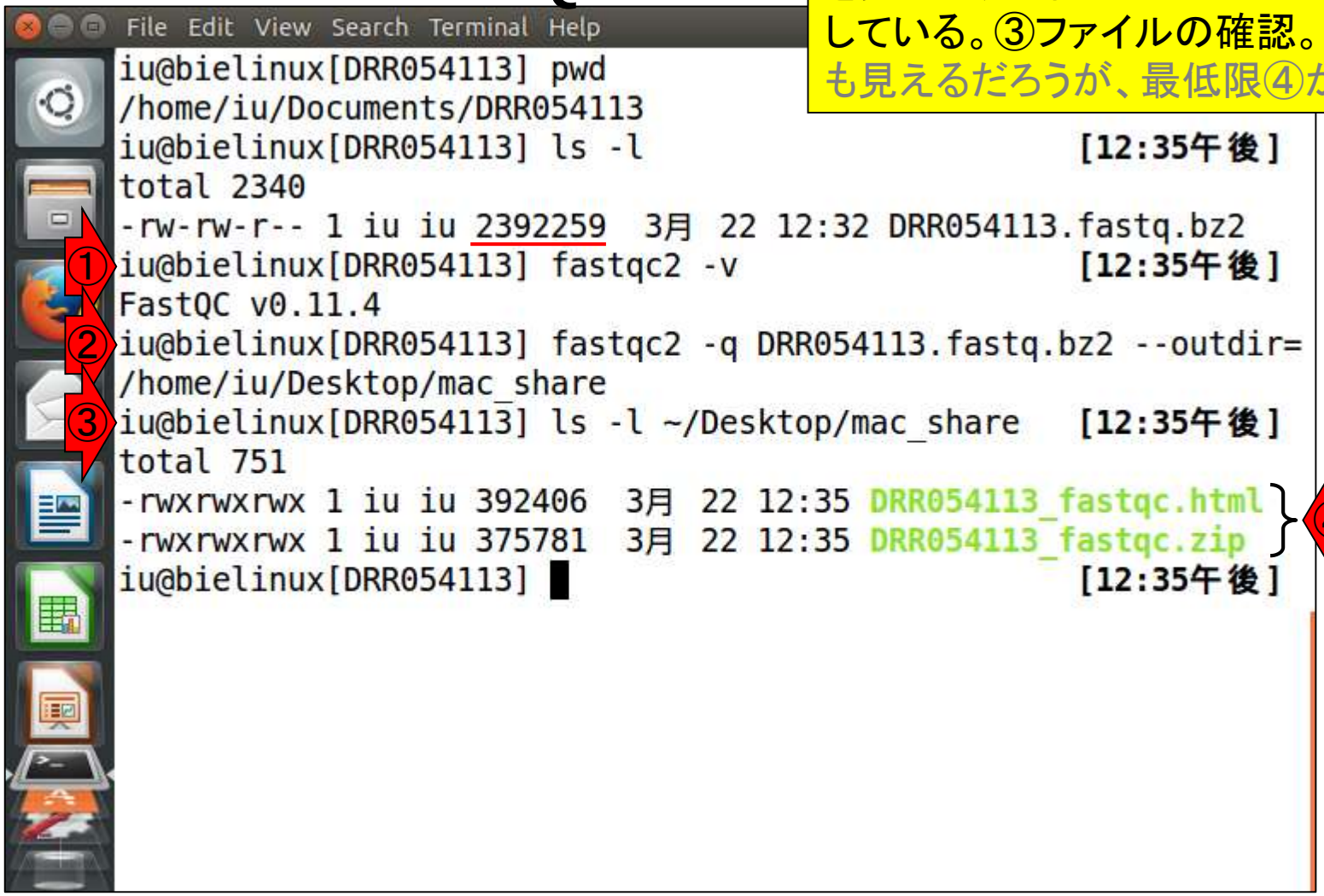
W3-1：FASTQダウンロード

つまり、①~/Documents/DRR054113で、②wgetの部分はcpコマンドのほうのコピペにしてください、ということ。
④ディレクトリ作成は絶対に行い、指定された場所で作業を行うこと！

```
File  Edit  View  Search  Terminal  Help
iu@bielinux[iu] cd ~/Documents
iu@bielinux[Documents] pwd
/home/iu/Documents
iu@bielinux[Documents] mkdir DRR054113 ④          [12:32午後]
iu@bielinux[Documents] cd DRR054113               [12:32午後]
①iu@bielinux[DRR054113] pwd                       [12:32午後]
/home/iu/Documents/DRR054113
②iu@bielinux[DRR054113] wget -cq ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/DRA002/DRA002643/DRX022185/DRR054113.fastq.bz2
iu@bielinux[DRR054113] ls -l                      [12:32午後]
total 2340
-rw-rw-r-- 1 iu iu 2392259  3月 22 12:32 DRR054113.fastq.bz2
③iu@bielinux[DRR054113] ls -lh                    [12:33午後]
total 2.3M
-rw-rw-r-- 1 iu iu 2.3M  3月 22 12:32 DRR054113.fastq.bz2
iu@bielinux[DRR054113]                            [12:33午後]
```

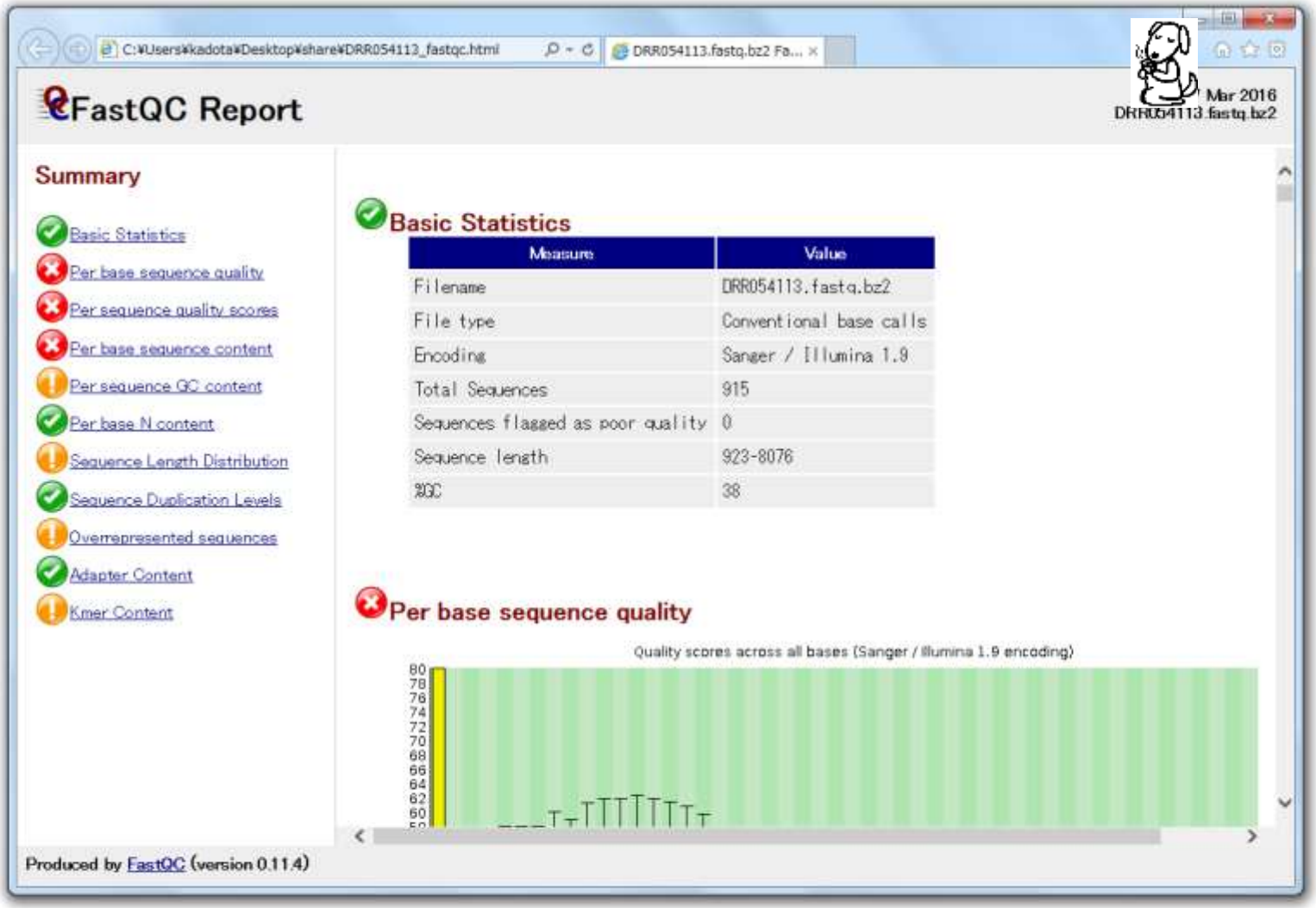
W3-2：FastQC

①FastQC ver. 0.11.4は、第4回W9-2でインストールし、fastqc2というコマンドでパスを通している。②FastQCを実行し、共有フォルダ(~/Desktop/mac_share)に保存している。③ファイルの確認。ヒトによっては他のものも見えるだろうが、最低限④が見えていれば問題ない



W3-3：結果を眺める

①共有フォルダに保存することで、使いなれたホストOS(この場合Windows)上で②FastQC実行結果ファイルを眺めることができる



W3-3：結果を眺める

FastQC実行結果の解説は第4回W8とW17、および第6回W4にもあり。①入力ファイル。②リード数は915、③配列長は923-8076 bpの範囲であることがわかる

FastQC Report

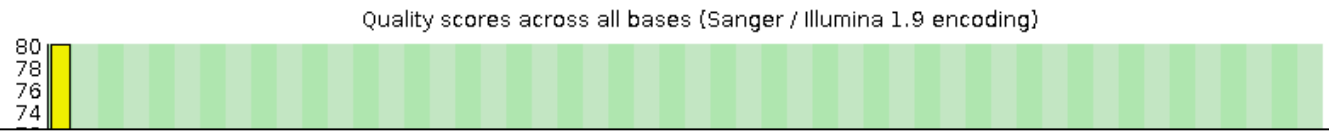
Summary

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

Basic Statistics

Measure	Value
Filename	DRR054113.fastq.bz2 ①
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	915 ②
Sequences flagged as poor quality	0
Sequence length	923-8076 ③
%GC	38

Per base sequence quality



W3-3：結果を眺める

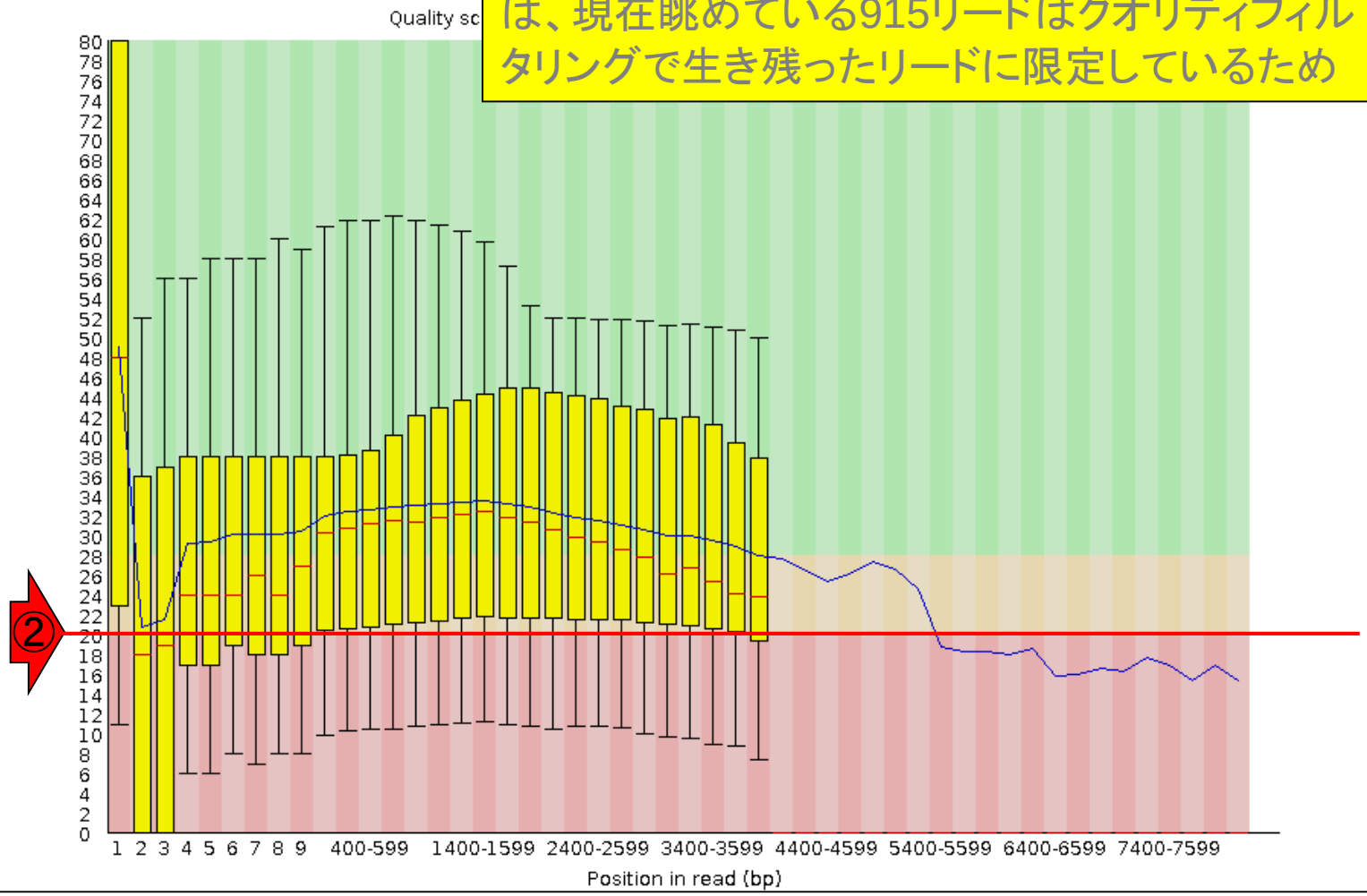
①Per base sequence quality。この図の縦軸はクオリティスコア。②赤線のスコア20を超えているかどうかが1つの目安。Illumina HiSeq2000 (第4回W8)やMiSeq (第6回W4)とは傾向が異なる。第7回W6では詳述しているが、PacBioの公称スペックよりも全体的なクオリティが高い。この理由は、現在眺めている915リードはクオリティフィルタリングで生き残ったリードに限定しているため

FastQC Report

Summary

- Basic Statistics
- Per base sequence quality ①
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Per base sequence quality



W3-3：結果を眺める

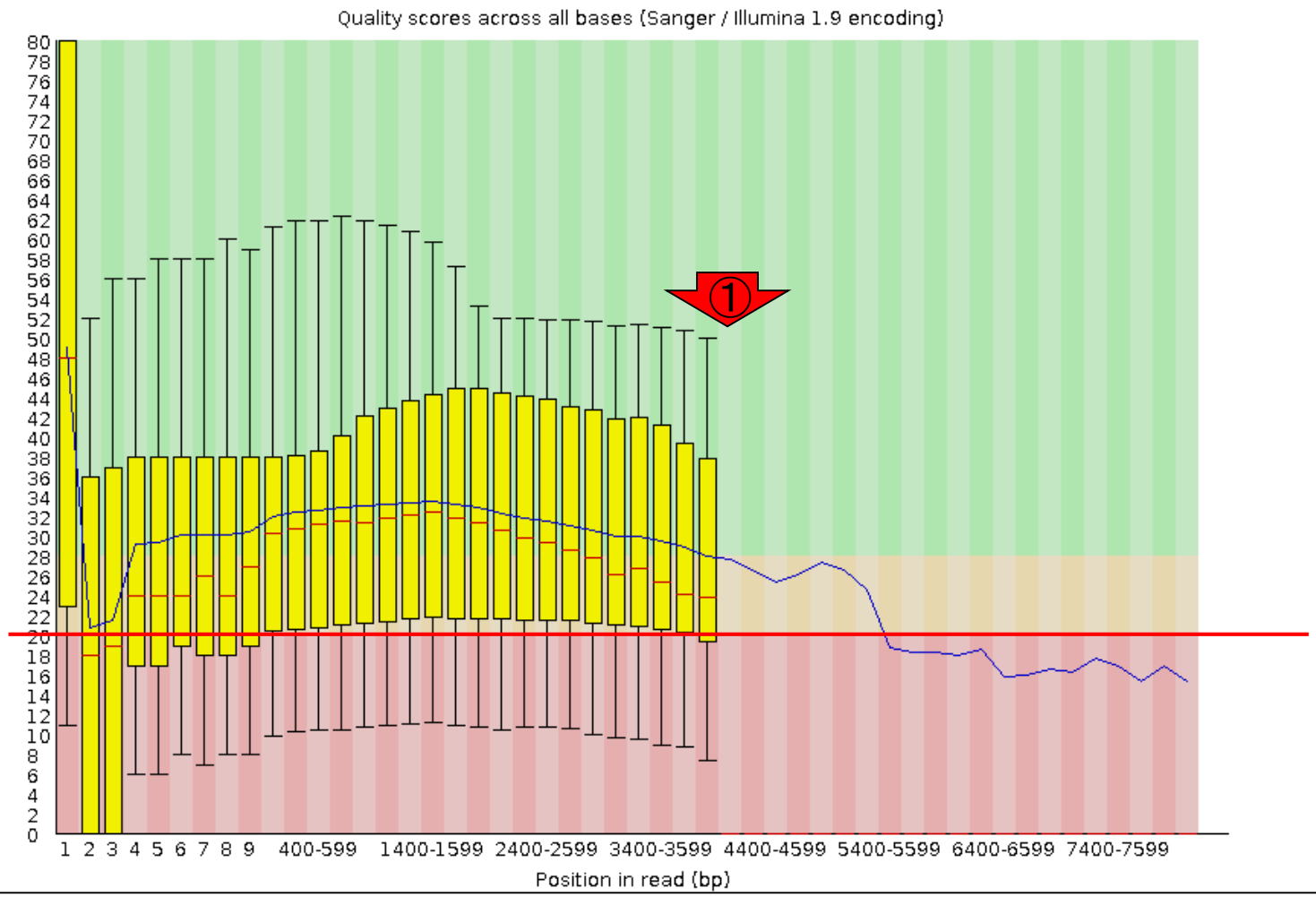
①横軸のリードポジションが4000 bpあたりで黄色の縦棒がなくなっている。この理由は、4000 bp以上のリードが少数だからだと思われる。それは②の配列長分布(Sequence Length Distribution)でも確認できる

FastQC Report

Summary

- Basic Statistics
- Per base sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution ②
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Per base sequence quality



W3-4：配列長分布

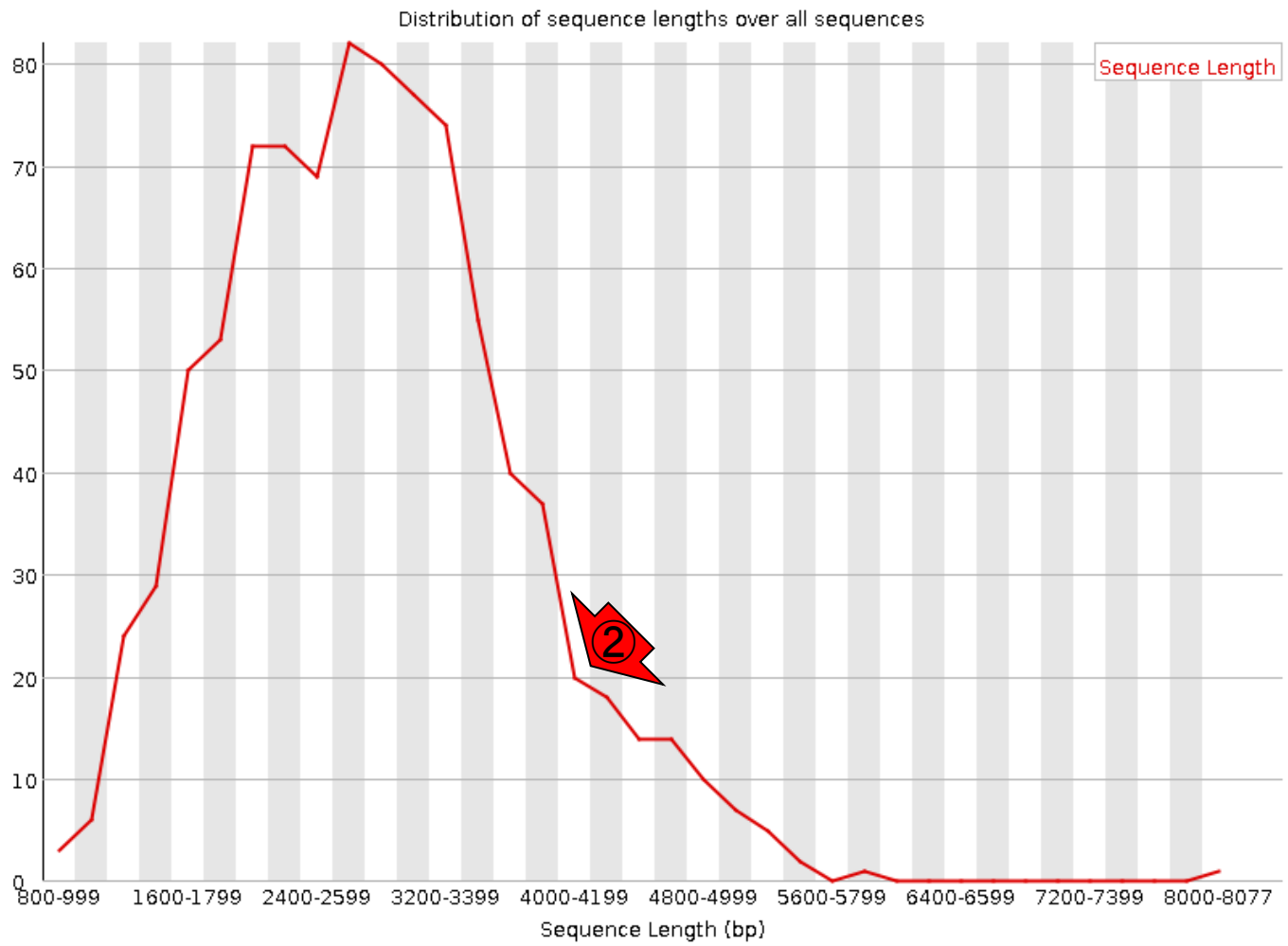
①配列長分布(Sequence Length Distribution)。②このあたりで黄色の縦棒がなくなっているので、おそらく20リードが黄色の縦棒の有無の閾値なのだろう

FastQC Report

Summary

- Basic Statistics
- Per base sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution ①
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Sequence Length Distribution



Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

• 書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ

NGS連載第7回の…

②場所はこのあたり。次のスライドは③のリンク先ですが、基本的にコマンド打ち込み部分以外は、リンク先にアクセスせずに、スライドを眺めるだけでよい。ネットワーク経由の作業なのでうまくいかない可能性もあるが、その場合はスライドを眺めながらのエアーハンズオンに切り替え。インストールに失敗しても、その後に影響しないような構成にしてます

(Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデ

(last modified 2016/06/03, since 2011)

What's new?

- このウェブページ
- リーソフトRと必要
- 法(Windows2015.

- 書籍 | 日本乳酸菌学会誌 | 第5回アセンブル、マッピング、そしてQC
- 書籍 | 日本乳酸菌学会誌 | 第6回ゲノムアセンブリ (last modified 20
- 書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ
- イントロ | 一般 | ランダムに行を抽出 (last modified 2014/07/17)
- イントロ | 一般 | 任意の文字列を行の最初に挿入 (last modified 2014/07/17)
- イントロ | 一般 | 任意の
- イントロ | 一般 | ランダ
- イントロ | 一般 | 任意の

① ③

書籍 | 日本乳酸菌学会誌 | 第7回ロングリードアセンブリ

日本乳酸菌学会誌の第7回分です。Linuxコマンドのリンク先は主に日経BP社様です。

- 原稿PDF
- ウェブ資料
  - Win
  - Macint

Linuxコマンド

- apt-cache (パッ

② ③

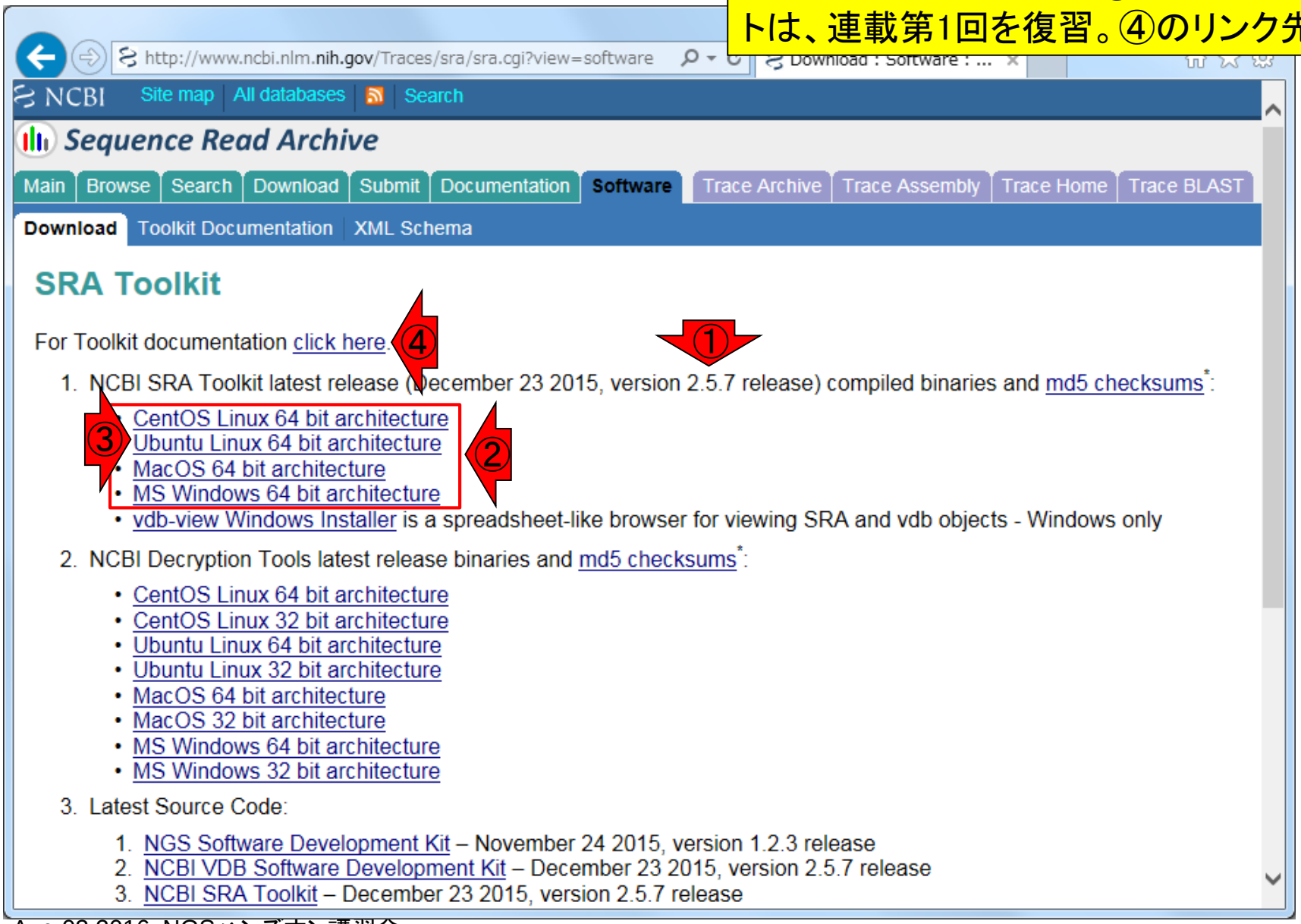
SRA Toolkit (ver. 2.5.7)のインストールと利用

- W4-1: SRA Toolkit
  2016年3月22日現在、ver. 2.5.7が最新。
  - wget利用時のソフトウェアURL情報
  - SRA Toolkit Documentation
  - Installation and Configuration Guide
- W4-2: apt-cache
  「apt-cache -n search キーワード」で任意のキーワードを含むソフトウェア名をリストアップ。
  -nオプションは、パッケージ名のみ表示させよという意味。--names-onlyでもよい。

```
cd ~/Documents/DRR054113

pwd
apt-cache -n search SRA
apt-cache -n search SRA | wc
```

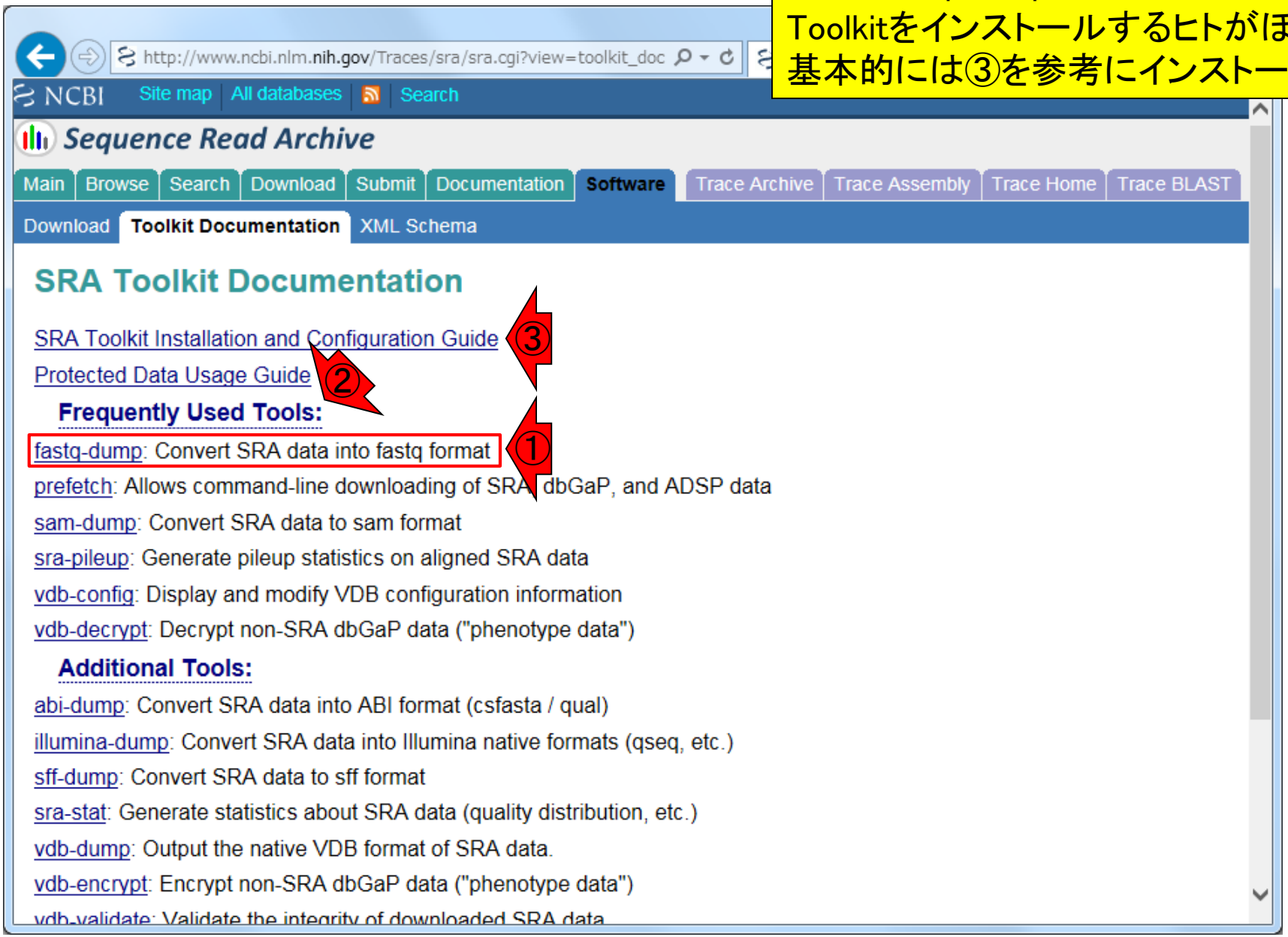
W4-1：SRA Toolkit

①2016年3月22日現在の最新版はver. 2.5.7。②プログラムはOSの種類ごとに用意されている。Bio-Linuxの実体はUbuntuなので③ここ。話についてこれないヒトは、連載第1回を復習。④のリンク先が次のスライド



W4-1：SRA Toolkit

①目的のfastq-dumpプログラムは、②よく使われるツール群(Frequently Used Tools)の最初に位置する。fastq-dumpを利用したいがために、SRA Toolkitをインストールするヒトがほとんどであろう。基本的には③を参考にインストールする。クリック



W4-1：SRA Toolkit

①wgetでtar.gzをダウンロードし、②解凍するやり方が示されているが…折角なので第4回W4-5, W13-5, W14, W15で紹介した「sudo apt-get install *ソフトウェア名*」でSRA Toolkitのインストールを行うやり方を伝授

SRA Toolkit Installation and Configuration Guide

Table of Contents

Contact: sra-tools@ncbi.nlm.nih.gov

The following guide will outline the download, installation, and configuration of the SRA Toolkit. Detailed information regarding the usage of individual tools in the SRA Toolkit can be found on the tool-specific documentation pages.

The NCBI SRA Toolkit enables reading ("dumping") of sequencing files from the SRA database and writing ("loading") files into the .sra format (Note that this is not required for submission). The Toolkit source code is provided in the form of the SRA SDK, and may be compiled with GCC. However, pre-built software executables are available for Linux, Windows, and Mac OS X, and we highly recommend using these pre-built executables whenever possible.

Downloading and installing the SRA Toolkit

Download the Toolkit from the SRA website

1. If you are using a web browser, the following page contains download links to the most current version of the toolkit for each of the supported platforms: SRA Toolkit download page: //www.ncbi.nlm.nih.gov/Traces/sra/?view=software

2. If you are instead working from a command line interface, you may use FTP or wget to obtain the software from the following directory: "//ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current". Example:

```
wget "//ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-centos_linux64.tar.gz"
```

①

Unpack the Toolkit:

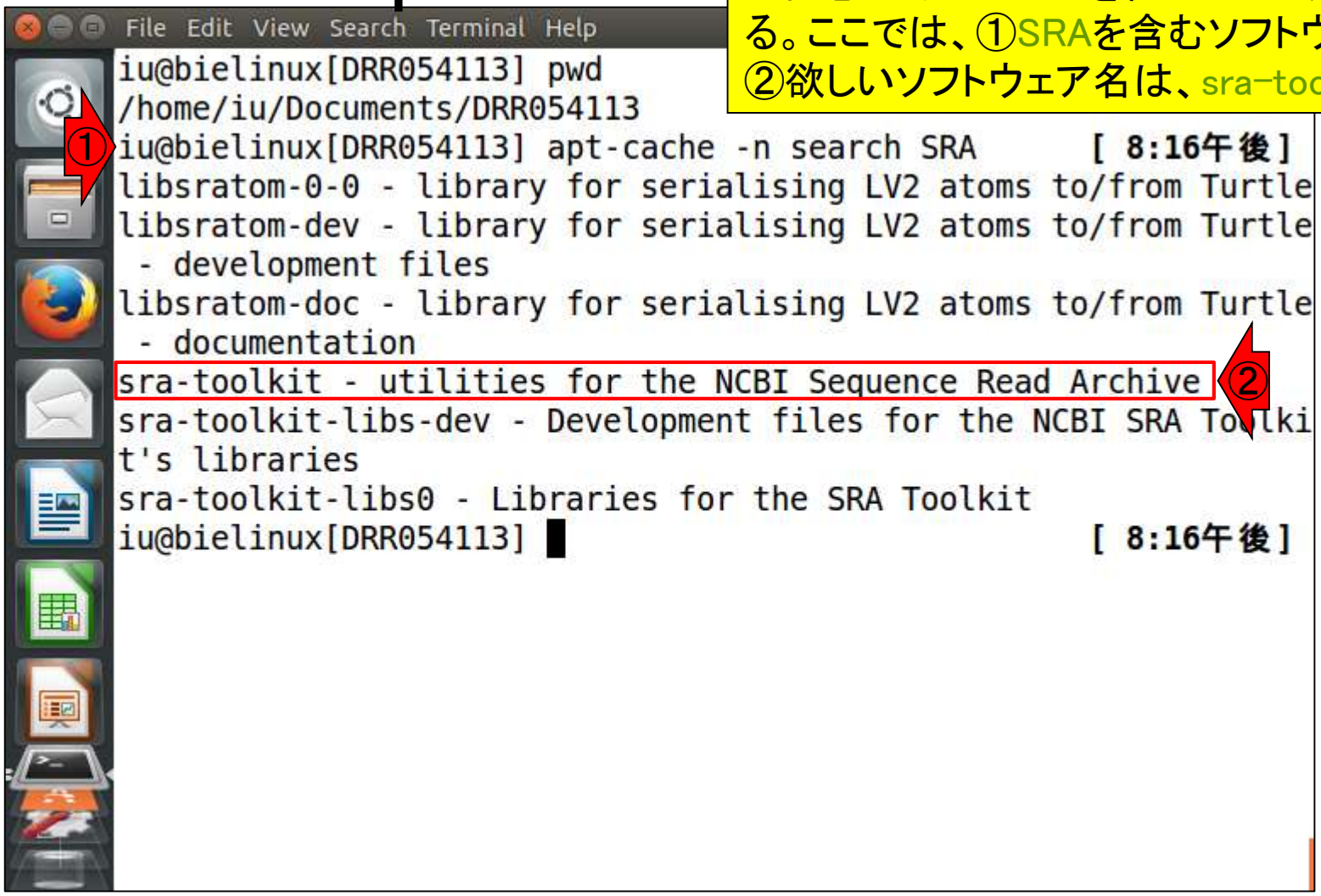
1. For Linux, use tar:

```
tar -xzf sratoolkit.current-centos_linux64.tar.gz
```

②

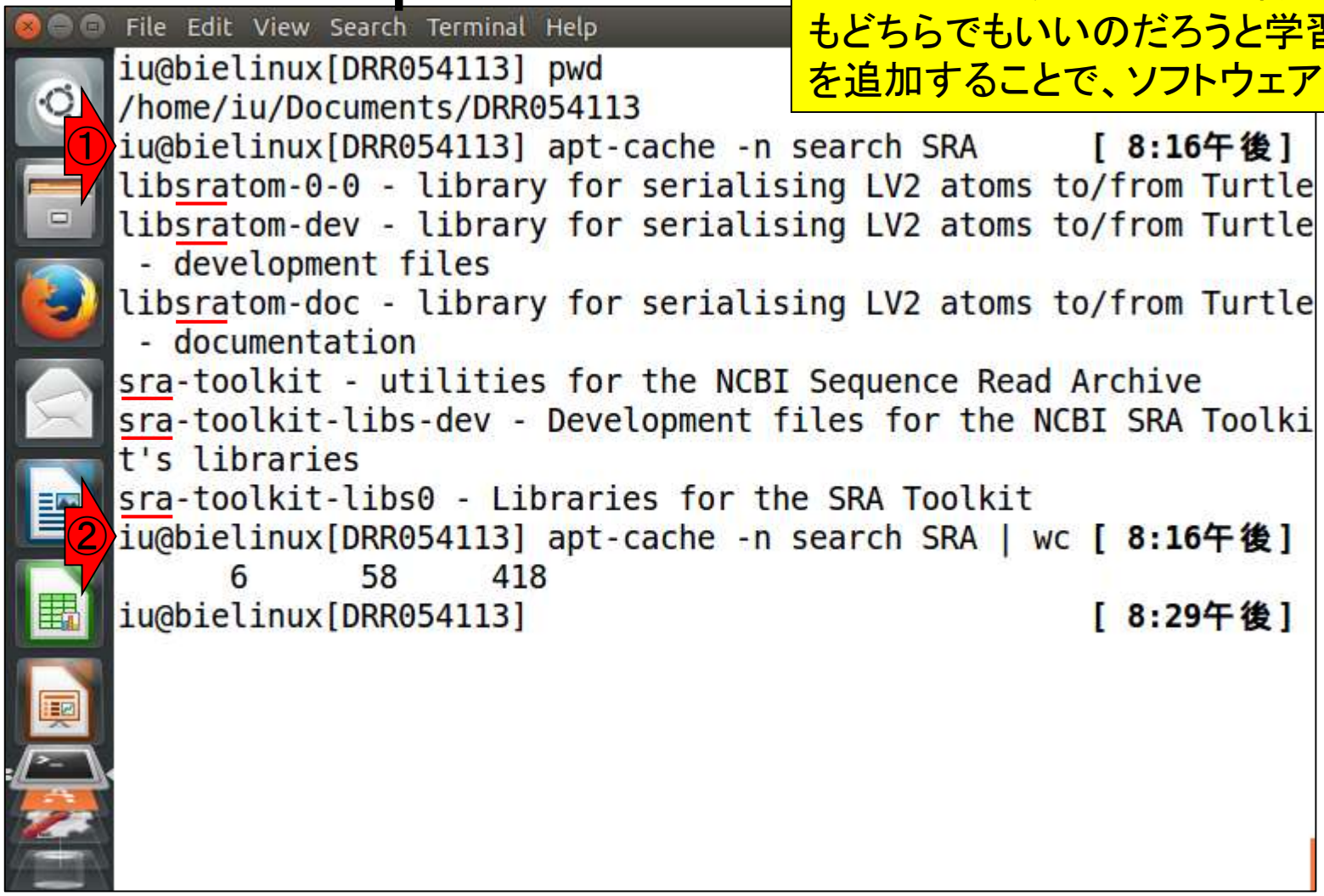
W4-2：apt-cache

目的：「sudo apt-get install *ソフトウェア名*」で指定する名前を知りたい！やり方：「apt-cache -n search *キーワード*」で任意のキーワードを含むソフトウェア名をリストアップする。ここでは、①SRAを含むソフトウェア名をリストアップ。②欲しいソフトウェア名は、sra-toolkitだということがわかる



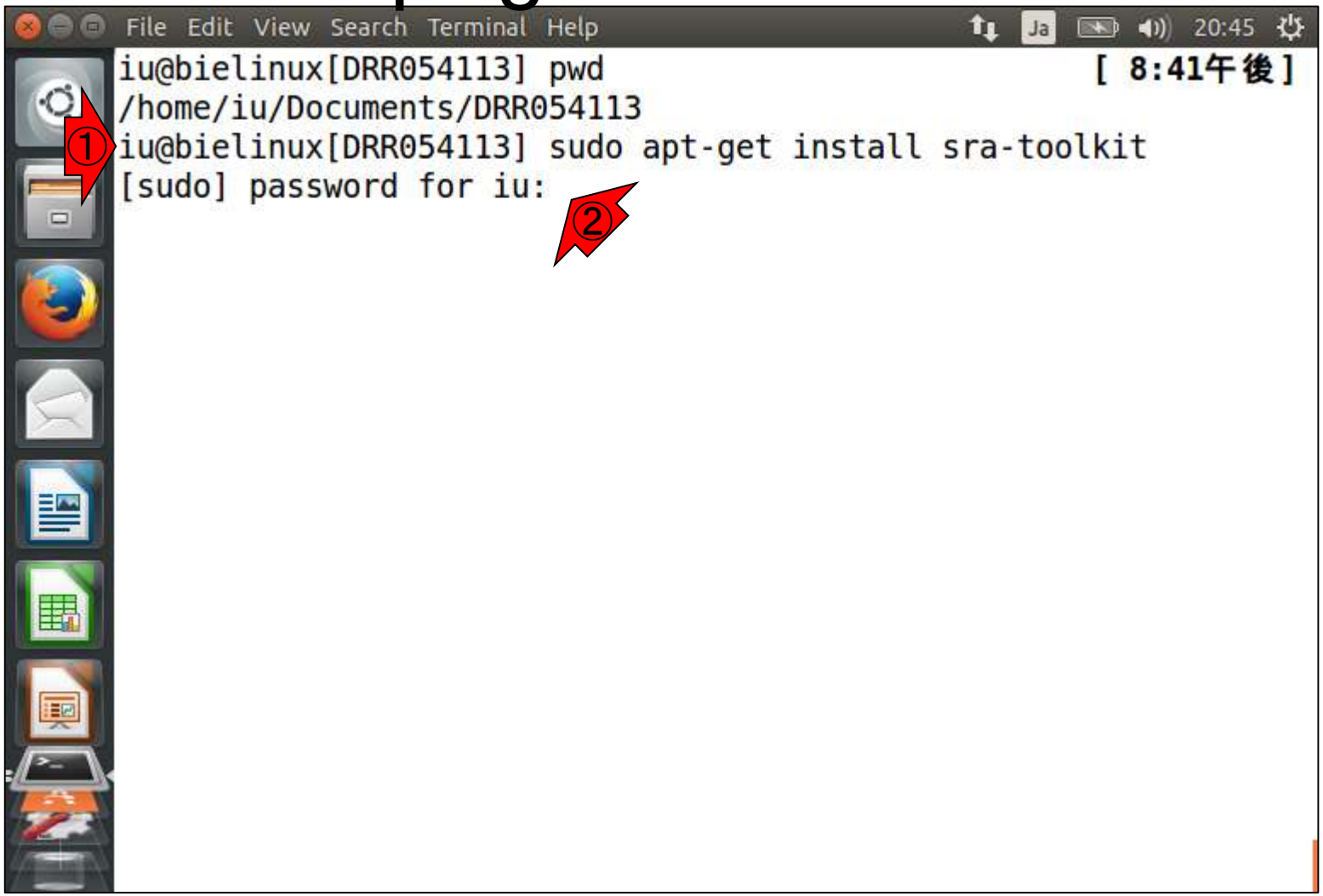
W4-2：apt-cache

おまけ。①「apt-cache -n search SRA」実行結果として、小文字のsraを含むソフトウェア名もリストアップされたことから、*キーワード*部分は、大文字でも小文字でもどちらでもいいのだろうと学習する。また、②「｜wc」を追加することで、ソフトウェア名が6個だったと認識



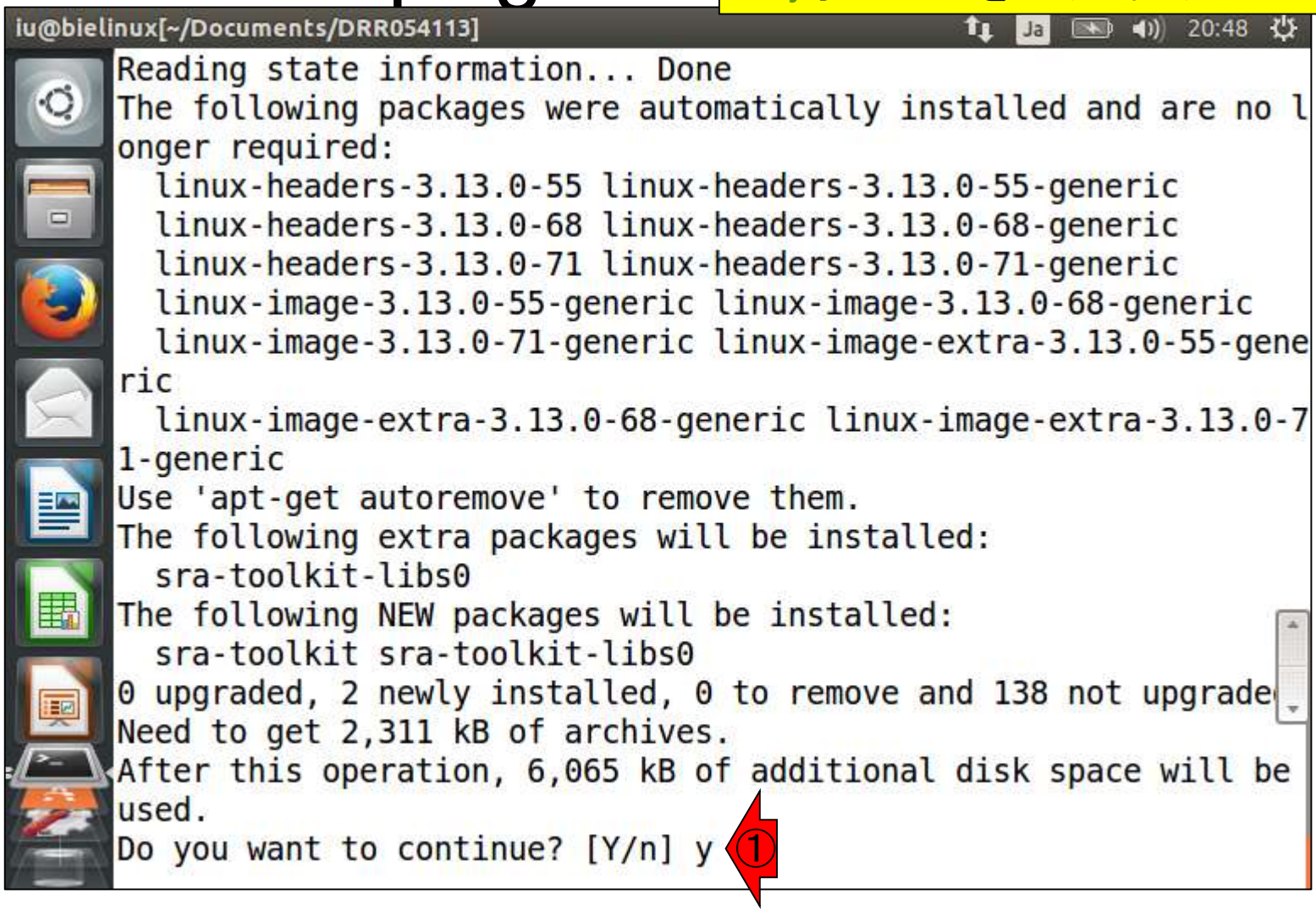
W4-3：apt-get

①「sudo apt-get install sra-toolkit」。rootのパスワードを聞かれたら打ち込む（推奨手順通りだとpass1409）



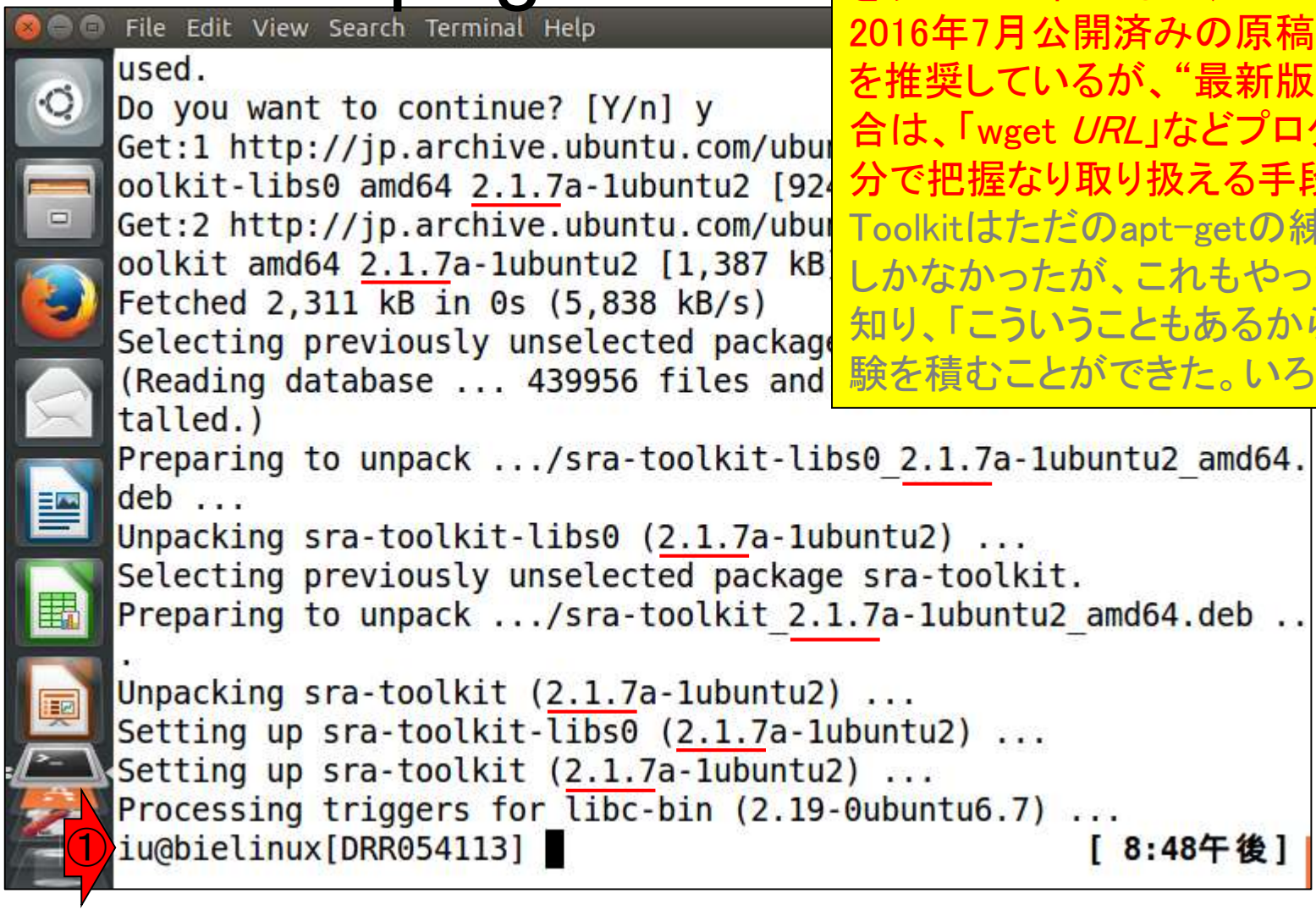
W4-3：apt-get

①Do you want to continue?と聞かれるので、y。イチイチ聞かれたくない場合は、「sudo apt-get -y install sra-toolkit」と-yオプションをつけておけばよい（第4回W15-1）



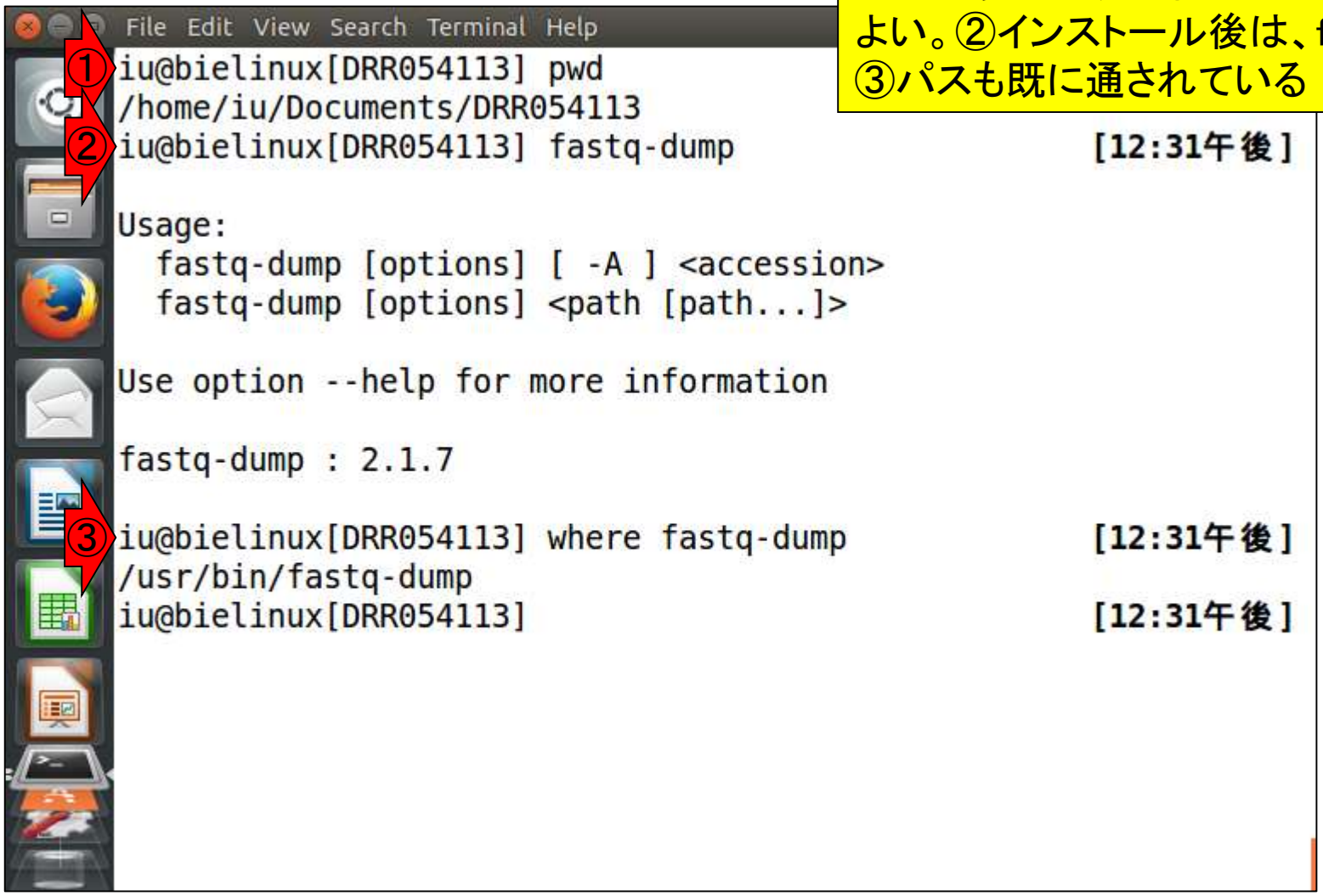
W4-3：apt-get

①インストール完了後の状態。2016年06月19日に気づいたこととして、赤下線部分とこの後のスライドを眺めればわかるが、バージョンがかなり古い。2016年7月公開済みの原稿PDFではapt-getの手順を推奨しているが、“最新版”をインストールしたい場合は、「wget *URL*」などプログラムのバージョンを自分で把握なり取り扱える手段でやりましょう。SRA Toolkitはただのapt-getの練習台程度の位置づけでしかなかったが、これもやったおかげで上記事実を知り、「こういうこともあるから気をつけねば」という経験を積むことができた。いろいろやるのは大事



W4-4：確認

①インストール作業は「~/Documents/DRR054113」で行ったが、「sudo apt-get install *ソフトウェア名*」でやる場合は、基本的にどの作業ディレクトリ上でもよい。②インストール後は、fastq-dumpを使用可能。③パスも既に通されている



W4-5：パスが通っている

「sudo apt-get install sra-toolkit」で無事インストール完了したあとは、W4-4で示したようにパスを通し終わった状態。それゆえ①SRA Toolkit Installation and Configuration Guide中の、②パスに関する注意書きは、気にしなくてもよい

SRA Toolkit Installation and Configuration Guide

①

Table of Contents

1. Downloading and installing the SRA Toolkit
2. Testing the Toolkit configuration
3. Configuring the Toolkit
4. Links and help documents

Contact: sra-tools@ncbi.nlm.nih.gov

The following guide will ou
usage of individual tools i

The NCBI SRA Toolkit en
the .sra format (Note that
may be compiled with GC
recommend using these p

Downloading and in

Download the Toolkit from

1. If you are using a we
the supported platfo

2. If you are instead wo
directory: "//ftp-trace
`wget "//ftp-tra`

Unpack the Toolkit:

1. For Linux, use tar:
`tar -xzf sra`

Unpack the Toolkit:

1. For Linux, use tar:
`tar -xzf sratoolkit.current-centos_linux64.tar.gz`

2. For Mac OS X, double-click on the .tar.gz file and the Archive Utility will unpack it. Alternatively, command-line tar will also work (see Linux example, above).

3. For Windows, either use an archiving and compression utility (e.g., Winzip, 7-Zip, etc.), or simply double-click on the .zip file and drag the 'sratoolkit...' folder to the preferred install location.

②

Note: For most users, the Toolkit functions (fastq-dump, sam-dump, etc.) will not be located in their PATH environmental variable. This may require providing directory information about the location of the Toolkit. See the below examples for how 'fastq-dump' would be called in different circumstances:

- `~/[user_name]/sra-toolkit/fastq-dump`
  YES: The Toolkit "bin" directory has been placed in the user-specified directory "sra-toolkit"

- `./fastq-dump`
  YES: The Toolkit components are the in the current working directory

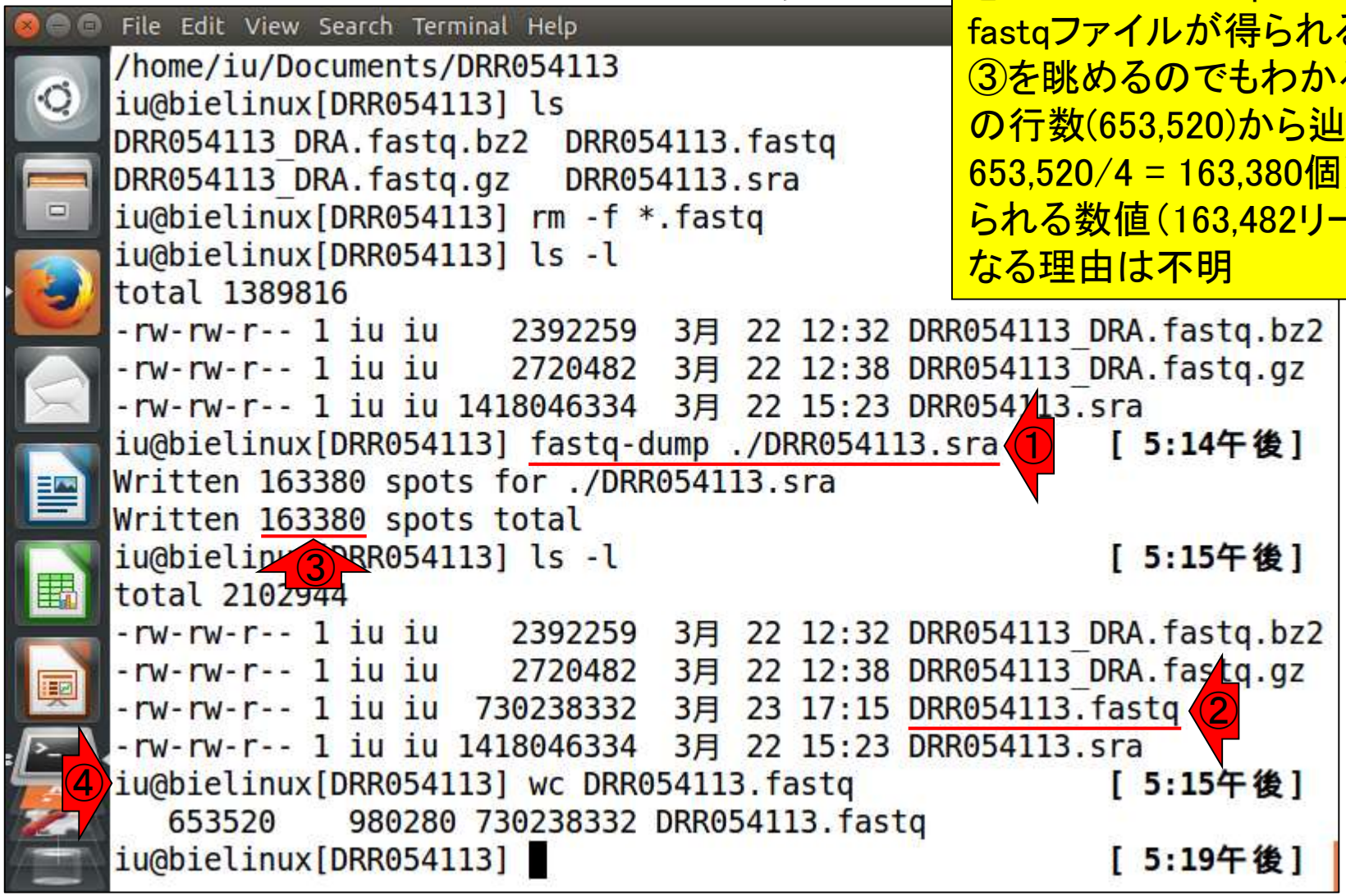
- `fastq-dump`
  NO: If the toolkit location is not specified in your $PATH variable, then the OS cannot locate the fastq-dump program, even if it is in the current directory. NOTE: Windows users should be able to enter only "fastq-dump.exe" if you have navigated to the Toolkit "bin" directory.

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

W5-6：デフォルトで実行

ここはダイジェスト版のみで眺めるだけ（詳細は第7回ウェブ資料PDF）。①sraファイルを入力としてfastq-dumpを実行すると、②fastqファイルが得られる。リード数情報は、③を眺めるのでもわかる。④fastqファイルの行数(653,520)から辿れるリード数（653,520/4 = 163,380個）と同じ。DRA上で見られる数値（163,482リード；W2-5）と若干異なる理由は不明



W6-3：結果を眺める

FastQC実行結果。①入力は、さきほどのgzip圧縮ファイル。②リード数は163,380、③配列長は116-28874 bpの範囲であることがわかる

FastQC Report

Thu 24 Mar 2016
DRR054113.fastq.gz

Summary

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

Basic Statistics

Measure	Value
Filename	DRR054113.fastq.gz ①
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	163380 ②
Sequences flagged as poor quality	0
Sequence length	116-28874 ③
%GC	44

W6-3：結果を眺める

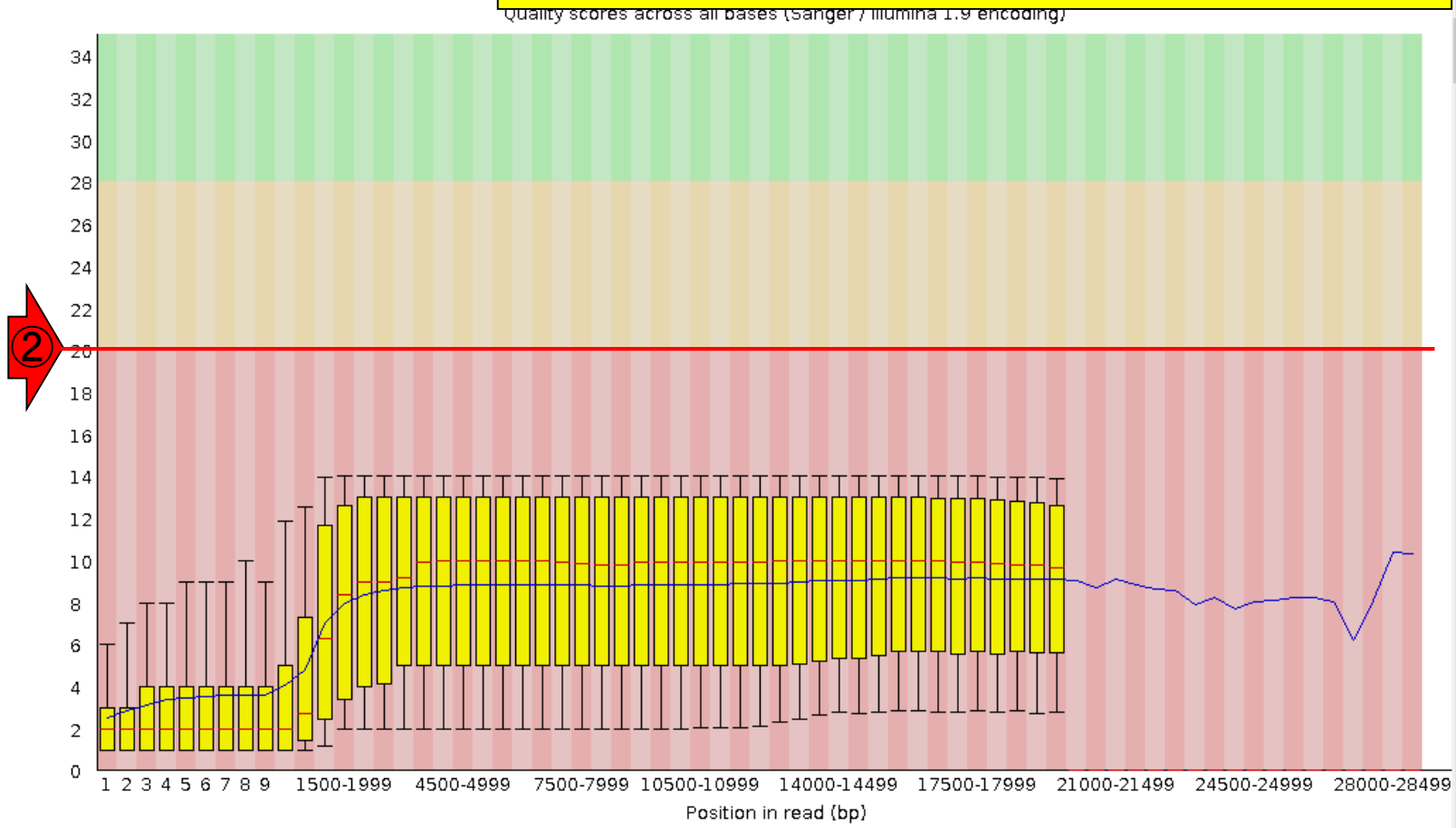
①Per base sequence quality。この図の縦軸はクオリティスコア。②赤線のスコア20を基準としてみると、DRAから直接ダウンロードした915リードからなるFASTQファイルのFastQC実行結果（W3-3）と比べて、明らかに低くなっていることがわかる。クオリティスコアは、平均で10弱程度

FastQC Report

Summary

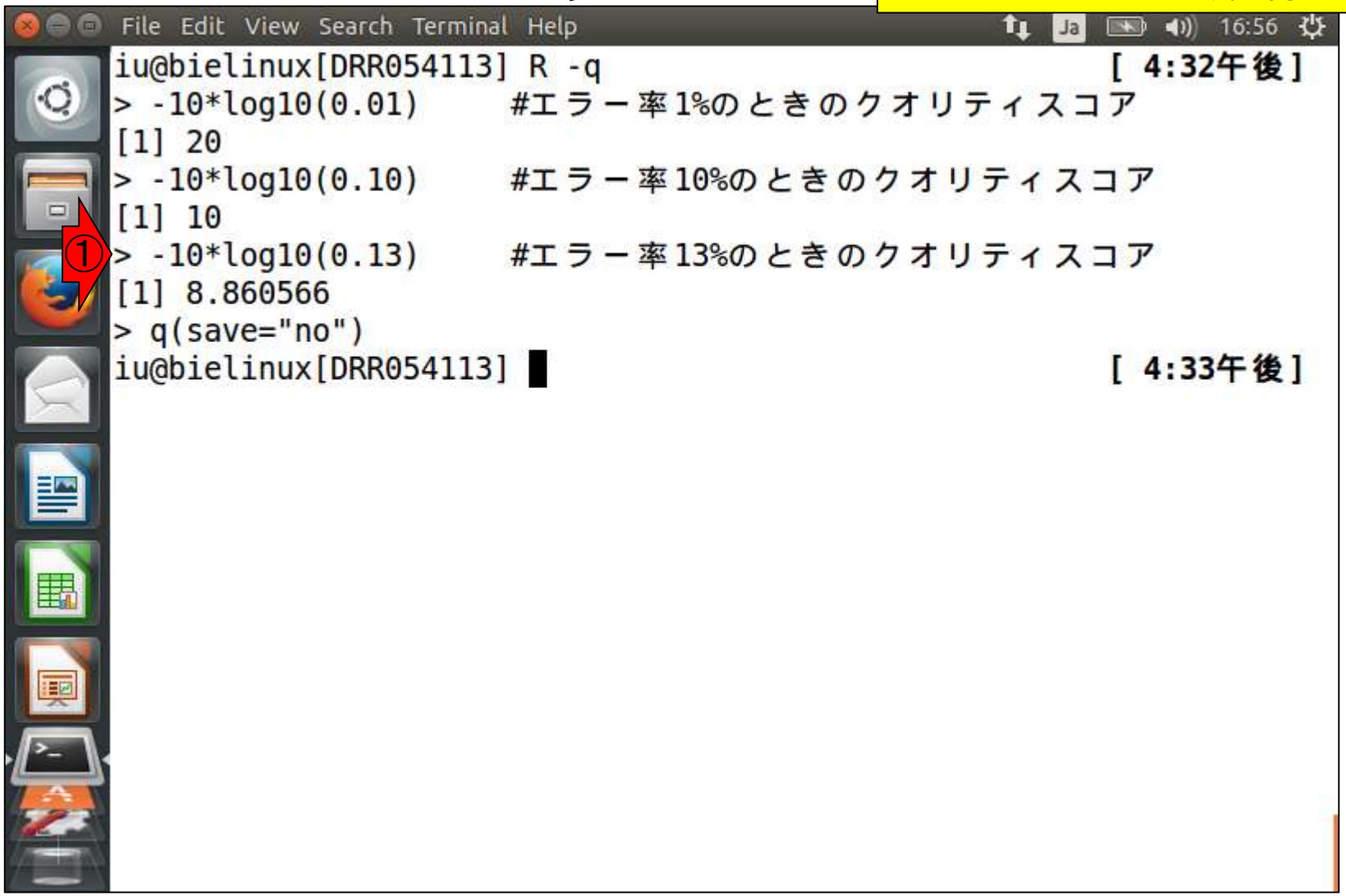
- Basic Statistics
- Per base sequence quality ①
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Per base sequence quality



W6-4：Rで計算

塩基配列決定精度（エラー率）からクオリティスコアをRで計算する。①エラー率13%のときは、スコア8.86となるので、実際のスペック通りで安心



①

Rhoads and Au, *Genomics Proteomics Bioinformatics*, **13**: 278-289, 2015

W6は

①のあたりです。②FastQC実行結果のhtmlファイルはここにあるので、じっくり眺めたいヒトはどうぞ。~/Desktop/backupにもあります

PacBioデータの概観2

- W6-1: FastQC (ver. 0.11.4)実行
  FastQC ver. 0.11.4は、第4回W9-2でインストールし、fastqc2というコマンドでパスを通している。

```
cd ~/Documents/DRR054113

pwd
ls -l
fastqc2 -v
fastqc2 -q DRR054113.fastq.gz
ls -l
```

- W6-2: 改名して移動
  共有フォルダ(~/Desktop/mac_share)内にW3-2で作成した同じファイル名のものが存在する。このため、mvコマンドで共有フォルダ(~/Desktop/mac_share)に移動させる際に、(163380リードからなるファイルの結果という意味で) _163380をファイル名に追加している。.zipファイルは実質的に不要ではあるが、一応同時に移動させている。

```
pwd
ls -l *fastqc*
ls -l ~/Desktop/mac_share
mv DRR054113_fastqc.html ~/Desktop/mac_share/DRR054113_163380_fastqc.html
mv DRR054113_fastqc.zip ~/Desktop/mac_share/DRR054113_163380_fastqc.zip
ls -l ~/Desktop/mac_share
```

- W6-3: 結果を眺める
  DRR054113_163380_fastqc.html
- W6-4: Rで計算
  塩基配列決定精度が87%(エラー率13%)のときのクオリティスコアは8.86。

```
R -q
-10*log10(0.01)      #エラー率1%のときのクオリティスコア
-10*log10(0.10)      #エラー率10%のときのクオリティスコア
-10*log10(0.13)      #エラー率13%のときのクオリティスコア
```

Contents

- W11：ゲノムサイズ推定（KmerGenieのインストールと利用）
  - インストール、single-endで実行（結果の解説）、paired-endで実行（結果の解説）
- W12：配列長によるフィルタリング（Pythonプログラムの利用）
- DDBJ Pipeline（W13からW17まではほぼ省略）
  - W18：k=131でのVelvet実行結果の解析
  - W19：Platanusの実行、W20：結果の解析、W21：ACGTカウント（塩基ごとの出現頻度解析）
- ロングリード(PacBio)データと公共DB
  - W2： PacBio生データはbax.h5形式、公共DBはsra形式とFASTQ形式
  - W3：公共DB (DRA)のFASTQファイルを入力としてFastQC
  - W4：NCBI SRA (SRA)が提供するSRA Toolkitのインストール
  - W5：利用(.sra → .fastqへの変換)、W6：FastQC
- DDBJ PipelineでHGAPを実行
  - W7：DDBJ Pipelineに解析したい生データファイル(.bax.h5)をアップロード
  - W8：DDBJ PipelineでHGAPを実行
  - W9：HGAPアセンブリ結果を眺める

W2-7 : bax.h5ファイル

おさらい。DDBJ Pipeline上では、PacBio用の *de novo*アセンブリ用プログラムHGAPを利用可能。しかし、①HGAPはbax.h5ファイルのみしか受け付けない

PacBioのファイル形式とデータ解析の概要

- W1-1: PacificBiosciencesの YouTubeサイト
  - Introduction to SMRT Sequencing
  - Single Molecule Real Time Sequencing
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - DRR024500: Tanizawa et al., BMC Genomics, 2015
- W2-3: DRR024501 -> DRP002401 -> DRX022185
- W2-4: DRR024501 -> DRA002643
- W2-5: PacBioデータ概観
  - DRR054113
  - DRR054114
  - DRR054115
  - DRR054116
- W2-6: SMRT Portal(PacBio提供のHGAPを含む解析ソフトウェア群)の場所
  - PacBio -> DevNet -> SMRT Analysis
  - SMRT Analysis 2.3までは、HGAPを実行するためにはbax.h5ファイルが必須。
  - SMRT Analysis 3.0からは、BAMファイルが入力フォーマットになる。但しここでのBAMファイルは、マッピングデータではなく、シークエンス生データ。
  - PacBio RSIIの後継機であるSequelの 出力ファイル形式はBAM。
  - PacBioのファイル形式の 説明についてはこちら (http://pacbiofileformats.readthedocs.io/en/3.0/)。
- W2-7: DRR054113のbax.h5ファイル (下記3ファイル合わせてDRR054113に相当)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5 (747 MB; 784,301,199 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5 (766 MB; 803,938,042 bytes)
  - m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5 (901 MB; 945,597,712 bytes)

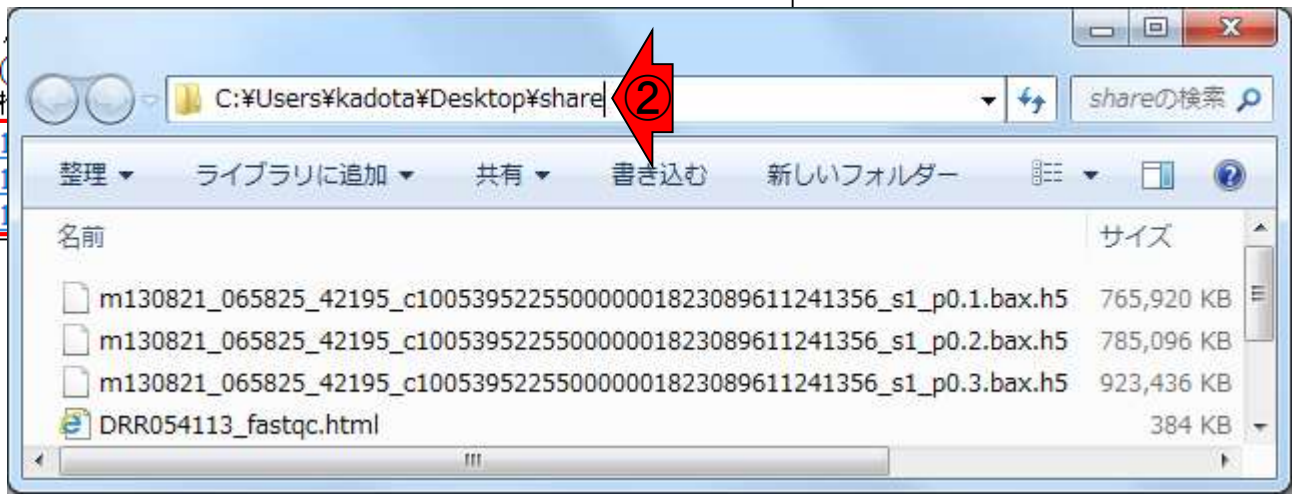
W7-1：bax.h5ファイル準備

①DDBJ Pipelineにアップロードしたいbax.h5ファイルを手元のPCにダウンロードしておく。②ここではDesktop上のshareフォルダにダウンロード。ファイルサイズは約2.4GBに達するため、それなりに時間はかかるだろう。もちろん講習会ではやりません！見るだけ！

PacBioのファイル形式とデータ解析の概要

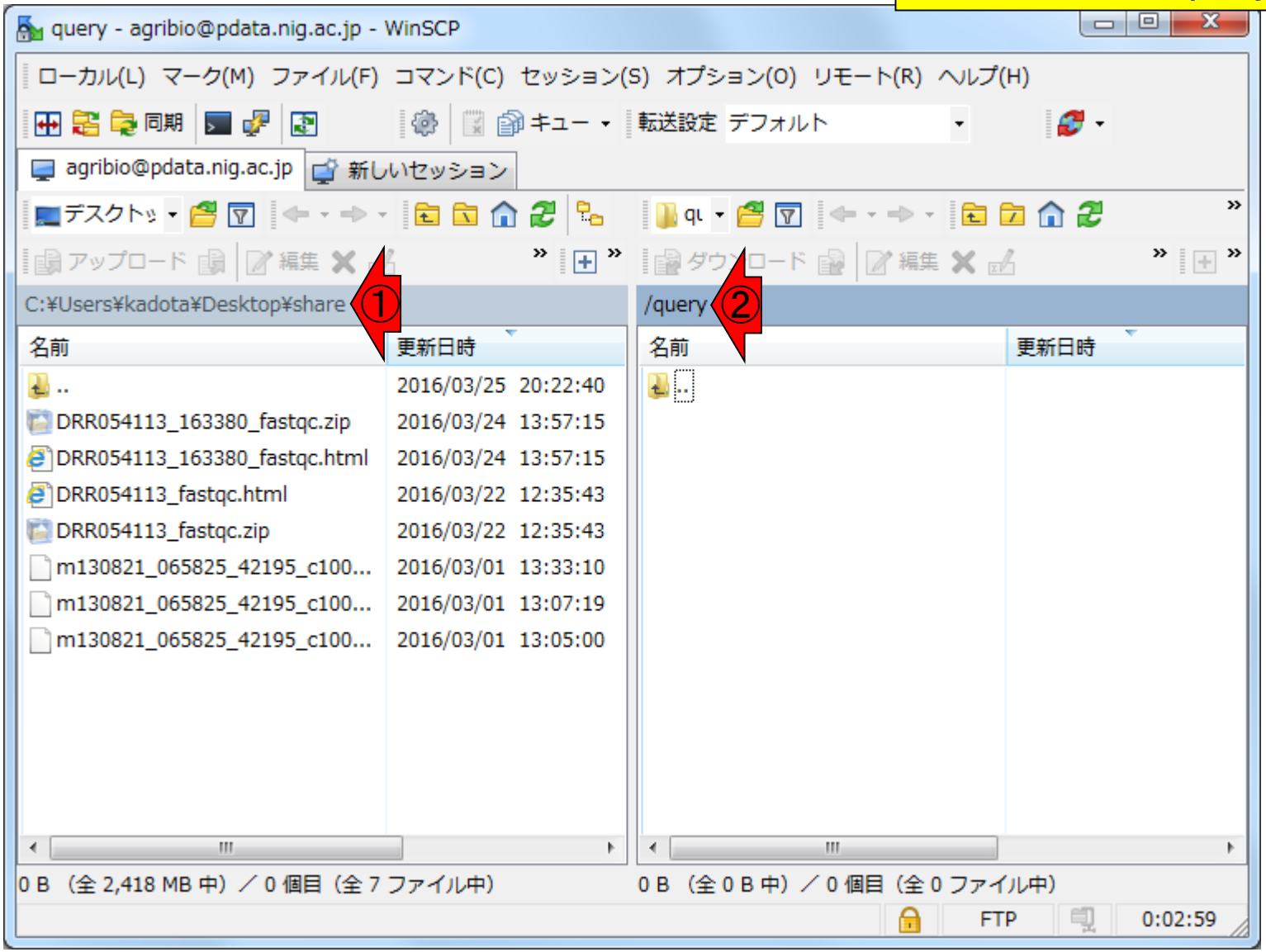
- W1-1: PacificBiosciencesの YouTubeサイト
  - Introduction to SMRT Sequencing
  - Single Molecule Real Time Sequencing
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - DRR024500: Tanizawa et al., BMC Genomics, 2015
- W2-3: DRR024501 -> DRP002401 -> DRX022185
- W2-4: DRR024501 -> DRA002643
- W2-5: PacBioデータ概観
  - DRR054113
  - DRR054114
  - DRR054115
  - DRR054116
- W2-6: SMRT Portal(PacBio提供のHGAPを含む解析ソフトウェア群)の場所
  - PacBio -> DevNet -> SMRT Analysis
  - SMRT Analysis 2.3までは、HGAPを実行するためにはbax.h5ファイルが必須。
  - SMRT Analysis 3.0からは、BAMファイルが入力フォーマットになる。但しここでのBAMファイルは、マッピングデータではなく、シークエンス生データ。
  - PacBio RSIIの後継機であるSequelの 出力ファイ
  - PacBioのファイル形式の説明についてはこちら (
- W2-7: DRR054113のbax.h5ファイル (下記3ファイル合
  - m130821_065825_42195_c100539522550000001
  - m130821_065825_42195_c100539522550000001
  - m130821_065825_42195_c100539522550000001

①



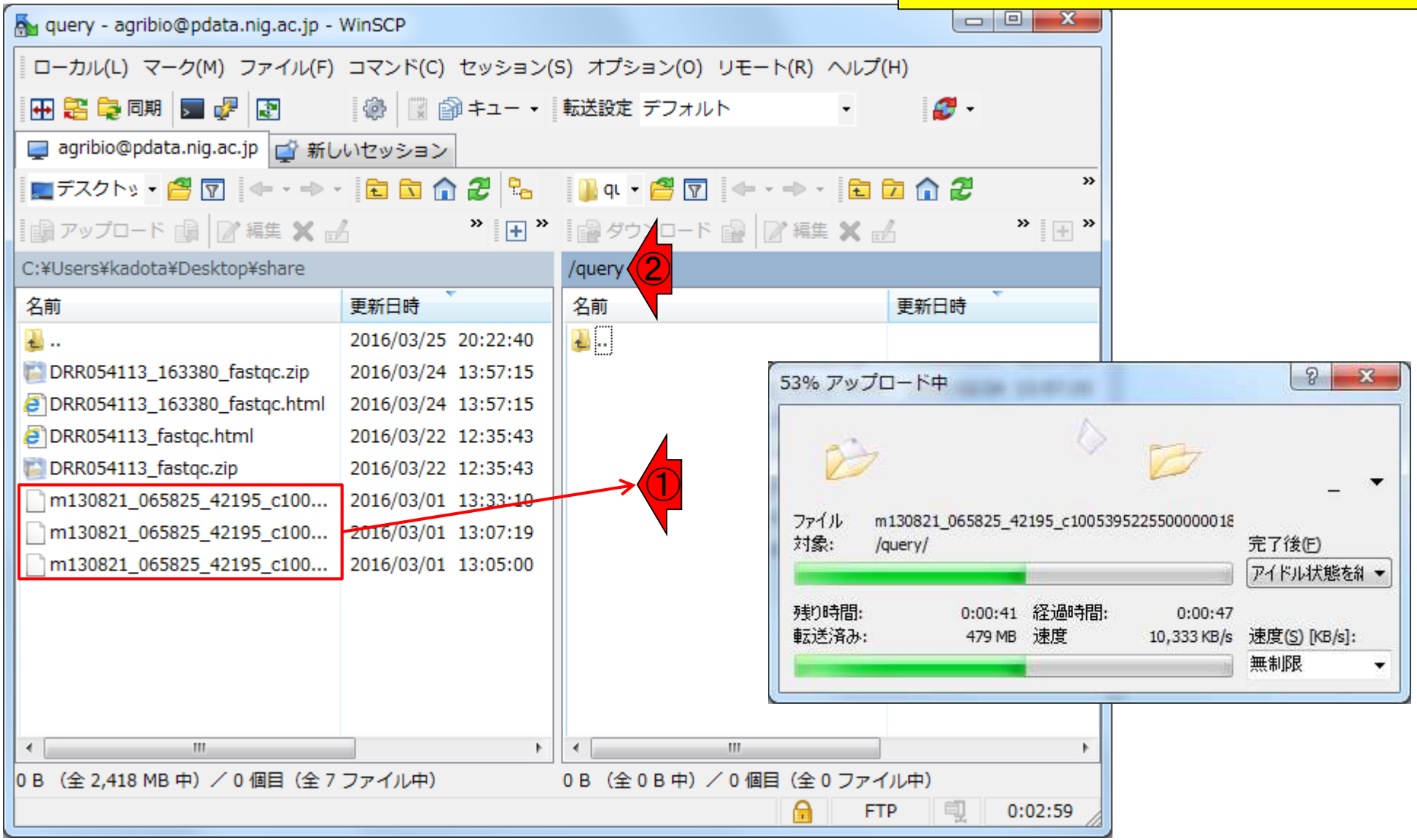
W7-2：アップロード

WinSCPを用いてDDBJ Pipelineの実体である遺伝研スパコンにログインし、①「デスクトップ – share」、②queryフォルダに移動した状態



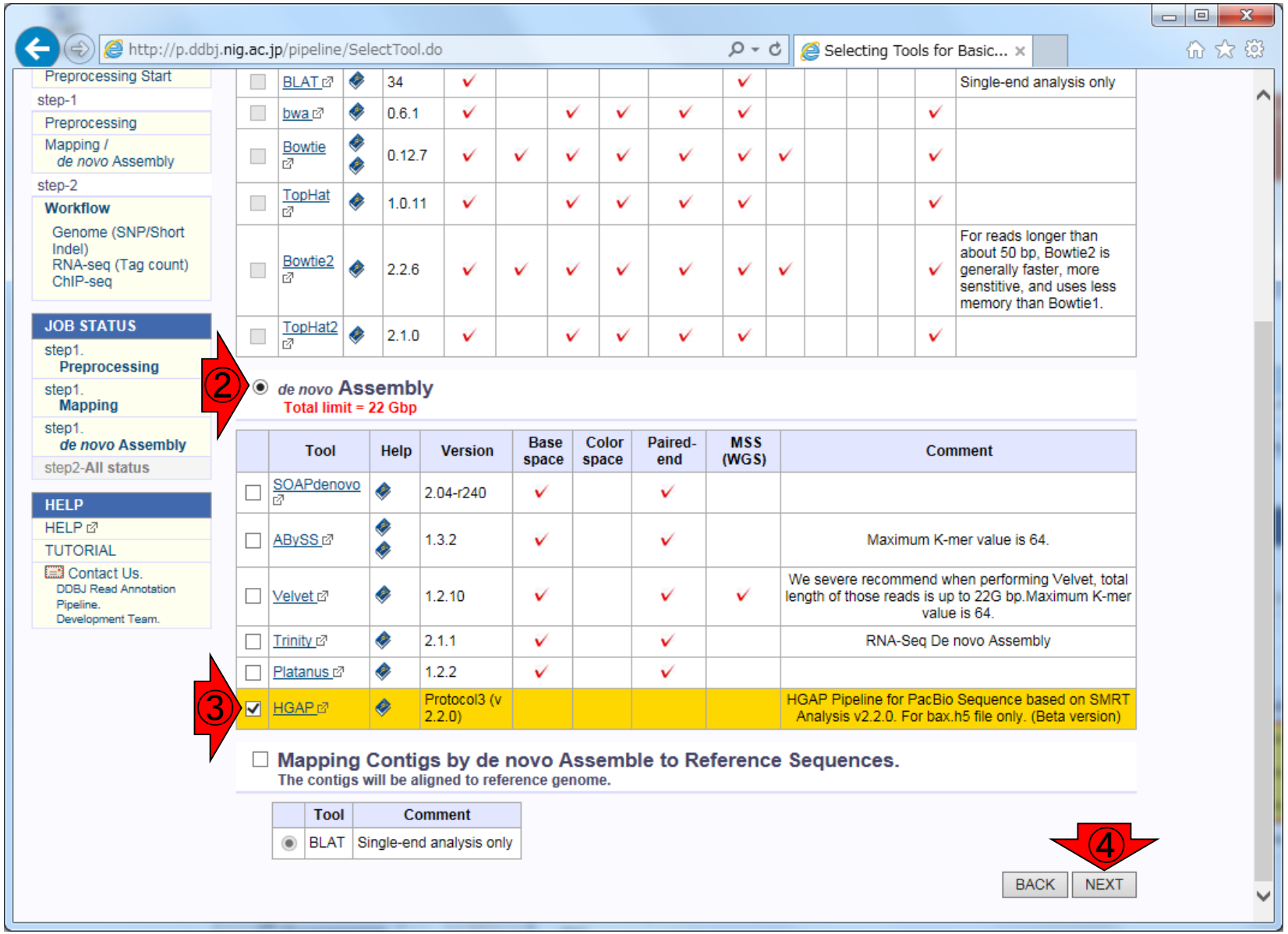
W7-2：アップロード

①アップロードしたい3つのファイルを
②queryフォルダにドラッグ&ドロップ。
アップロードもそれなりに時間がかかる



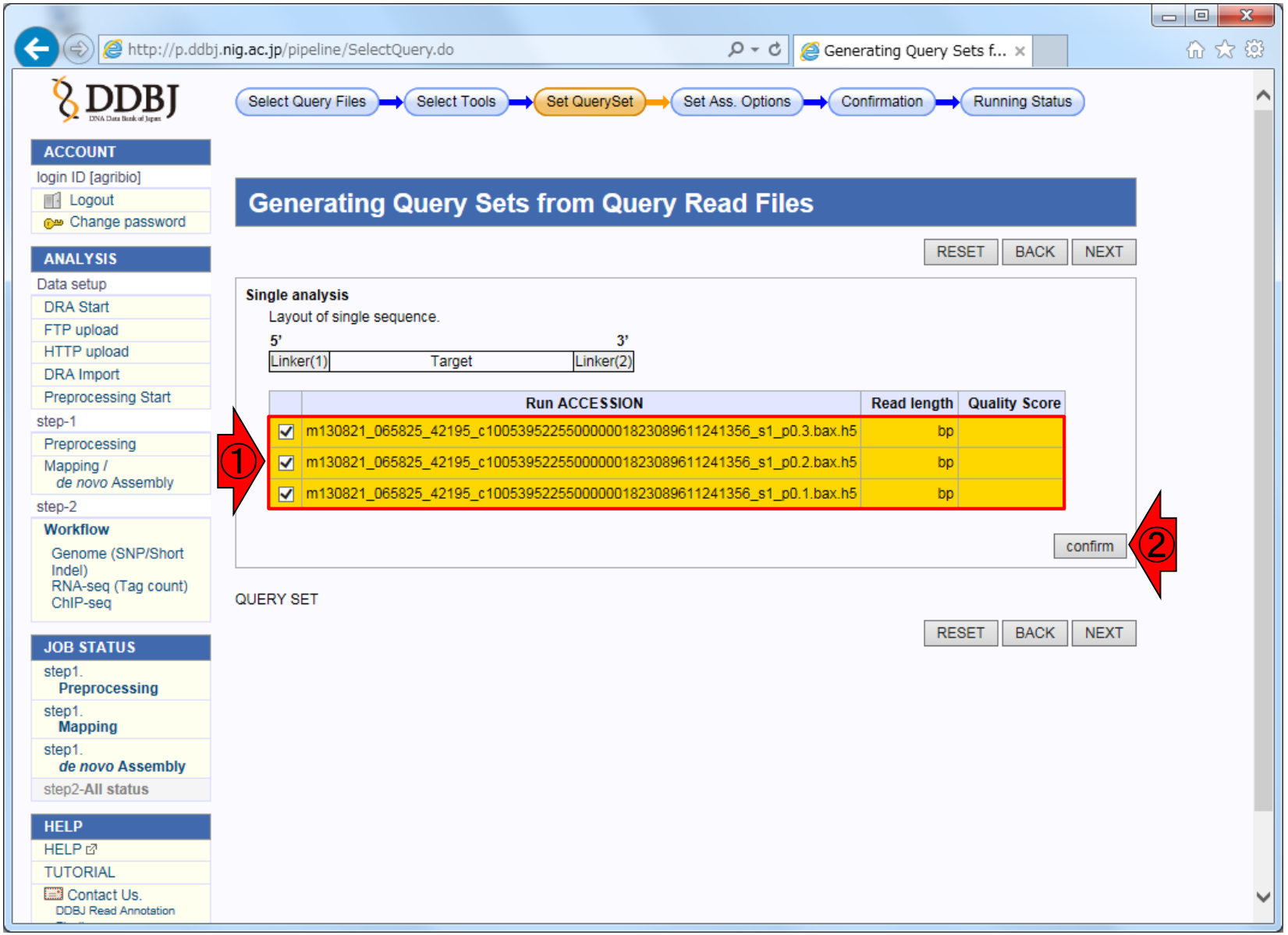
②*de novo* Assembly、③HGAP、④NEXT

W8-1：HGAP実行



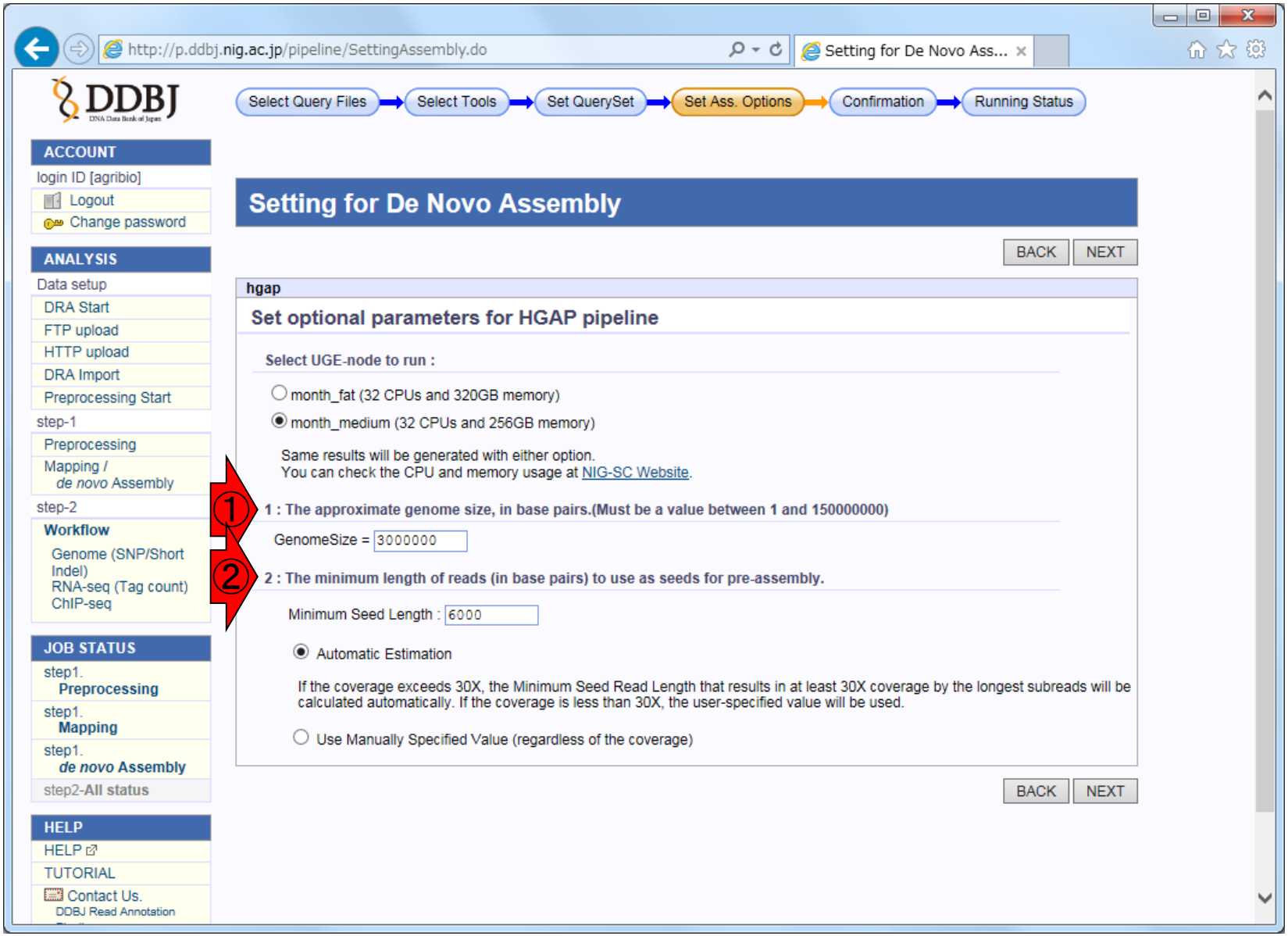
W8-1：HGAP実行

①解析したい3つのファイルにチェックを入れて、②confirm



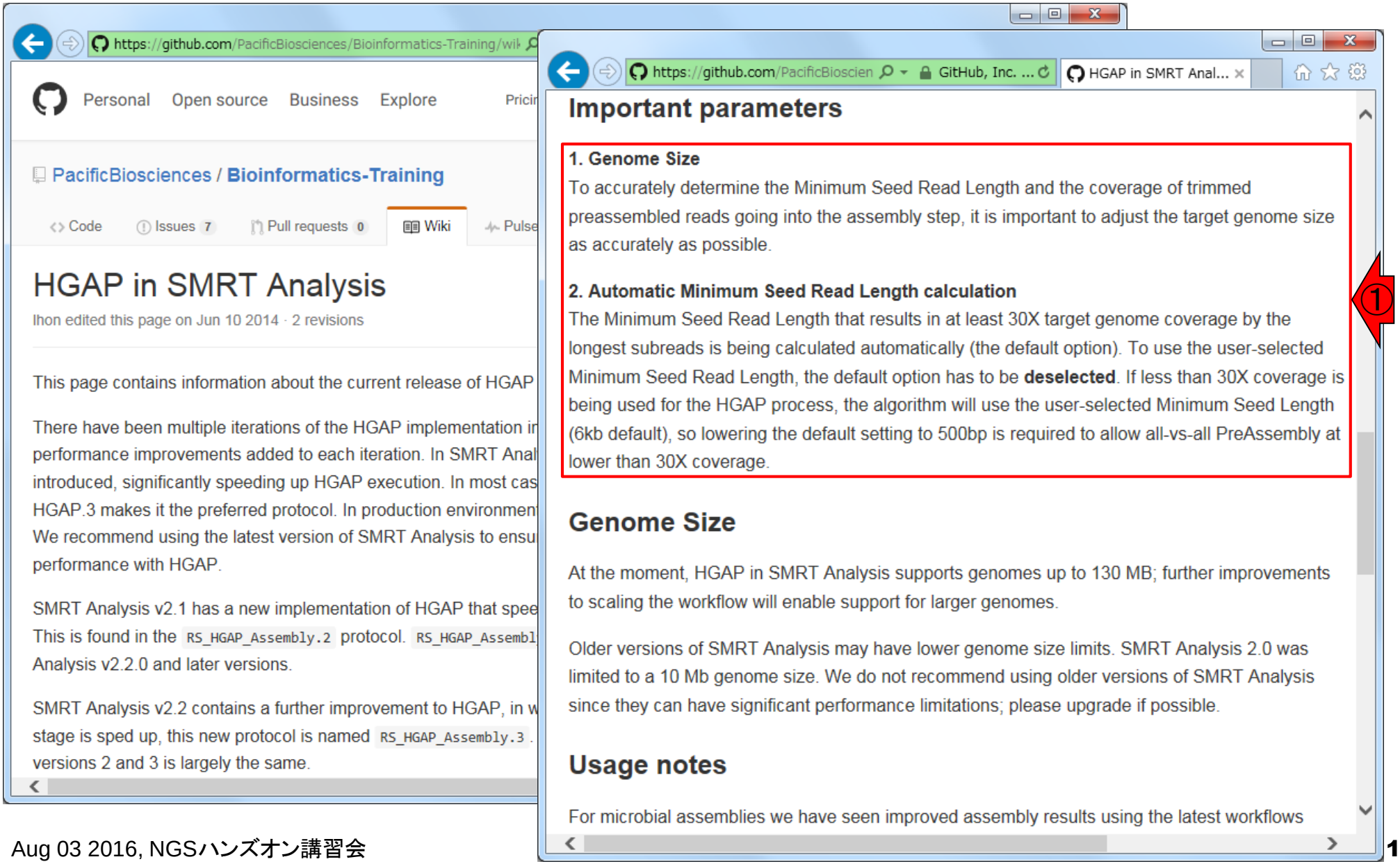
W8-2：2つのパラメータ

①と②、2つの指定可能なパラメータがあります



①2つのパラメータに関する解説

W8-2：パラメータの解説



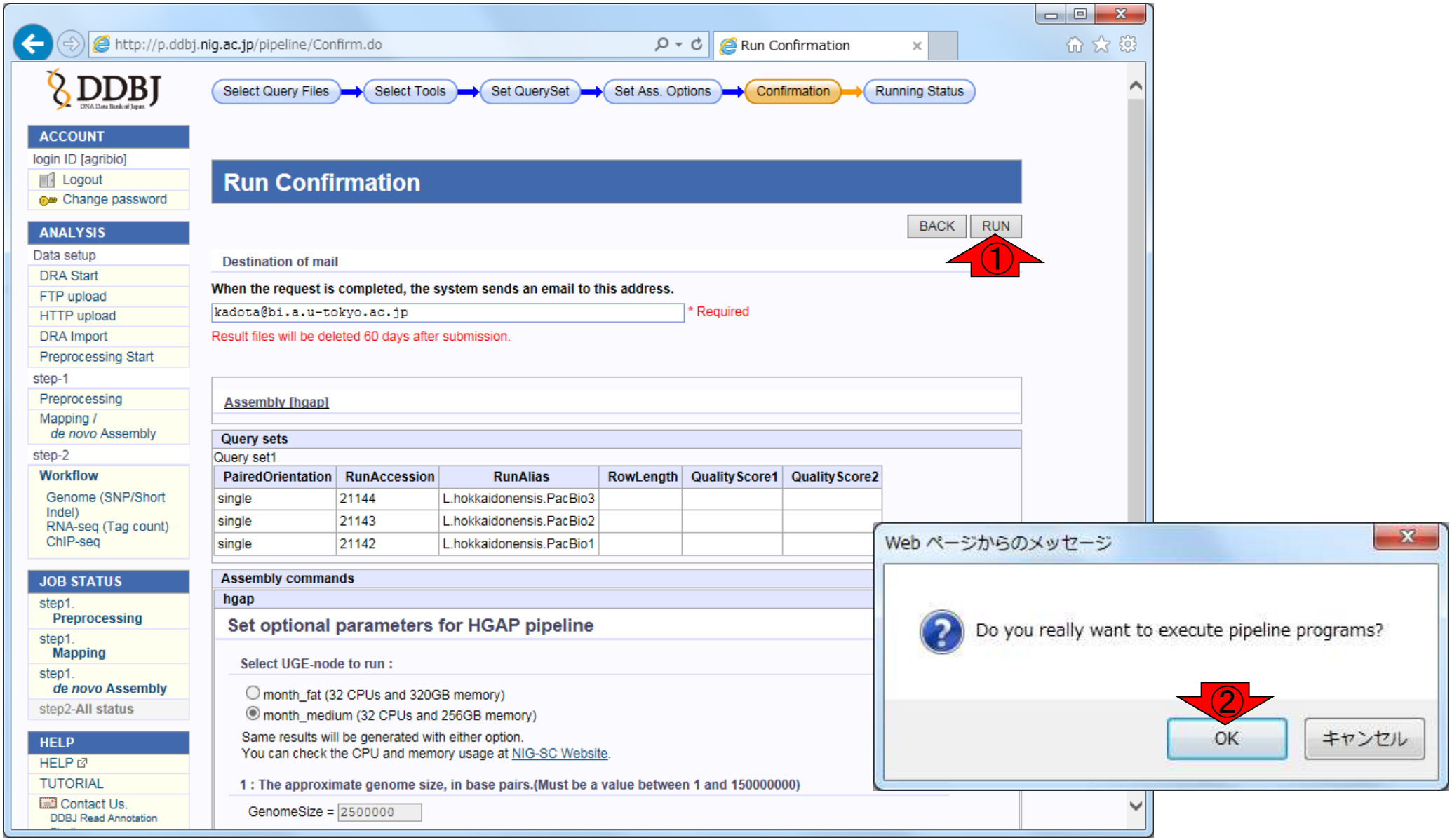
W8-3：HGAP実行

①ゲノムサイズは、乳酸菌の平均的なゲノムサイズである2.5MB。②minimum lengthは、Automatic Estimation（デフォルト）を指定して③NEXT

PDF原稿p104左下

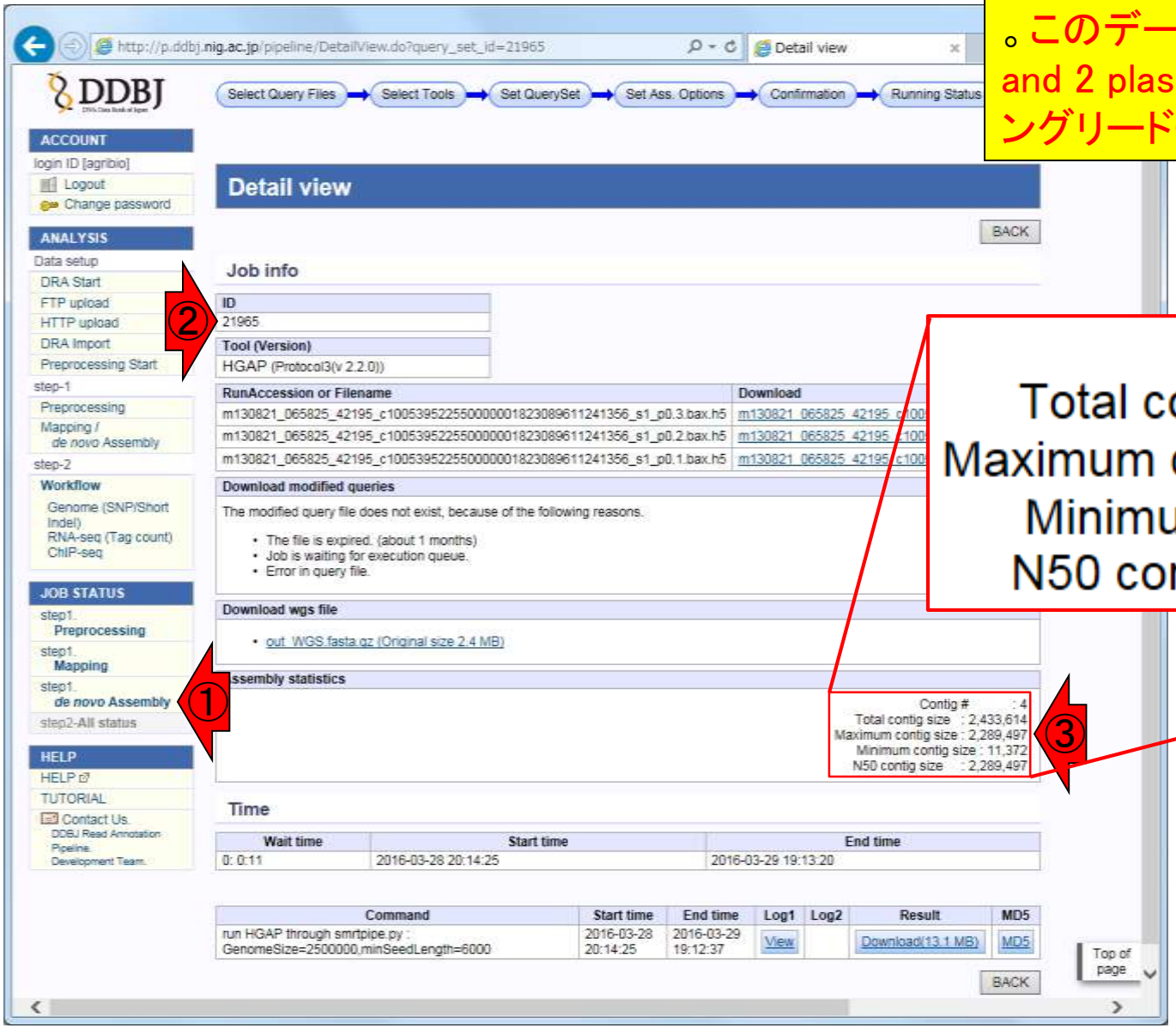
①RUN、②OK

W8-3：HGAP実行



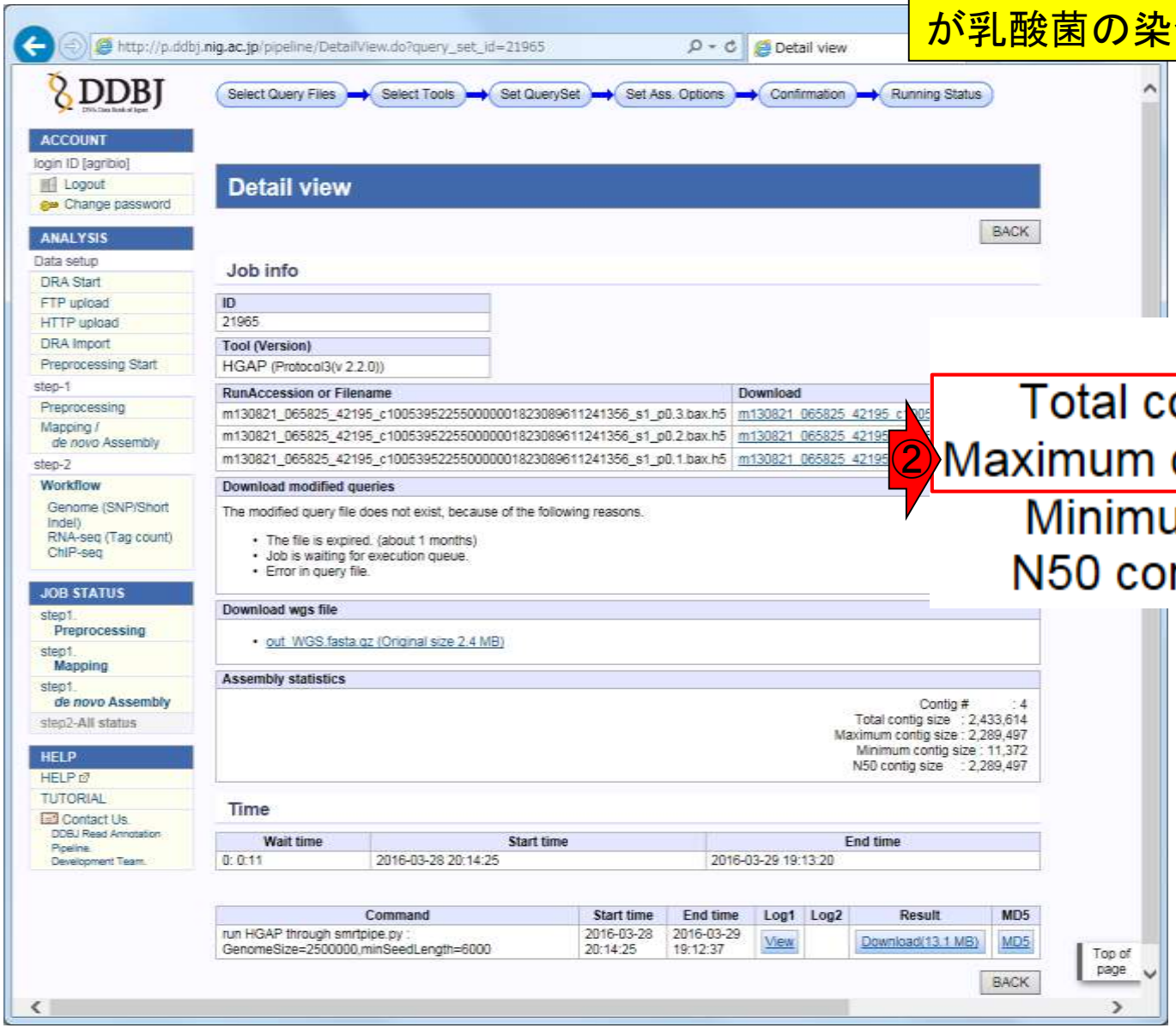
W9-2：結果を眺める

①*de novo* Assembly、②Job ID番号(21965)を頼りにすれば、このページに辿り着ける。③赤枠部分を見ると、④コンティグ数は4つ。このデータの正解は3つ（1 chromosome and 2 plasmids）。世間一般の評価通りのロングリード(PacBio)の長所がよく表れた結果



W9-2：結果を眺める

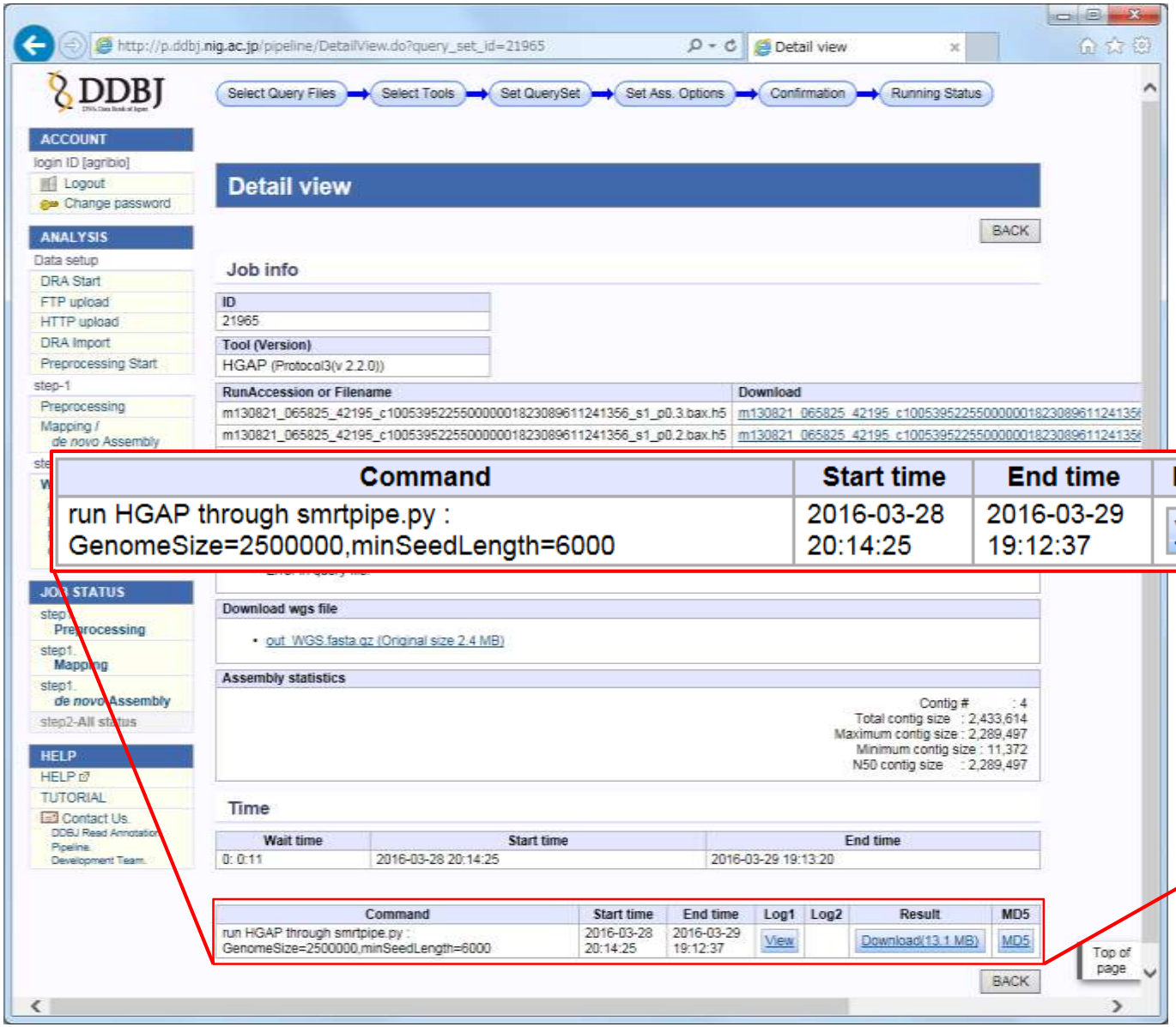
①Total contig sizeは、乳酸菌の一般的なゲノムサイズと近く妥当。②Maximum contig sizeのものが全体の9割以上を占めていることから、これが乳酸菌の染色体ゲノムなのだろうと妄想する



Contig # : 4
Total contig size : 2,433,614 ①
② Maximum contig size : 2,289,497
Minimum contig size : 11,372
N50 contig size : 2,289,497

W9-3：ダウンロード

①result.zipというzip圧縮ファイルをダウンロードして、その後の解析へと進んでいく



Command	Start time	End time	Log1	Log2	Result	MD5
run HGAP through smrtpipe.py : GenomeSize=2500000,minSeedLength=6000	2016-03-28 20:14:25	2016-03-29 19:12:37	View		Download(13.1 MB)	MD5

①

詳細は第7回原稿PDFとウェブ資料をご覧ください

この後の展開は…

- W10：multi-FASTAファイルの分割
  - プログラムによっては、single-FASTAのほうが取扱いやすい場合もある
- W11：FASTQファイルの分割とクオリティスコア分布
  - FASTAファイル用がうまく動かなくても、4行で1リードだということが分かっていれば、Linuxコマンドでどうにかなる、という話
  - PacBioはFASTQファイルも出力する。Rで(single-)FASTQファイルを読み込んでクオリティスコア分布を描画し、PacBioデータは両末端のクオリティが低い傾向にあることを確認
- 環状化（ゲノム解読のfinishing作業の一部）
  - アセンブリ結果として、最初と最後の末端部分が同じ配列の場合は、通常そのコンティグは環状と判断。それを確認するための基本的な考え方、手段、および環状化のノウハウを伝授
  - W12：seqinrパッケージを用いて、仮想環状コンティグのドットプロットで感覚をつかむ
  - W13：重複配列の除去の感覚をつかむ（これが環状化作業の実体）
  - W14：dotterプログラムで実際のPacBio出力結果に対して適用し、環状状態の概要を知る
  - W15：Bio-Linux上のblastnで、環状の同一コンティグ同士をDB側とquery側にして実行
  - W16：正確なアラインメントを眺め、切断箇所をクオリティスコア分布と合わせて判断する
  - W17：両端切断後のコンティグに対し、クオリティスコア分布やdotterを再度実行して確認
  - W18：NCBI blastのやり方も示し、様々な解析手段を伝授