

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、疲れたところに初耳状態で聞くと相当疲弊すると思われる。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](mailto:kadota@iu.a.u-tokyo.ac.jp)

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



# 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

NCBI Resources How To Sign in to NCBI

PubMed US National Library of Medicine National Institutes of Health

Advanced Help

Abstract Send to: Full text links

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

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

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

Author information

**Abstract**

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

**RESULTS:** The genome **①** 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] PMID: PMC4377027 Free PMC Article

Full text links: Read free full text at BioMed Central, PMC Full text

Save items: Add to Favorites

Similar articles: *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]

Related information

②

# おさらい

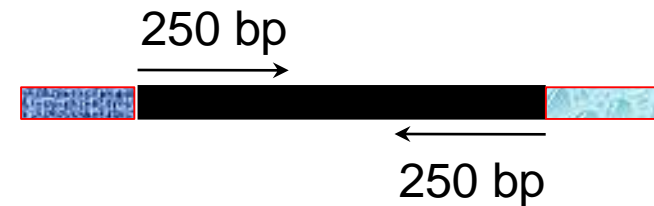
①乳酸菌ゲノム配列決定論文は、2種類のNGS機器から得られたデータを併用している。第6回はIllumina MiSeqデータを(DRR024501)を、そして第7回はPacBioデータを取り扱っている

## ■ 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(DRR024501 sub\_1.fastq.gz)
- reverse側(DRR024501 sub\_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





# 第6回原稿PDFのp45

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…

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<sup>4)</sup>。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



# 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)
- 롱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

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

- 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 modified 2016/06/13)
- 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)

## 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(追加)。

- [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)追加資料
  - [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\[W11-2\]](#)

```
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

## 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

Download KmerGenie sources here: [kmergenie-1.6982.tar.gz](#)  
You will need Python and R.

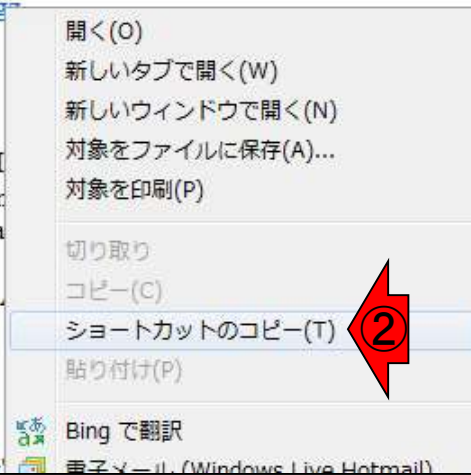
Latest [README](#) and [CHANGELOG](#). Major changes since

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

## Support

Please use [Biostars](#) (dynamic FAQ system, click "New Post")

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



for small

tion. Histograms

ny question. (An

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

# W11-2: ダウンロードと解凍

```
File Edit View Search Terminal Help
iu@bielinux[result] cd ~/Downloads [ 8:06午後]
iu@bielinux[Downloads] ls [ 8:06午後]
FaQCs IGV_2.3.67.zip velvet_1.2.10
FastQC nohup.out velvet_1.2.10.tgz
fastqc_v0.11.4.zip Rockhopper.jar
IGV_2.3.67 Rockhopper_Results
① iu@bielinux[Downloads] wget -cq http://kmergenie.bx.psu.edu/kmer
genie-1.6982.tar.gz
② iu@bielinux[Downloads] ls -l kmer* [ 8:06午後]
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads]
```

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w  
get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

```
iu@bielinux[Downloads] cd ~/Downloads [ 3:54午後 ]
iu@bielinux[Downloads] ls [ 3:55午後 ]
boost_1_61_0.tar.bz2 master.zip
Bridger_r2014-12-01.tar.gz nohup.out
FaQCs Rockhopper.jar
FastQC Rockhopper_Results
fastqc_v0.11.4.zip sratoolkit.2.6.3-ubuntu64.tar.gz
IGV_2.3.67 v2.2.0.tar.gz
IGV_2.3.67.zip velvet_1.2.10
kmergenie-1.6982.tar.gz velvet_1.2.10.tgz
iu@bielinux[Downloads] ls -l kmer* [ 3:55午後 ]
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads] tar -zxvf kmergenie-1.6982.tar.gz
```



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

# W11-2: ダウンロードと解凍

```
File Edit View Search Terminal Help
kmergenie-1.6982/scripts/generate_report.pyc
kmergenie-1.6982/scripts/est-mean.r
kmergenie-1.6982/scripts/decide
kmergenie-1.6982/scripts/model-diploid.r
kmergenie-1.6982/scripts/zeta.r
kmergenie-1.6982/scripts/plot_histogram.r
kmergenie-1.6982/scripts/cutoff.r
kmergenie-1.6982/scripts/est-params.r
kmergenie-1.6982/scripts/fit-histogram.r
kmergenie-1.6982/scripts/model.r
kmergenie-1.6982/scripts/plot_genomic_kmers.r
kmergenie-1.6982/scripts/generate_report.py
kmergenie-1.6982/LICENSE
kmergenie-1.6982/__init__.py
kmergenie-1.6982/main.cpp
kmergenie-1.6982/kmergenie
iu@bielinux[Downloads] ls -ld kmer*           [ 8:10午後 ]
drwxr-xr-x 5 iu iu 4096 6月 18 2015 kmergenie-1.6982
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads] █                     [ 8:10午後 ]
```



# 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)

```
iu@bielinux[Downloads] pwd [ 8:23午後 ]
/home/iu/Downloads
iu@bielinux[Downloads] ls -ld kmer* [ 8:24午後 ]
drwxr-xr-x 5 iu iu 4096 6月 18 2015 kmergenie-1.6982
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads] cd kmergenie-1.6982 [ 8:24午後 ]
iu@bielinux[kmergenie-1.6982] pwd [ 8:24午後 ]
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls [ 8:24午後 ]
CHANGELOG LICENSE minia third_party
__init__.py main.cpp README wrapper.py
kmergenie makefile scripts
iu@bielinux[kmergenie-1.6982] less README [ 8:24午後 ]
```





# W11-3: README

```
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
```



# 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

```
File Edit View Search Terminal Help
summary of the results, and contains an Advanced
elp analysis.

* There is no need to use Kmergenie for a multi-k assembler,
like SPAdes. Default parameters of multi-k assemblers are gener-
ally better than a single best k.

* To support even larger values of k, e.g. up to 200, type `
make clean ; make k=200`. By default, Kmergenie is compiled to s-
upport values of k up to 121.

* By default, KmerGenie will perform another pass to estimat-
e k more precisely. You may skip it by using the "--one-pass" op-
tion (roughly 2x faster).

* To run multiple instances of KmerGenie on the same folder,
specify the "-o" and "-t" parameters (output prefix, number of
threads per instance).

:█
```



まずは①make clean。実行前後で特に変わった様子はないようだ

# W11-4: make clean

```

iu@bielinux[Downloads] pwd [ 8:23午後]
/home/iu/Downloads
iu@bielinux[Downloads] ls -ld kmer* [ 8:24午後]
drwxr-xr-x 5 iu iu 4096 6月 18 2015 kmergenie-1.6982
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads] cd kmergenie-1.6982 [ 8:24午後]
iu@bielinux[kmergenie-1.6982] pwd [ 8:24午後]
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls [ 8:24午後]
CHANGELOG LICENSE minia third_party
__init__.py main.cpp README wrapper.py
kmergenie makefile scripts
iu@bielinux[kmergenie-1.6982] less README [ 8:24午後]
iu@bielinux[kmergenie-1.6982] make clean [ 8:38午後]
rm -rf *.o minia/*.o
iu@bielinux[kmergenie-1.6982] ls [ 8:38午後]
CHANGELOG LICENSE minia third_party
__init__.py main.cpp README wrapper.py
kmergenie makefile scripts
iu@bielinux[kmergenie-1.6982] [ 8:39午後]

```

①



# W11-5: make k=200

```

iu@bielinux[Downloads] pwd [ 8:23午後]
/home/iu/Downloads
iu@bielinux[Downloads] ls -ld kmer* [ 8:24午後]
drwxr-xr-x 5 iu iu 4096 6月 18 2015 kmergenie-1.6982
-rw-rw-r-- 1 iu iu 423235 6月 18 2015 kmergenie-1.6982.tar.gz
iu@bielinux[Downloads] cd kmergenie-1.6982 [ 8:24午後]
iu@bielinux[kmergenie-1.6982] pwd [ 8:24午後]
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls [ 8:24午後]
CHANGELOG LICENSE minia third_party
__init__.py main.cpp README wrapper.py
kmergenie makefile scripts
iu@bielinux[kmergenie-1.6982] less README [ 8:24午後]
iu@bielinux[kmergenie-1.6982] make clean [ 8:38午後]
rm -rf *.o minia/*.o
iu@bielinux[kmergenie-1.6982] ls [ 8:38午後]
CHANGELOG LICENSE minia third_party
__init__.py main.cpp README wrapper.py
kmergenie makefile scripts
iu@bielinux[kmergenie-1.6982] make k=200 [ 8:39午後]

```



# W11-5: make k=200

```
File Edit View Search Terminal Help
geint -DKMER_PRECISION=7
g++ -o specialk minia/Pool.o minia/Bank.o minia/Hash16.o minia/Bloom.o minia/Kmer.o minia/Utils.o minia/LinearCounter.o minia/LargeInt.o minia/MultiConsumer.o minia/Hashing.o main.cpp -O4 -pth read -D_largeint -DKMER_PRECISION=7 -lz -DSVN_REV=1.6982
scripts/test_install
Testing presence of specialk....
OK
Testing presence of Rscript....
R scripting front-end version 3.2.0 (2015-04-16)
OK
Testing basic Rscript functionality....
Rscript --no-init-file -e 'rnorm(1)'
[1] "rnorm(1)"
OK
Testing a simple KmerGenie example....
OK
Test successful. KmerGenie is ready, type `./kmergenie`.
iu@bielinux[kmergenie-1.6982] [ 8:44午後]
```





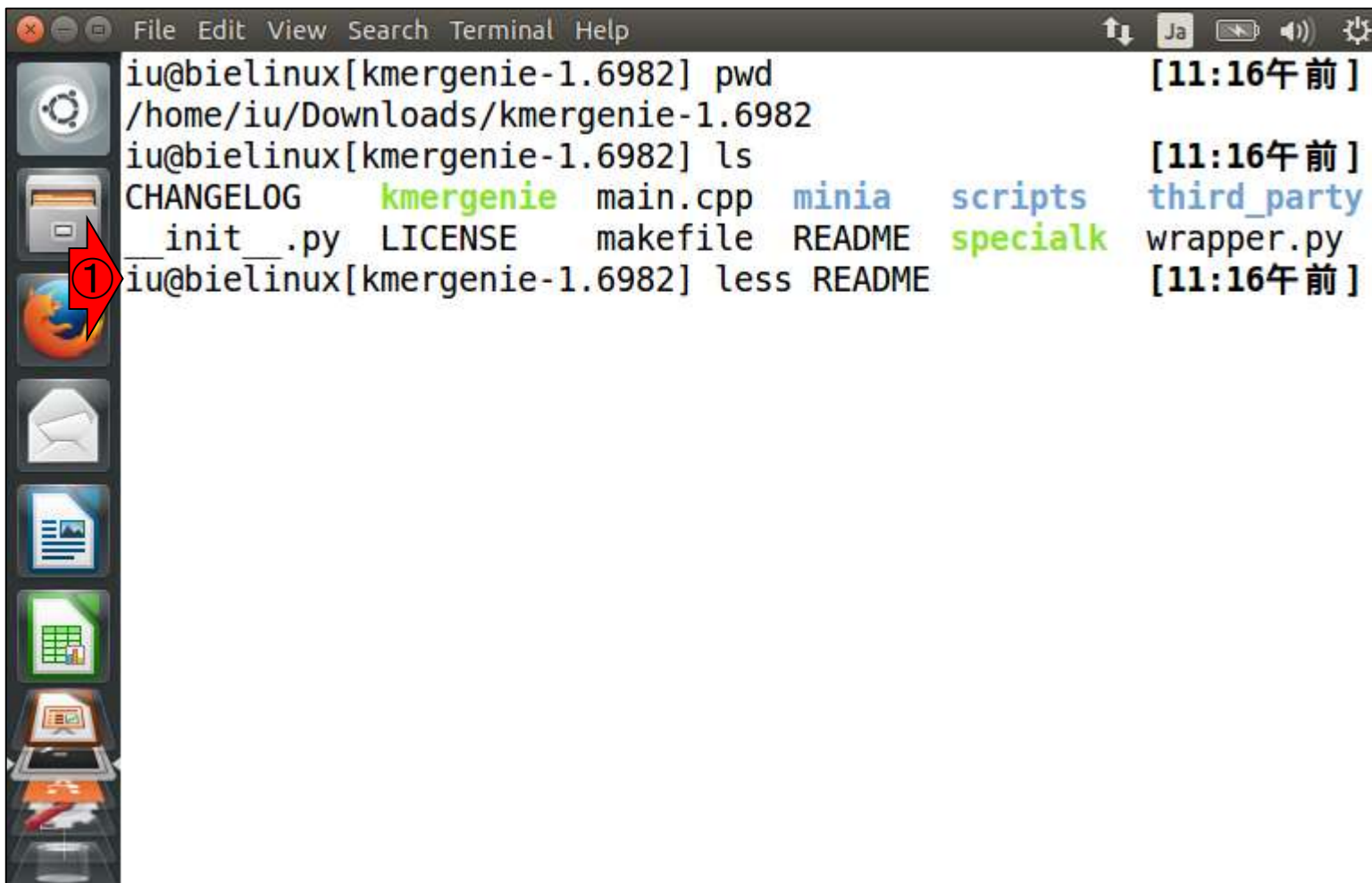
# W11-5: make k=200

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。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

```
File Edit View Search Terminal Help
OK
Test successful. KmerGenie is ready, type `./kmer
iu@bielinux[kmergenie-1.6982] ls
CHANGELOG      kmergenie  main.cpp  minia  scripts  third_party
__init__.py  LICENSE    makefile  README specialk  wrapper.py
iu@bielinux[kmergenie-1.6982] ls -l
[ 8:45午後]
total 200
-rw-r--r-- 1 iu iu   7511  6月 18  2015 CHANGELOG
-rw-r--r-- 1 iu iu     0  6月 18  2015 __init__.py
-rwxr-xr-x 1 iu iu  4401  6月 18  2015 kmergenie
-rw-r--r-- 1 iu iu 22745  6月 18  2015 LICENSE
-rw-r--r-- 1 iu iu 10753  6月 18  2015 main.cpp
-rw-r--r-- 1 iu iu  2432  6月 18  2015 makefile
drwxr-xr-x 2 iu iu  4096  1月  5  20:44 minia
-rw-r--r-- 1 iu iu  3220  6月 18  2015 README
drwxr-xr-x 2 iu iu  4096  6月 18  2015 scripts
-rwxrwxr-x 1 iu iu 122924  1月  5  20:44 specialk
drwxr-xr-x 2 iu iu  4096  6月 18  2015 third_party
-rw-r--r-- 1 iu iu  1945  6月 18  2015 wrapper.py
iu@bielinux[kmergenie-1.6982] [ 8:45午後]
```



# W11-6: README



```
iu@bielinux[kmergenie-1.6982] pwd [11:16午前]
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls [11:16午前]
CHANGELOG kmergenie main.cpp minia scripts third_party
__init__.py LICENSE makefile README specialk wrapper.py
iu@bielinux[kmergenie-1.6982] less README [11:16午前]
```

# 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
```





# W11-7: make install

(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



```
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
iu@bielinux[kmergenie-1.6982] make install
rm -f /usr/local/bin/kmergenie && ln -s `pwd`/kmergenie /usr/local/bin
ln: failed to create symbolic link '/usr/local/bin/kmergenie': Permission denied
make: *** [install] Error 1
iu@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
iu@bielinux[kmergenie-1.6982] whoami [12:09午後]
iu
iu@bielinux[kmergenie-1.6982] [12:09午後]
```





# 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

```
iu@bielinux[kmergenie-1.6982] pwd
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls
CHANGELOG  kmergenie  main.cpp  minia  scri
__init__.py LICENSE  makefile  README  spe
iu@bielinux[kmergenie-1.6982] kmergenie -h
zsh: command not found: kmergenie
iu@bielinux[kmergenie-1.6982] sudo make install
[sudo] password for iu:
rm -f /usr/local/bin/kmergenie && ln -s `pwd`/kmergenie /usr/local/bin
Kmergenie is installed, via a symlink to /usr/local/bin (do not delete the contents of /home/iu/Downloads/kmergenie-1.6982)
iu@bielinux[kmergenie-1.6982]
```



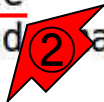
# W11-7: make install

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

```
iu@bielinux[kmergenie-1.6982] pwd
/home/iu/Downloads/kmergenie-1.6982
iu@bielinux[kmergenie-1.6982] ls
CHANGELOG      kmergenie  main.cpp  minia  scripts  third_party
__init__.py  LICENSE   makefile  README specialk  wrapper.py
iu@bielinux[kmergenie-1.6982] kmergenie -h
zsh: command not found: kmergenie
iu@bielinux[kmergenie-1.6982] sudo make install
[sudo] password for iu:
rm -f /usr/local/bin/kmergenie && ln -s `pwd`/kmergenie /usr/local/bin
Kmergenie is installed, via a symlink to /usr/local/bin (do not delete the contents of /home/iu/Downloads/kmergenie-1.6982)
iu@bielinux[kmergenie-1.6982] kmergenie -h
KmerGenie

Usage:
  kmergenie <read_file> [options]

Options:
```





# 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

```
File Edit View Search Terminal Help
iu@bielinux[kmergenie-1.6982] kmergenie -h
KmerGenie

Usage:
  kmergenie <read_file> [options]

Options:
  --diploid      use the diploid model (default: haploid model)
  --one-pass     skip the second pass to estimate k at 2 bp resolution (default: two passes)
  -k <value>    largest k-mer size to consider (default: 121)
  -l <value>    smallest k-mer size to consider (default: 15)
  -s <value>    interval between consecutive kmer sizes (default: 10)
  -e <value>    k-mer sampling value (default: auto-detected to use ~200 MB memory/thread)
  -t <value>    number of threads (default: number of cores minus one)
  -o <prefix>   prefix of the output files (default: histograms)
iu@bielinux[kmergenie-1.6982] [ 1:06午後 ]
```



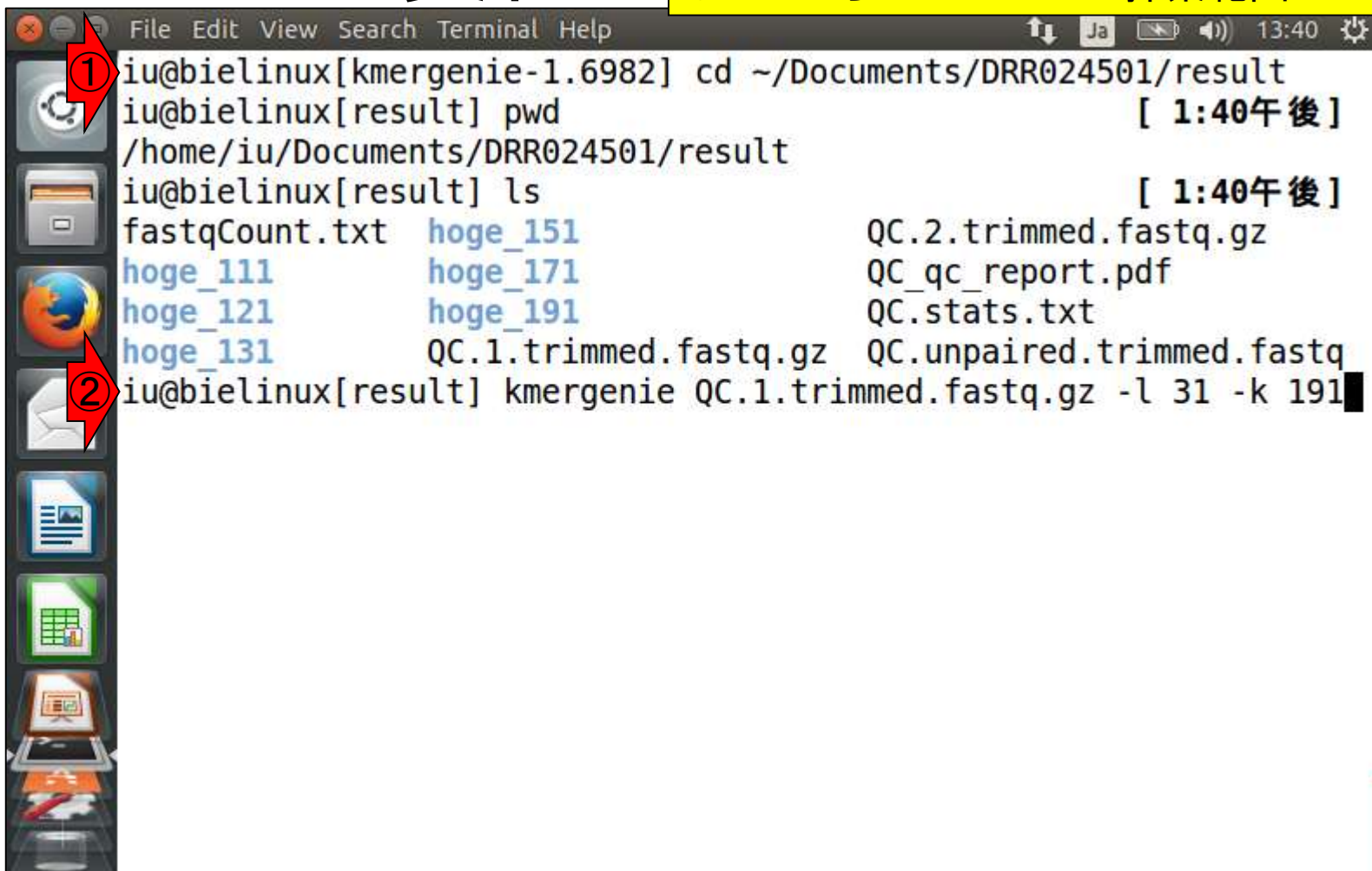
# Contents

- 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)
- 롱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



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: 実行



```
File Edit View Search Terminal Help 13:40
iu@bielinux[kmergenie-1.6982] cd ~/Documents/DRR024501/result
iu@bielinux[result] pwd [ 1:40午後 ]
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls [ 1:40午後 ]
fastqCount.txt hoge_151 QC.2.trimmed.fastq.gz
hoge_111 hoge_171 QC_qc_report.pdf
hoge_121 hoge_191 QC.stats.txt
hoge_131 QC.1.trimmed.fastq.gz QC.unpaired.trimmed.fastq
iu@bielinux[result] kmergenie QC.1.trimmed.fastq.gz -l 31 -k 191
```



# 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

```
iu@bielinux[kmergenie-1.6982] cd ~/Documents/DRR024501/result
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls
fastqCount.txt  hoge_151  QC.2.trimmed.fastq.gz
hoge_111        hoge_171  QC_qc_report.pdf
hoge_121        hoge_191  QC.stats.txt
hoge_131        QC.1.trimmed.fastq.gz QC.unpaired.trimmed.fastq
iu@bielinux[result] kmergenie QC.1.trimmed.fastq.gz -l 31 -k 191
running histogram estimation
Linear estimation: ~48 M distinct 106-mers are in the reads
K-mer sampling: 1/14
| processing
[going to estimate histograms for values of k: 191 181 171 161 15
1 141 131 121 111 101 91 81 71 61 51 41 31
-----
```





# 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

```
-----Total time Wallclock 447.581 s
fitting model to histograms to estimate best k
estimation of the best k so far: 91
refining estimation around [85; 97], with a step of 2
running histogram estimation
Linear estimation: ~65 M distinct 59-mers are in the reads
K-mer sampling: 1/19
| processing
[going to estimate histograms for values of k: 97 95 93 91 89 87
85
-----Total time Wallclock 189.77 s
fitting model to histograms to estimate best k
table of predicted num. of genomic k-mers: histograms.dat
recommended coverage cut-off for best k: 1
best k: 87
iu@bielinux[result] ls [ 1:53午後]
```



# 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

```
File Edit View Search Terminal Help
histograms-k151.histo
histograms-k151.histo.pdf
histograms-k161.histo
histograms-k161.histo.pdf
histograms-k171.histo
histograms-k171.histo.pdf
histograms-k181.histo
histograms-k181.histo.pdf
histograms-k191.histo
histograms-k191.histo.pdf
histograms-k31.histo
histograms-k31.histo.pdf
histograms-k41.histo
histograms-k41.histo.pdf
histograms-k51.histo
histograms-k51.histo.pdf
histograms-k61.histo
histograms-k61.histo.pdf
iu@bielinux[result] cp histograms_report.html ~/Desktop/mac_share
iu@bielinux[result] █
```

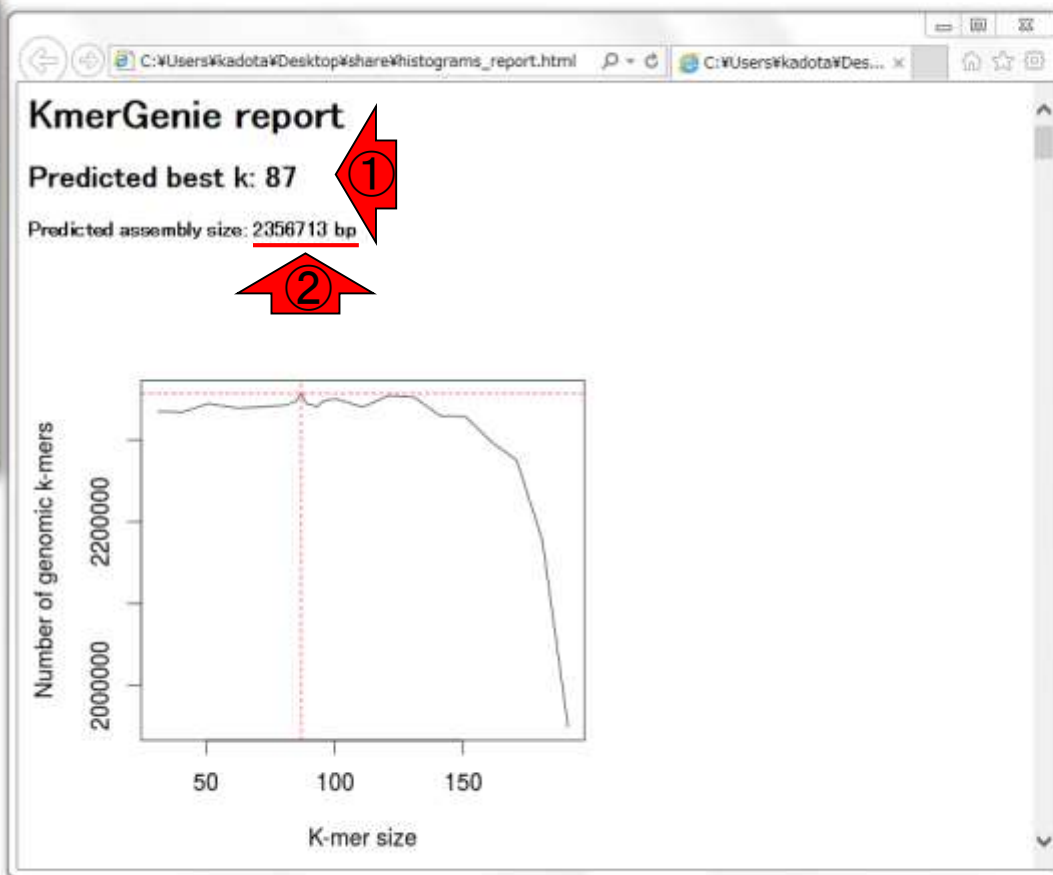
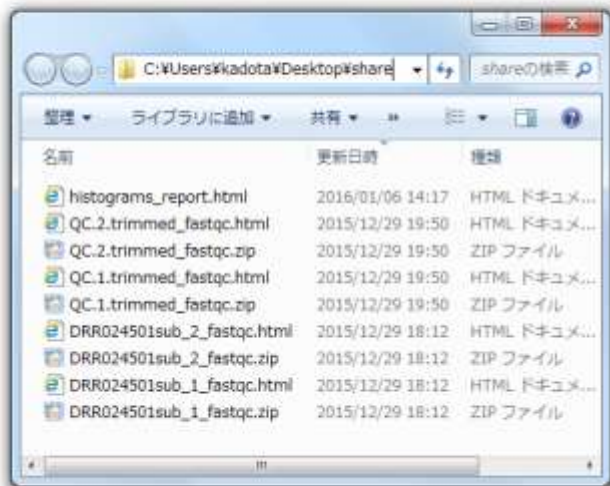
histograms-k97.histo.pdf  
histograms\_report.html  
hoge\_111  
hoge\_121  
hoge\_131  
hoge\_151  
hoge\_171  
hoge\_191  
QC.1.trimmed.fastq.gz  
QC.2.trimmed.fastq.gz  
QC\_qc\_report.pdf  
QC.stats.txt  
QC.unpaired.trimmed.fastq

[ 2:17午後 ]



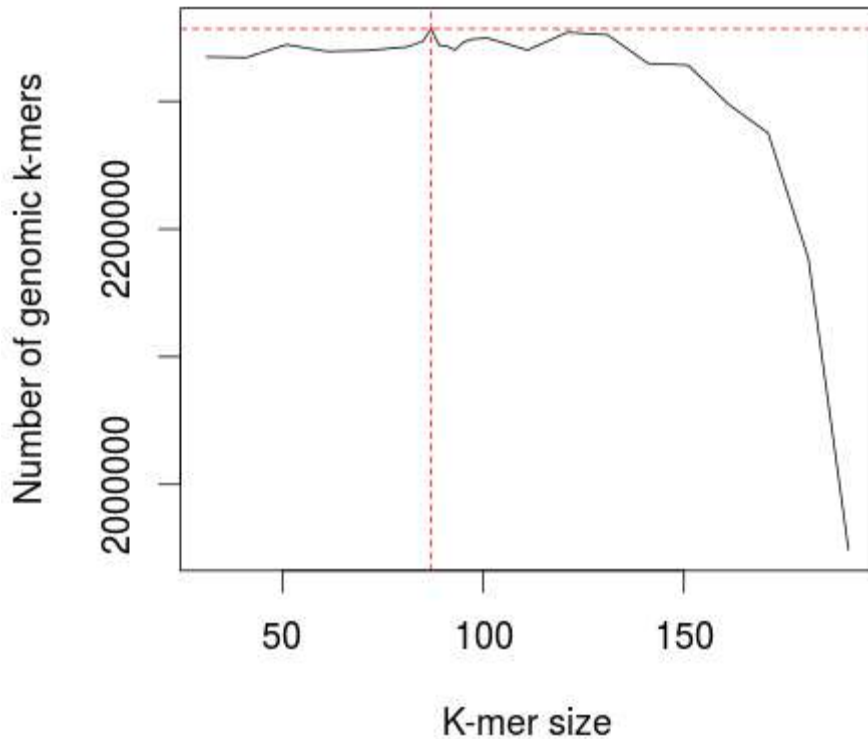
# W11-9: 確認

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最適な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



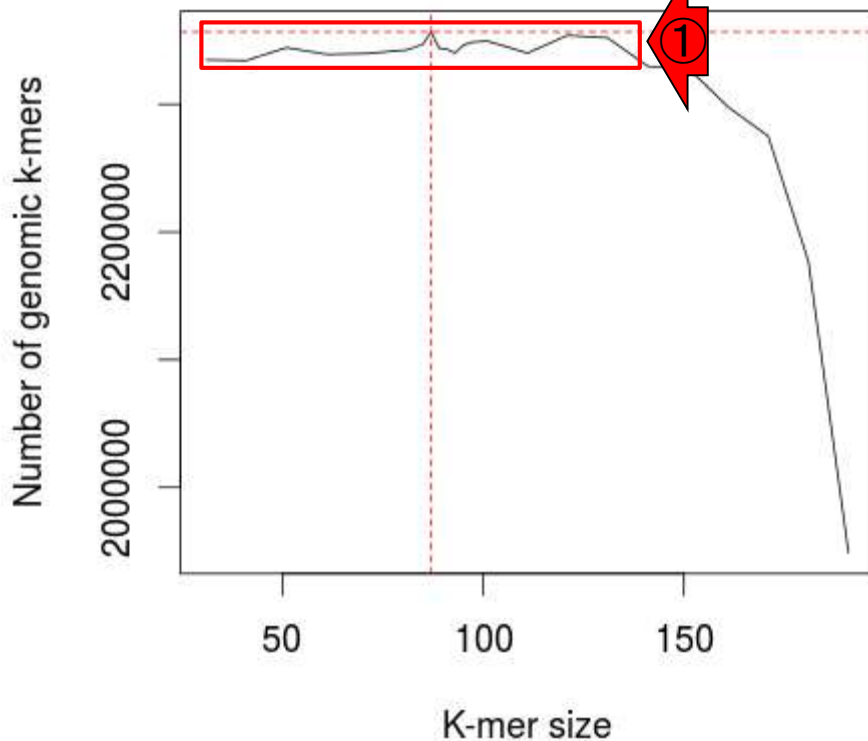
# KmerGenie結果解説

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「シーケ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

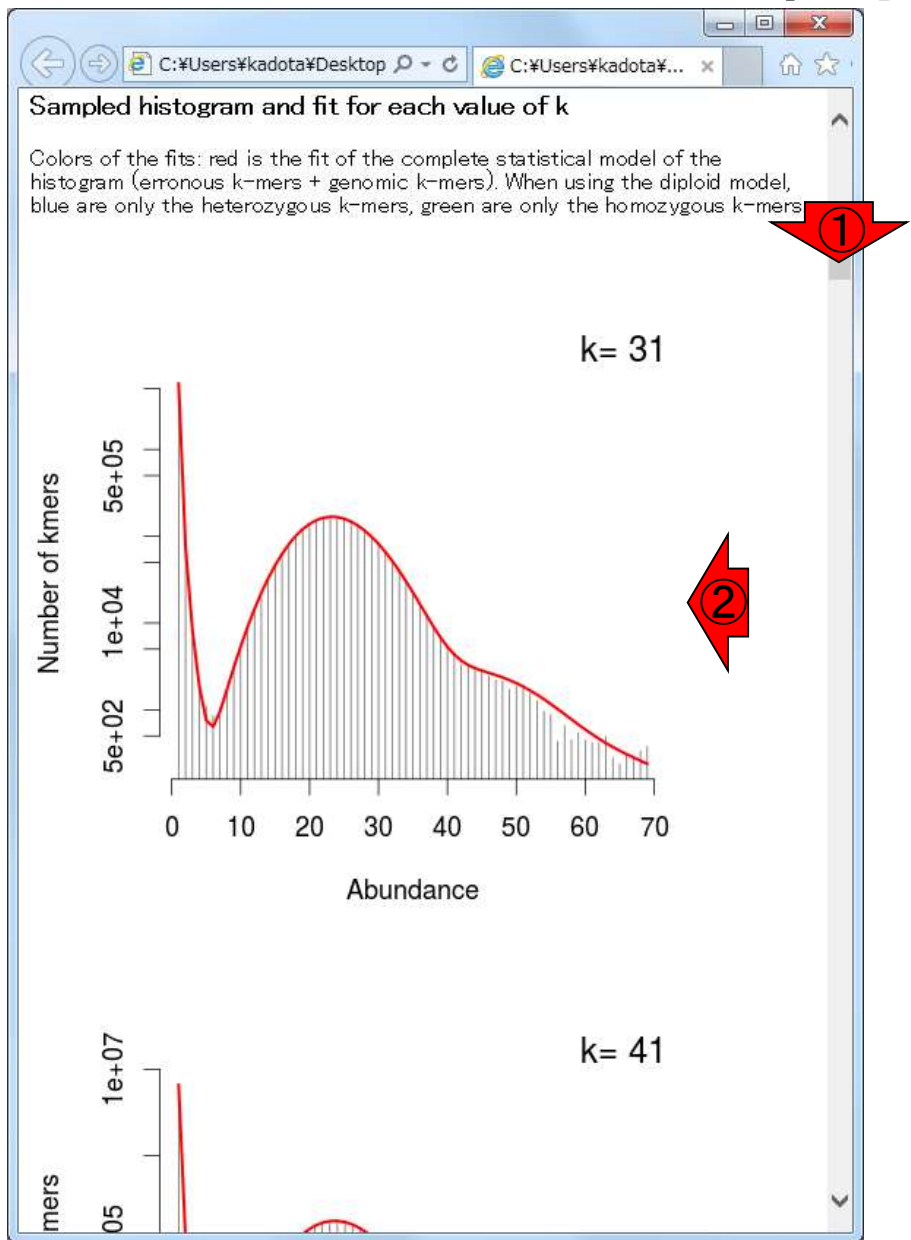


①のあたりを眺めることで、 $k=30-130$ くらいの範囲でどのk-merを用いてもゲノムサイズが2.3-2.4MBという結果になるのだろうと解釈する。但し、これは「genomic k-mersの種類数」なのでシーケンスエラー由来k-merのフィルタリングが実装されていない(甘い)アセンブリプログラムだとそうはいかないだろう、とも想像する



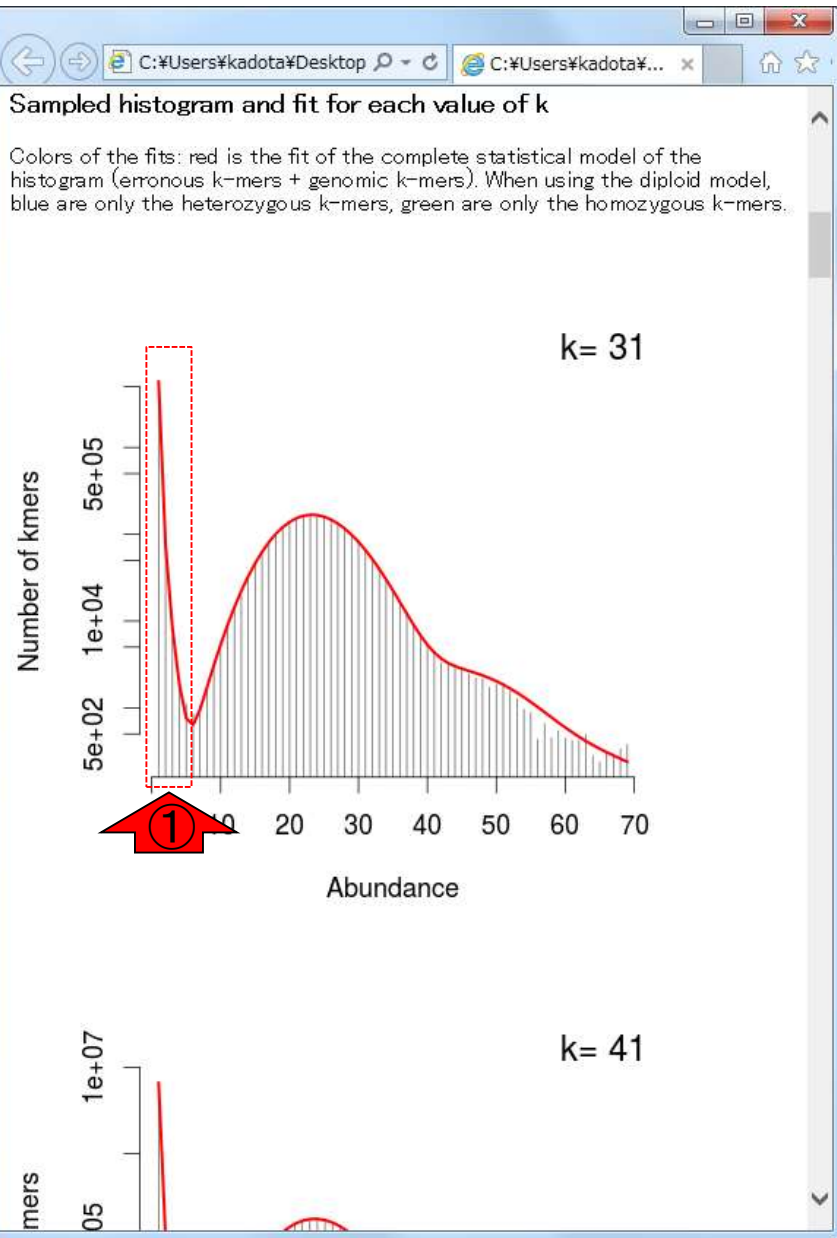
# KmerGenie結果解説

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



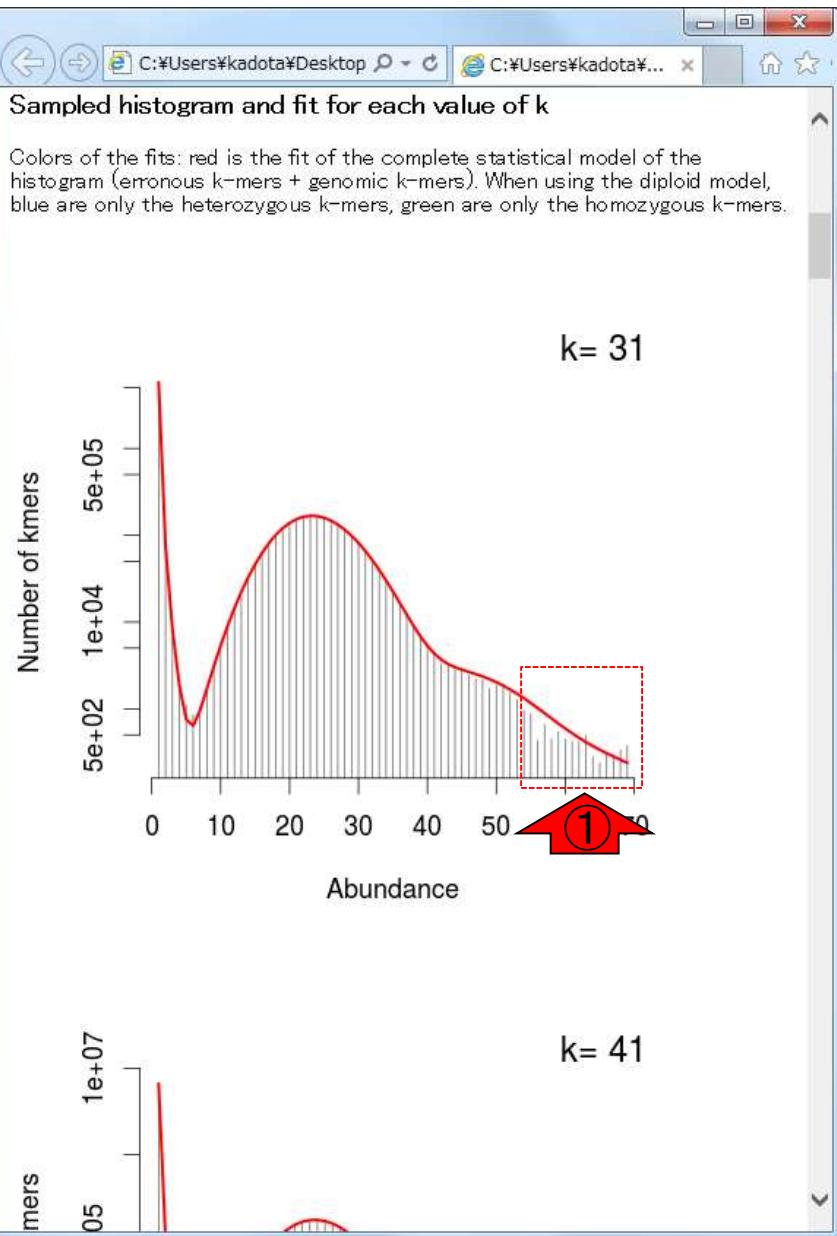
# KmerGenie結果解説

KmerGenieは、①のシーケンスエラー由来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





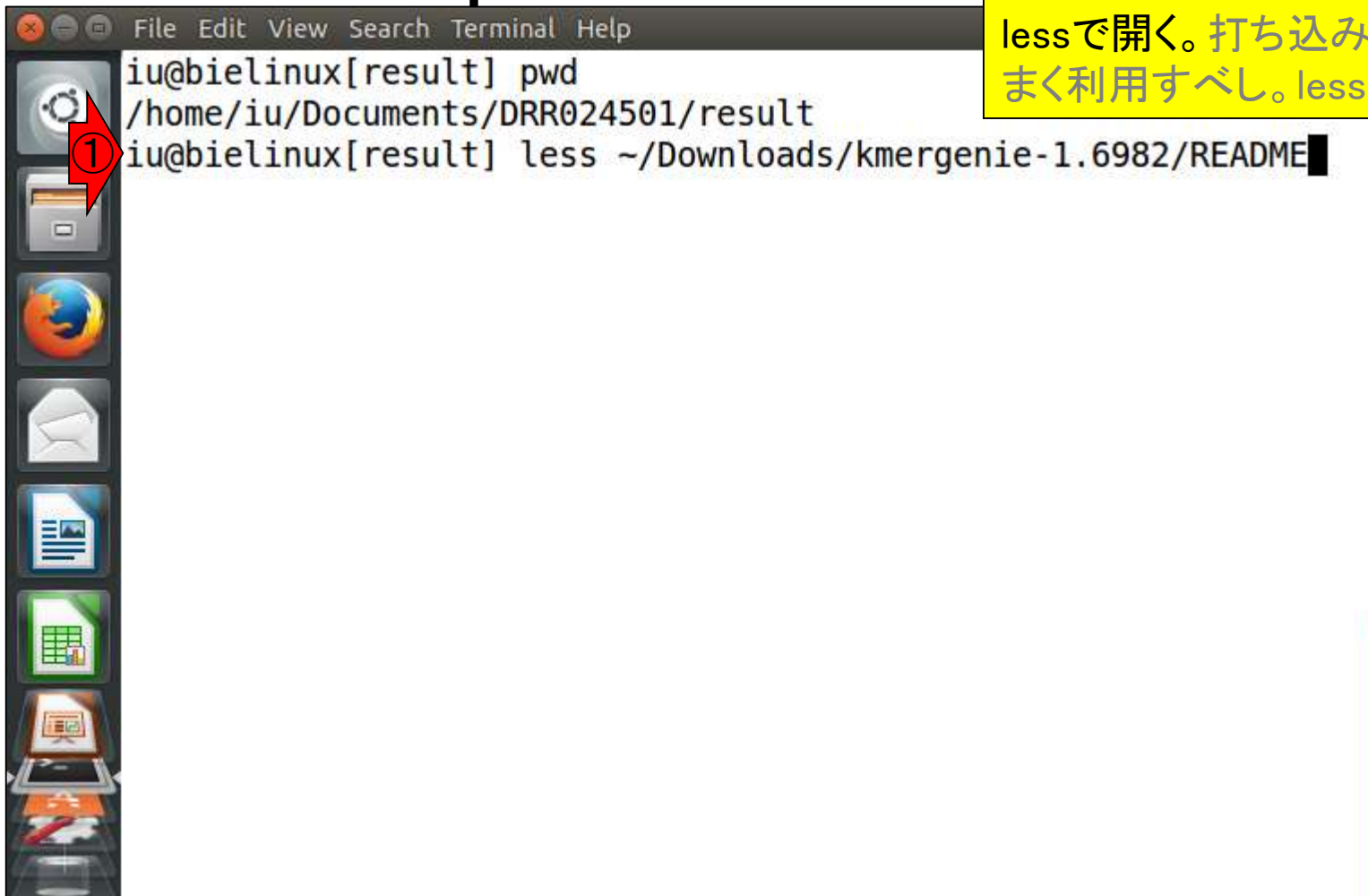
# Contents

- 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)
- 롱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



```
File Edit View Search Terminal Help
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] less ~/Downloads/kmergenie-1.6982/README
```

# 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

```
File Edit View Search Terminal Help
type ./kmergenie to see extra options

input:

Input reads should be exactly those the de novo assembler will use to create contigs, i.e. the list of all single and paired-end reads.

The order does not matter, KmerGenie treats the reads as an unordered set of k-mers. Orientation of the reads also does not matter.

With Velvet, if you have mate-pairs, Velvet uses them to create contigs, so do include them in KmerGenie.
Otherwise, if the mate-pairs are used only for scaffolding (i.e. asm_flag=2 in SOAPdenovo), do not include them.

tips:

* Take a look at the generated HTML report. It provides a s
```





# 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)

```
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] less ~/Downloads/kmergenie-1.6982/README
iu@bielinux[result] ls
fastqCount.txt
histograms.dat
histograms.dat.pdf
histograms-k101.histo
histograms-k101.histo.pdf
histograms-k111.histo
histograms-k111.histo.pdf
histograms-k121.histo
histograms-k121.histo.pdf
histograms-k131.histo
histograms-k131.histo.pdf
histograms-k141.histo
histograms-k141.histo.pdf
histograms-k151.histo
histograms-k151.histo.pdf
histograms-k161.histo
histograms-k71.histo
histograms-k71.histo.pdf
histograms-k81.histo
histograms-k81.histo.pdf
histograms-k85.histo
histograms-k85.histo.pdf
histograms-k87.histo
histograms-k87.histo.pdf
histograms-k89.histo
histograms-k89.histo.pdf
histograms-k91.histo
histograms-k91.histo.pdf
histograms-k93.histo
histograms-k93.histo.pdf
histograms-k95.histo
histograms-k95.histo.pdf
```



①

[10:26午前]

[10:39午前]

# 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たとし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

```
File Edit View Search Terminal Help
histograms-k181.histo.pdf hoge_121
histograms-k191.histo hoge_131
histograms-k191.histo.pdf hoge_151
histograms-k31.histo hoge_171
histograms-k31.histo.pdf hoge_191
histograms-k41.histo QC.1.trimmed.
histograms-k41.histo.pdf QC.2.trimmed.
histograms-k51.histo QC_qc_report.pdf
histograms-k51.histo.pdf QC.stats.txt
histograms-k61.histo QC.unpaired.trimmed.fastq
histograms-k61.histo.pdf
iu@bielinux[result] rm -f histograms* [10:58午前]
iu@bielinux[result] ls [10:59午前]
fastqCount.txt hoge 191
hoge_111 QC.1.trimmed.fastq.gz
hoge_121 QC.2.trimmed.fastq.gz
hoge_131 QC_qc_report.pdf
hoge_151 QC.stats.txt
hoge_171 QC.unpaired.trimmed.fastq
iu@bielinux[result] [10:59午前]
```



QC.1.trimmed.fastq.gz  
QC.2.trimmed.fastq.gz



# 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

```
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls
fastqCount.txt  hoge_191
hoge_111        QC.1.trimmed.fastq.gz
hoge_121        QC.2.trimmed.fastq.gz
hoge_131        QC_qc_report.pdf
hoge_151        QC.stats.txt
hoge_171        QC.unpaired.trimmed.fastq
① iu@bielinux[result] ls QC.*
QC.1.trimmed.fastq.gz QC.stats.txt
QC.2.trimmed.fastq.gz QC.unpaired.trimmed.fastq
② iu@bielinux[result] ls QC.*.tri*
QC.1.trimmed.fastq.gz QC.unpaired.trimmed.fastq
QC.2.trimmed.fastq.gz
③ iu@bielinux[result] ls QC.[0-9].*
QC.1.trimmed.fastq.gz QC.2.trimmed.fastq.gz
iu@bielinux[result] █
```

[11:18午前]

[11:18午前]

[11:18午前]

[11:18午前]

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

```
iu@bielinux[result] ls
fastqCount.txt  hoge_191
hoge_111        QC.1.trimmed.fastq.gz
hoge_121        QC.2.trimmed.fastq.gz
hoge_131        QC_qc_report.pdf
hoge_151        QC.stats.txt
hoge_171        QC.unpaired.trimmed.fastq

① iu@bielinux[result] ls QC.*
QC.1.trimmed.fastq.gz QC.stats.txt
QC.2.trimmed.fastq.gz QC.unpaired.trimmed.fastq

② iu@bielinux[result] ls QC.*.tri*
QC.1.trimmed.fastq.gz QC.unpaired.trimmed.fastq
QC.2.trimmed.fastq.gz

③ iu@bielinux[result] ls QC.[0-9].*
QC.1.trimmed.fastq.gz QC.2.trimmed.fastq.gz

④ iu@bielinux[result] ls QC.[0-9].* > list.txt
iu@bielinux[result] more list.txt
QC.1.trimmed.fastq.gz
QC.2.trimmed.fastq.gz
iu@bielinux[result] █
```

[11:18午前]

[11:18午前]

[11:18午前]

[11:18午前]

[11:56午前]

[11:57午前]



# 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

```
iu@bielinux[result] ls
fastqCount.txt  list.txt
hoge_111       QC.1.trimmed.fastq.gz
hoge_121       QC.2.trimmed.fastq.gz
hoge_131       QC_qc_report.pdf
hoge_151       QC.stats.txt
hoge_171       QC.unpaired.trimmed.fastq
hoge_191

iu@bielinux[result] kmergenie list.txt -l 31 -k 191
running histogram estimation
File list.txt starts with character "Q", hence is interpreted as
a list of file names
Reading 2 read files
Linear estimation: ~103 M distinct 106-mers are in the reads
K-mer sampling: 1/31
| processing

[going to estimate histograms for values of k: 191 181 171 161
151 141 131 121 111 101 91 81 71 61 51 41 31
-----█
```

[12:09午後]

# 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、① paired-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

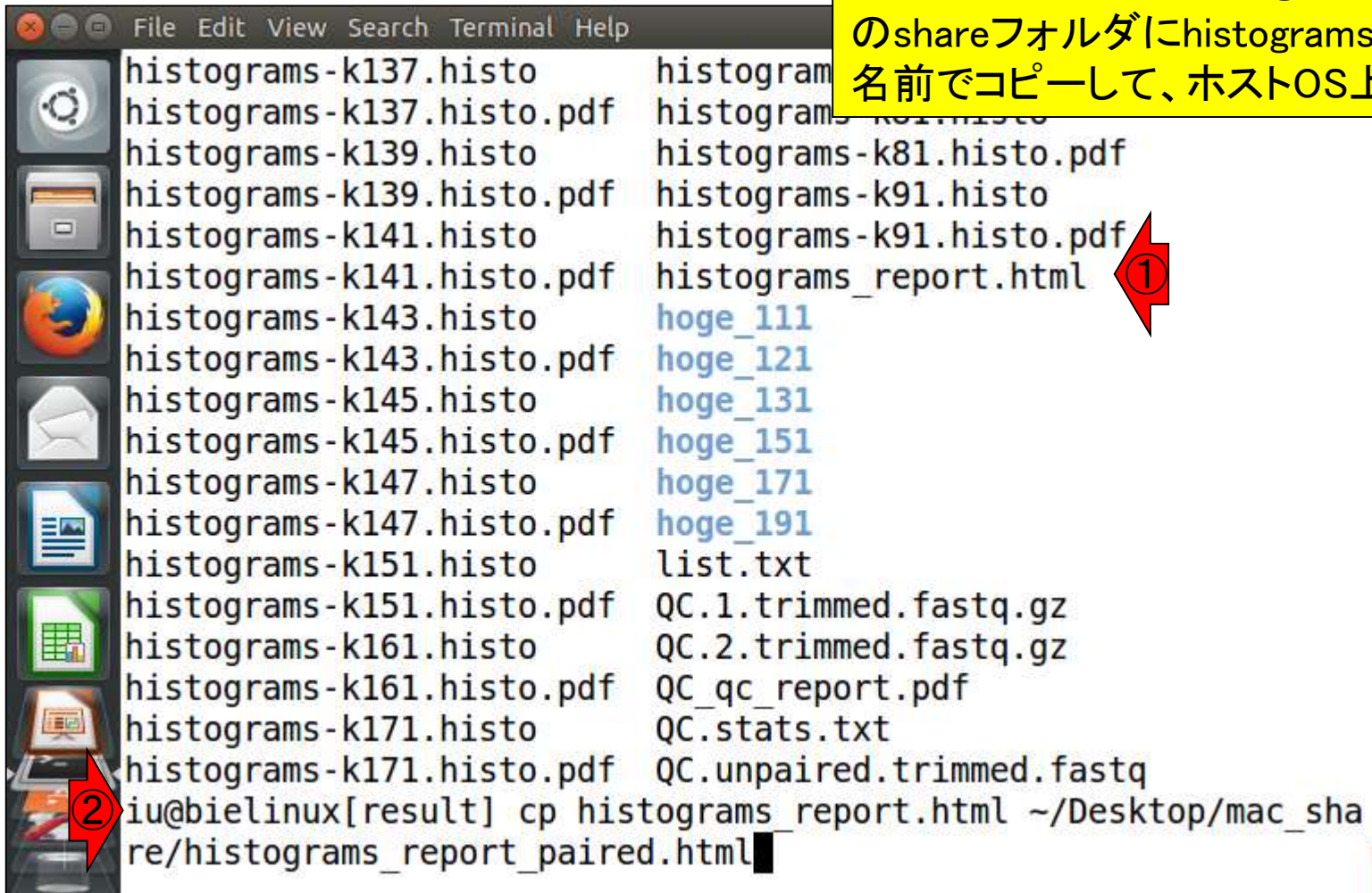
```
File Edit View Search Terminal Help
estimation of the best k so far: 141
refining estimation around [135; 147], with a step of 2
running histogram estimation
File list.txt starts with character "Q", hence is interpreted as
a list of file names
Reading 2 read files
Linear estimation: ~119 M distinct 84-mers are in the reads
K-mer sampling: 1/36
| processing
[going to estimate histograms for values of k: 147 145 143 141
139 137 135
-----Total time Wallclock 286.84
s
fitting model to histograms to estimate best k
table of predicted num. of genomic k-mers: histograms.dat
recommended coverage cut-off for best k: 2
best k: 141
iu@bielinux[result] ls [12:30午後]
```





# 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



```
File Edit View Search Terminal Help
histograms-k137.histo histogram
histograms-k137.histo.pdf histograms
histograms-k139.histo histograms-k81.histo.pdf
histograms-k139.histo.pdf histograms-k91.histo
histograms-k141.histo histograms-k91.histo.pdf
histograms-k141.histo.pdf histograms_report.html
histograms-k143.histo hoge_111
histograms-k143.histo.pdf hoge_121
histograms-k145.histo hoge_131
histograms-k145.histo.pdf hoge_151
histograms-k147.histo hoge_171
histograms-k147.histo.pdf hoge_191
histograms-k151.histo list.txt
histograms-k151.histo.pdf QC.1.trimmed.fastq.gz
histograms-k161.histo QC.2.trimmed.fastq.gz
histograms-k161.histo.pdf QC_qc_report.pdf
histograms-k171.histo QC.stats.txt
histograms-k171.histo.pdf QC.unpaired.trimmed.fastq
iu@bielinux[result] cp histograms_report.html ~/Desktop/mac_sh
re/histograms_report_paired.html
```

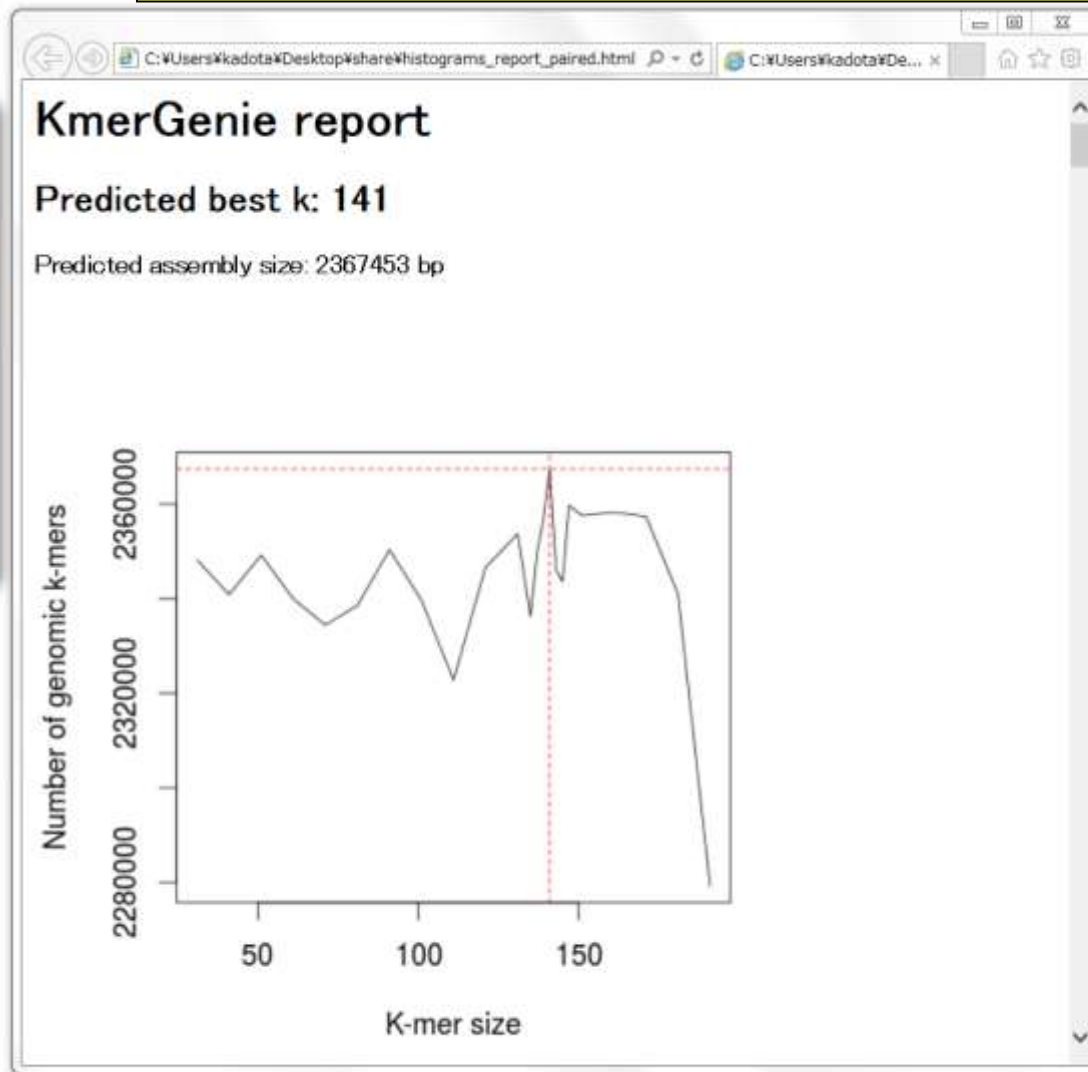


single-end (k=87)とpaired-end (k=141)で推奨k値は大きく異なるものの、ゲノムサイズの推定値はほとんど変わらないことがわかる(single-endは2356713 bp、paired-endは2367453 bp)

# W11-11: 確認



名前	更新日時	種類
histograms_report_paired.html	2016/01/07 13:28	HTML ドキュメント
histograms_report.html	2016/01/06 14:17	HTML ドキュメント
QC.2.trimmed_fastqc.html	2015/12/29 19:50	HTML ドキュメント
QC.2.trimmed_fastqc.zip	2015/12/29 19:50	ZIP ファイル
QC.1.trimmed_fastqc.html	2015/12/29 19:50	HTML ドキュメント
QC.1.trimmed_fastqc.zip	2015/12/29 19:50	ZIP ファイル
DRR024501sub_2_fastqc.html	2015/12/29 18:12	HTML ドキュメント
DRR024501sub_2_fastqc.zip	2015/12/29 18:12	ZIP ファイル
DRR024501sub_1_fastqc.html	2015/12/29 18:12	HTML ドキュメント
DRR024501sub_1_fastqc.zip	2015/12/29 18:12	ZIP ファイル



# 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)
- 롱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<sup>4)</sup>。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、paired-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



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：



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

```
File Edit View Search Terminal Help
iu@bielinux[result] cd
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls
Desktop Downloads Music Public Videos
Documents igv Pictures Templates
iu@bielinux[iu] mkdir bin [12:23午後]
iu@bielinux[iu] ls [12:23午後]
bin Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd bin [12:23午後]
iu@bielinux[bin] pwd [12:23午後]
/home/iu/bin
iu@bielinux[bin] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/fastaLengthFilter.py
iu@bielinux[bin] ls -l [12:23午後]
total 4
-rw-rw-r-- 1 iu iu 1109 1月 6 18:57 fastaLengthFilter.py
iu@bielinux[bin]
```



# 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

```
File Edit View Search Terminal Help
iu@bielinux[bin] pwd
/home/iu/bin
iu@bielinux[bin] ls -l
total 4
-rw-rw-r-- 1 iu iu 1109  1月  6 18:57 fastaLengthFilter.py
iu@bielinux[bin] chmod 755 fastaLengthFilter.py [12:44午後]
iu@bielinux[bin] ls -l [12:44午後]
total 4
-rwxr-xr-x 1 iu iu 1109  1月  6 18:57 fastaLengthFilter.py
iu@bielinux[bin] █ [12:44午後]
```



# 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

```
File Edit View Search Terminal Help
① iu@bielinux[bin] cd
iu@bielinux[iu] pwd
/home/iu
② iu@bielinux[iu] echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/cd-hit:/usr/lib/cd-hit:/home/iu/Downloads/FastQC
③ iu@bielinux[iu] echo $SHELL
/bin/zsh
iu@bielinux[iu]
```

[ 1:15午後 ]  
[ 1:15午後 ]  
[ 1:15午後 ]

# 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

```
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls -l .zshrc
-rw----- 1 iu iu 392 12月 22 15:19 .zshrc
iu@bielinux[iu] tail -n 5 .zshrc
# fi
export PATH=$PATH:/home/iu/Downloads/FastQC
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
iu@bielinux[iu] gedit .zshrc
```

[ 1:18午後 ]  
[ 1:18午後 ]  
[ 1:18午後 ]

~/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

```
iu@bielinux[iu] pwd [ 1:17午後 ]
/home/iu
① iu@bielinux[iu] ls -l .zshrc [ 1:18午後 ]
-rw----- 1 iu iu 392 12月 22 15:19 .zshrc
② iu@bielinux[iu] tail -n 5 .zshrc [ 1:18午後 ]
# fi
export PATH=$PATH:/home/iu/Downloads/FastQC
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
③ iu@bielinux[iu] gedit .zshrc [ 1:18午後 ]
** (gedit:32481): WARNING **: Error querying file info: Error
when getting information for file '/home/iu/.goutputstream-3XD
NAY': No such file or directory
④ iu@bielinux[iu] tail -n 5 .zshrc [ 1:23午後 ]
```



# W12-3: パスを通す

```
iu@bielinux[iu] ls -l .zshrc [ 1:18午後 ]
-rw----- 1 iu iu 392 12月 22 15:19 .zshrc
iu@bielinux[iu] tail -n 5 .zshrc [ 1:18午後 ]
# fi
export PATH=$PATH:/home/iu/Downloads/FastQC
iu@bielinux[iu] gedit .zshrc [ 1:18午後 ]
** (gedit:32481): WARNING **: Error querying file info: Error
when getting information for file '/home/iu/.goutputstream-3XD
NAY': No such file or directory
iu@bielinux[iu] tail -n 5 .zshrc [ 1:23午後 ]
# fi
export PATH=$PATH:/home/iu/Downloads/FastQC:/home/iu/bin
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
iu@bielinux[iu] [ 1:27午後 ]
```



# 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

```
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls -l .zshrc
-rw----- 1 iu iu 405  1月  8 13:23 .zshrc
① iu@bielinux[iu] echo $PATH [ 1:37午後 ]
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/cd-hit:/usr/lib/cd-hit:/home/iu/Downloads/FastQC
② iu@bielinux[iu] source .zshrc [ 1:37午後 ]
iu@bielinux[iu] echo $PATH [ 1:38午後 ]
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/cd-hit:/usr/lib/cd-hit:/home/iu/Downloads/FastQC:/home/iu/Downloads/FastQC:/home/iu/bin
iu@bielinux[iu] [ 1:38午後 ]
```



③



# W12-4: 確認

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だけ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...

```
File Edit View Search Terminal Help
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls -l .zshrc
-rw----- 1 iu iu 405 1月 8 13:23 .zshrc
iu@bielinux[iu] echo $PATH [ 1:37午後 ]
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/cd-hit:/usr/lib/cd-hit:/home/iu/Downloads/FastQC
iu@bielinux[iu] source .zshrc [ 1:37午後 ]
iu@bielinux[iu] echo $PATH [ 1:38午後 ]
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/cd-hit:/usr/lib/cd-hit:/home/iu/Downloads/FastQC:/home/iu/Downloads/FastQC:/home/iu/bin
iu@bielinux[iu] █ [ 1:38午後 ]
```

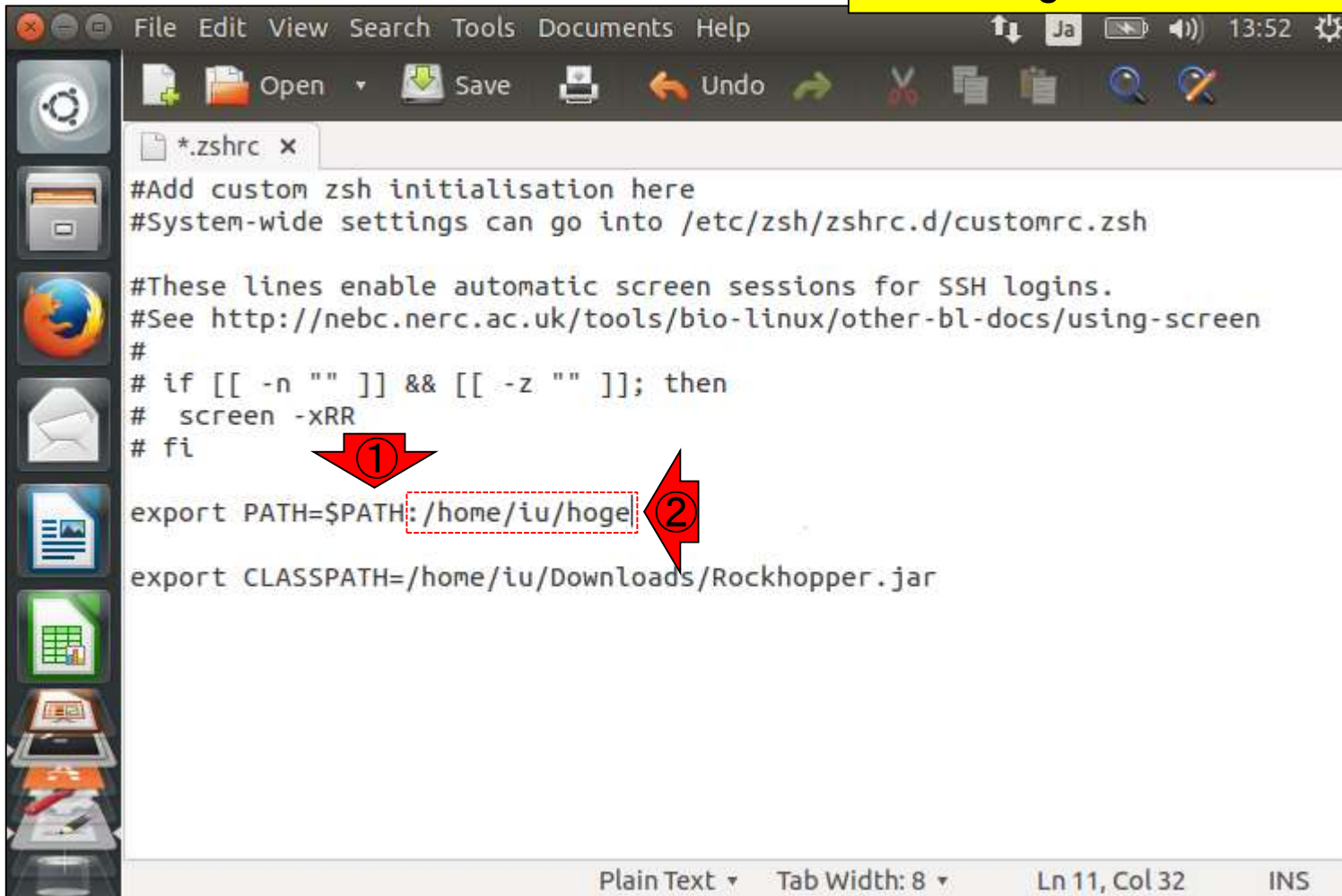




# W12-4: 確認

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



The screenshot shows a text editor window with a dark theme. The title bar indicates the file is \*.zshrc. The editor contains the following text:

```
#Add custom zsh initialisation here
#System-wide settings can go into /etc/zsh/zshrc.d/customrc.zsh

#These lines enable automatic screen sessions for SSH logins.
#See http://nebc.nerc.ac.uk/tools/bio-linux/other-bl-docs/using-screen
#
# if [[ -n "" ]] && [[ -z "" ]]; then
#   screen -xRR
# fi

export PATH=$PATH:/home/iu/hoge

export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
```

Two red arrows with circled numbers point to specific parts of the code:

- Arrow ① points to the space between `$PATH` and `:` in the `export PATH=$PATH:/home/iu/hoge` line.
- Arrow ② points to the path `/home/iu/hoge` in the same line, which is enclosed in a red dashed box.

The status bar at the bottom shows "Plain Text", "Tab Width: 8", "Ln 11, Col 32", and "INS".

# 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

```
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls
bin      Documents  igv      Pictures  Templates
Desktop  Downloads  Music    Public    Videos
① iu@bielinux[iu] where fastaLengthFilter.py
/home/iu/bin/fastaLengthFilter.py
/home/iu/bin/fastaLengthFilter.py
② iu@bielinux[iu] file ~/bin/fastaLengthFilter.py [ 6:12午後 ]
/home/iu/bin/fastaLengthFilter.py: Python script, ASCII text executable
iu@bielinux[iu] [ 6:12午後 ]
```

# 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ならず、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

```
iu@bielinux[iu] pwd
/home/iu
iu@bielinux[iu] ls
bin      Documents  igv      Pictures  Template
Desktop  Downloads  Music    Public    Videos
iu@bielinux[iu] fastaLengthFilter.py -h

② Remove sequences shorter than threshold.
Usage : fastaLengthFilter.py <filename> <threshold> ③

iu@bielinux[iu] fastaLengthFilter.py [ 6:23午後]

Remove sequences shorter than threshold.
Usage : fastaLengthFilter.py <filename> <threshold>

iu@bielinux[iu] [ 6:23午後]
```



# 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

```
iu@bielinux[iu] pwd [ 6:39午後 ]
/home/iu
iu@bielinux[iu] ls [ 6:39午後 ]
bin Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] fastaLengthFilter.py -h [ 6:39午後 ]
Remove sequences shorter than threshold.
Usage : fastaLengthFilter.py <filename> <threshold>
iu@bielinux[iu] fastaLengthFilter.py [ 6:39午後 ]
Remove sequences shorter than threshold.
Usage : fastaLengthFilter.py <filename> <threshold>
① iu@bielinux[iu] cd ~/Documents/DRR024501/result [ 6:39午後 ]
iu@bielinux[result] pwd [ 6:39午後 ]
/home/iu/Documents/DRR024501/result
② iu@bielinux[result] ls [ 6:39午後 ]
```



# 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

```
File Edit View Search Terminal Help
histograms-k143.histo      hoge_111
histograms-k143.histo.pdf hoge_121
histograms-k145.histo      hoge_131
histograms-k145.histo.pdf hoge_151
histograms-k147.histo      hoge_171
histograms-k147.histo.pdf hoge_191
histograms-k151.histo      list.txt
histograms-k151.histo.pdf QC.1.trimmed.fastq.gz
histograms-k161.histo      QC.2.trimmed.fastq.gz
histograms-k161.histo.pdf QC_qc_report.pdf
histograms-k171.histo      QC.stats.txt
histograms-k171.histo.pdf QC.unpaired.trimmed.fastq
iu@bielinux[result] rm -f histograms* [ 6:44午後 ]
iu@bielinux[result] ls [ 6:44午後 ]
fastqCount.txt list.txt
hoge_111 QC.1.trimmed.fastq.gz
hoge_121 QC.2.trimmed.fastq.gz
hoge_131 QC_qc_report.pdf
hoge_151 QC.stats.txt
hoge_171 QC.unpaired.trimmed.fastq
hoge_191
iu@bielinux[result] █ [ 6:44午後 ]
```



# 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

```
iu@bielinux[result] pwd [ 7:12午後 ]
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls -d hoge_* [ 7:12午後 ]
hoge_111 hoge_121 hoge_131 hoge_151 hoge_171 hoge_191
iu@bielinux[result] ls hoge_111 [ 7:12午後 ]
contigs.fa LastGraph PreGraph Sequences
Graph Log Roadmaps stats.txt
① iu@bielinux[result] head -n 8 hoge_111/contigs.fa [ 7:12午後 ]
>NODE_1_length_111_cov_25.702703
CCAGCTGGCAAGGGTAATCTAAACCACCCATTAGCTGTTATTGAAGCTTTGCAGCAACGA
GTTGATGATAAAATGACCGTTTCGGTTGATGTGGGGAGCCATTATATTTGGATGGCCCGG
CACTTCCGAAGTTATGAGCCTCGCCATTTATTGTTTAGTAATGGGATGCAGACGCTTGGA
GTGGCGCTACCTTGGTCAATTTTCGGCTGCGTTAGTTCGGCC
>NODE_3_length_141_cov_32.815601
CTTTTCACGGATGCGGCACGAGGATCCACAGGGCGATTATGGCCGGCAAACACGTCAACG
ACTTGTTATCACCGCATTATTACGTGAGTCGATTTTCGTATAAAACAGTGTTAAACACTAA
② iu@bielinux[result] wc hoge_111/contigs.fa [ 7:12午後 ]
127260 127260 6680703 hoge_111/contigs.fa
③ iu@bielinux[result] grep -c ">" hoge_111/contigs.fa
23761
iu@bielinux[result] █ [ 7:12午後 ]
```



# W12-8: 実行

① hoge\_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

```
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 Roadmaps stats.txt
① iu@bielinux[result] fastaLengthFilter.py hoge_111/contigs.fa 3
00 > hoge_111/contigs_300.fa
iu@bielinux[result] ls hoge_111 [ 8:30午後 ]
contigs_300.fa Graph Log Roadmaps stats.txt
contigs.fa LastGraph PreGraph Sequences
iu@bielinux[result] [ 8:30午後 ]
```



# 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

```
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 Roadmaps stats.txt
iu@bielinux[result] fastaLengthFilter.py hoge_111/contigs.fa 3
00 > hoge_111/contigs_300.fa
iu@bielinux[result] ls hoge_111 [ 8:30午後 ]
contigs_300.fa Graph Log Roadmaps stats.txt
contigs.fa LastGraph PreGraph Sequences
① iu@bielinux[result] wc hoge_111/contigs_300.fa [ 8:30午後 ]
3874 3874 840668 hoge_111/contigs_300.fa
② iu@bielinux[result] grep -c ">" hoge_111/contigs_300.fa
1937
③ iu@bielinux[result] grep -v ">" hoge_111/contigs_300.fa | wc ←
1937 1937 813550
iu@bielinux[result] [ 8:33午後 ]
iu@bielinux[result] [ 8:33午後 ]
```

# W12-8: 実行

```

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
iu@bielinux[res
3874 3874 8
iu@bielinux[res
1937
iu@bielinux[result]
1937 1937 81
iu@bielinux[result] [ 8:33午後 ]
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と必 (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](#) (last modified 2014/02/07)
- 前処理 | フィルタリング | [任意のリード\(サブセット\)を抽出](#) (last modified 2014/08/21)
- 前処理 | フィルタリング | [指定した長さの範囲の配列を抽出](#) (last modified 2015/02/26)
- 前処理 | フィルタリング | [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。③ description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

```
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result

iu@bielinux[result] wc hoge_111/contigs_300.fa [ 8:56午後 ]
3874 3874 840668 hoge_111/contigs_300.fa

iu@bielinux[result] grep -c ">" hoge_111/contigs_300.fa
1937

iu@bielinux[result] grep -v ">" hoge_111/contigs_300.fa | wc ←
1937 1937 813550

iu@bielinux[result] tail -n 4 hoge_111/contigs_300.fa
>sequence1936
ATAATACGTTAACTTGCTTCAGCACTTCTTCACCGCTGAATAACATTTTACCTAAGAATCCT
CGCAAGAAAGTATTATCACTTTCTTCTTTACTAGCAAAGTACGTAACATTCCACAATACTG
ATGTCTTCAGTAAATTCTGGTGTTAGATCCTTGGGCATGTAAGCGCGGGTAGTAGTAACGCC
CCAGTTAACAGTACCGGTATCTGGTTTTACATTACCAGCAATAATCTGCATTAAAAGACTAG
TAGTTGCATCGGAACGAGAAACGAGGGCCGTTTTATCGCCTGGATGGAGAAT
>sequence1937
CTAACCGATAAATAACTAAAGTTAAAAATATACAACCTTGCTAGTATAACAAGTATTATCAAT
AAATCGCAACTGTAACGCTTAAATAGTCTATTTTAACTTTCCAGCACTGTTTCTAAATGAA
TTCCATGACTTGCTTTCAATAATAACAATGTCATCATTTGTAAGGCTATCTTGCAGTTGTGCG
GTCTTTTGTCTTTTTGTTCAATATCAAATAATGTAATGCATCTGCCGAATATTTATTTTG
TAGCTCATCTCGTAAAGCCTTCATATGGGGCCCAATTA AAAACACGGATTGA
iu@bielinux[result] [ 8:56午後 ]
```



# 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



```
File Edit View Search Terminal Help
iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls -d hoge_*
hoge_111 hoge_121 hoge_131 hoge_151 hoge_171 hoge_191
iu@bielinux[result] fastaLengthFilter.py hoge_121/contigs.fa 3
00 > hoge_121/contigs_300.fa
iu@bielinux[result] fastaLengthFilter.py hoge_131/contigs.fa 3
00 > hoge_131/contigs_300.fa
iu@bielinux[result] fastaLengthFilter.py hoge_151/contigs.fa 3
00 > hoge_151/contigs_300.fa
iu@bielinux[result] fastaLengthFilter.py hoge_171/contigs.fa 3
00 > hoge_171/contigs_300.fa
iu@bielinux[result] fastaLengthFilter.py hoge_191/contigs.fa 3
00 > hoge_191/contigs_300.fa
iu@bielinux[result]
iu@bielinux[result] █
```

[ 9:09午後 ]

[ 9:09午後 ]

# 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

```
iu@bielinux[result] pwd [ 9:11午後 ]
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls -d hoge_* [ 9:11午後 ]
hoge_111 hoge_121 hoge_131 hoge_151 hoge_171 hoge_191
iu@bielinux[result] grep -v ">" hoge_121/contigs_300.fa | wc
2942 2942 1566326
iu@bielinux[result] grep -v ">" hoge_131/contigs_300.fa | wc
2449 2449 2153849
iu@bielinux[result] grep -v ">" hoge_151/contigs_300.fa | wc
1306 1306 2600683
iu@bielinux[result] grep -v ">" hoge_171/contigs_300.fa | wc
168 168 2381691
iu@bielinux[result] grep -v ">" hoge_191/contigs_300.fa | wc
336 336 2405767
iu@bielinux[result] [ 9:11午後 ]
```



# W12-9: 結果を確認

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

- 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
```



# 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	ウェブ資料
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)
- 롱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	ウェブ資料
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に有意義であろう。

②  
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)<sup>18)</sup>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)<sup>3)</sup> 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

かには、DDBJ Pipeline イ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:

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

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
```

# 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

## 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](#)

スライドを見るだけ

## 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



大学共同利用法人 情報・システム研究機構 国立遺伝学研究所  
スーパーコンピュータシステム  
SuperComputer Facilities of National Institute of Genetics

サイトポリシー サイトマップ

検索...

検索

現在地: Home

2016年01月14日

Language/言語



ホーム

このサイトへのログイン

Login  
(スパコンユーザでログイン可)

システム構成

ハードウェア構成

ソフトウェア構成

登録ユーザ情報 (2015年3月1日~現在)

※1日1回正午更新

ユーザ登録数一覧

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	ウェブ資料
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	2				



```

iu@bielinux[result] pwd
/home/iu/Documents/DRR024501/result
iu@bielinux[result] ls
fastqCount.txt  hoge_171  QC_qc_report.pdf
hoge_111       hoge_191  QC.stats.txt
hoge_121       list.txt  QC.unpaired.trimmed.fastq
hoge_131       QC.1.trimmed.fastq.gz
hoge_151       QC.2.trimmed.fastq.gz
iu@bielinux[result]
    
```



# 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

**Workflow:** Select Query Files → **Select Tools** → Set QuerySet → Set GenomeSet → Set Map Options → Confirmation

**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			Evaluation			Analysis			Output format			Comment
				Base space	Color space	Paired end	Depth	Coverage	Error rate	SNP	Indel	.gff	.bed	SAM		
<input type="checkbox"/>	<a href="#">BLAT</a>		34	✓												Single-end analysis only
<input type="checkbox"/>	<a href="#">bwa</a>		0.6.1	✓		✓	✓	✓	✓						✓	
<input type="checkbox"/>	<a href="#">Bowtie</a>		0.12.7	✓	✓	✓	✓	✓	✓	✓					✓	
<input type="checkbox"/>	<a href="#">TopHat</a>		1.0.11	✓		✓	✓	✓	✓						✓	
<input type="checkbox"/>	<a href="#">Bowtie2</a>		2.0.0	✓	✓	✓	✓	✓	✓	✓					✓	For reads longer than about 50 bp, Bowtie2 is generally faster, more sensitive, and uses less memory than Bowtie1.
<input type="checkbox"/>	<a href="#">TopHat2</a>		2.0.9	✓		✓	✓	✓	✓						✓	

### de novo Assembly

Total limit = 22 Gbp

	Tool	Help	Version	Base space	Color space	Paired-end	MSS (WGS)	Comment
<input type="checkbox"/>	<a href="#">SOAPdenovo</a>		1.05	✓		✓		
<input type="checkbox"/>	<a href="#">ABYSS</a>		1.3.2	✓		✓		Maximum K-mer value is 64.
<input type="checkbox"/>	<a href="#">Velvet</a>		1.2.10	✓		✓	✓	We severe recommend when performing Velvet, total length of those reads is up to 22G bp. Maximum K-mer value is 64.

# W15-2: Select Tools

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.  
DBJ Read Annotation Pipeline.  
Development Team.

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

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

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

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

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

BACK NEXT

# W15-3: Set QuerySet

①解析したいデータにチェックをいれて、② Set 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

Generating Query Sets from Query Read Files

Paired-end analysis  
Layout of paired sequence. 5'-3' 3'-5'

5' Linker(1) Target Linker(2) Linker(3) Target Linker(4) 5'

Run ACCESSION	Read length	Quality Score
<input checked="" type="checkbox"/> QC.1.trimmed.fastq.gz	bp	

Set as Mate-Pair Set as Pair-End

QUERY SET

RESET BACK NEXT



# 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

Setting for De Novo Assembly

velvet

Set optional parameters of the paired-end analysis

Step1) Convert sequences

Shuffle the sequence.

perl shuffleSequences.pl query\_1.fastq query\_2.fastq shuffle\_query\_pe.fastq

Running velveth.

Velveth output\_directory/ 23 -fastq -shortPaired shuffle\_query\_pe.fastq

Step2) Assembly

Velvetg output\_directory/ -ins\_length 300 -exp\_cov auto

Step3) Set parameters of the CONFIG mapping tool

Step4) Create assembled sequences in FASTA file from pileupped reads to [submit WGS division of DDBJ.](#)

Set filtered length for contigs

perl lengthfilter.pl pileupFile 100 out\_WGS.txt

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

# W15-4: Set Ass.Options

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.

Select Query Files → Select Tools → Set QuerySet → **Set Ass. Options** → Confirmation → Running Status

## Setting for De Novo Assembly

BACK NEXT

velvet

### Set optional parameters of the paired-end analysis

Step1) Convert sequences

Shuffle the sequence.  
perl shuffleSequences\_fastq.pl query\_1.fastq query\_2.fastq shuffle\_query\_pe.fastq

Running velvet.  
Velveth output\_directory/ [131 -fastq] -shortPaired shuffle\_query\_pe.fastq

Step2) Assembly  
Velvetg output\_directory/ [ ]

Step3) Set parameters of the CONFIG mapping tool

Step4) Create assembled sequences in FASTA file from pileupped reads to [submit WGS division of DDBJ.](#)

Set filtered length for contigs  
 perl lengthfilter.pl pileupFile [300] out\_WGS.txt

BACK NEXT



# 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

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.

Select Query Files → Select Tools → Set QuerySet → Set Ass. Options → **Confirmation** → Running Status

## Run Confirmation

BACK RUN

Destination of mail

When the request is completed, the system sends an email to this address.

\* Required

Result files will be deleted 60 days after submission.

Assembly [velvet]

Query sets

PairedOrientation	RunAccession	RunAlias	RowLength	QualityScore1	QualityScore2
paired	20392	L.hokkaidonensis_MiSeq_denovo			

Assembly commands

velvet

### Set optional parameters of the paired-end analysis

Step1) Convert sequences

*Shuffle the sequence.*  
perl shuffleSequences\_fastq.pl query\_1.fastq query\_2.fastq shuffle\_query\_pe.fastq

*Running velveth.*  
Velveth output\_directory/  -shortPaired shuffle\_query\_pe.fastq

Step2) Assembly

Velvetg output\_directory/

Step3) Set parameters of the CONFIG mapping tool

Step4) Create assembled sequences in FASTA file from pileupped reads to [submit WGS division of DDBJ.](#)

Web ページからのメッセージ

Do you really want to execute pipeline programs?

OK キャンセル

# 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



宛先 kadota@bi.a.u-tokyo.ac.jp

C C 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/>



If you have troubles in this service, please write to [pipeline\\_dev@ddbj.nig.ac.jp](mailto: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

http://p.ddbj.nig.ac.jp/pipeline/DetailView.do?query\_set\_i  
Detail view

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.  
DBJ Read Annotation Pipeline.  
Development Team.

**ID**  
21119

**Tool (Version)**  
Velvet (1.2.10)

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

**Download modified queries**

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

**Download wgs file**

- [out\\_WGS.fasta.gz \(Original size 2.1 MB\)](#)

**Assembly statistics**

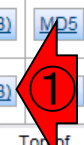
Contig # : 8,398  
Total contig size : 3,710,829  
Maximum contig size : 6,572  
Minimum contig size : 261  
N50 contig size : 506

**Time**

Wait time	Start time	End time
0: 0:17	2016-01-15 18:52:36	2016-01-15 20:36:28

Command	Start time	End time	Log1	Log2	Result	MD5
perl shuffleSequences_fastq.pl QC.1.trimmed.fastq QC.2.trimmed.fastq QC_pe.fastq	2016-01-15 18:52:36	2016-01-15 18:52:46				
velveth outputDir/ 131 -fastq -shortPaired /home/w3pipeline/refdata/tmp/agribio/21119/20926/unspecified/QC_pe.fastq	2016-01-15 18:52:46	2016-01-15 18:53:28	<a href="#">View</a>		<a href="#">Download(66.5 MB)</a>	<a href="#">MD5</a>
velvetg outputDir/	2016-01-15 18:54:22	2016-01-15 20:35:14	<a href="#">View</a>		<a href="#">Download(66.5 MB)</a>	

BACK Top of page



①



# 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

```
iu@bielinux[backup] pwd
/home/iu/Desktop/backup
iu@bielinux[backup] ls -l velvet.*
-rw-rw-r-- 1 iu iu 66517438  1月 18 14:01 velvet.zip
iu@bielinux[backup] unzip velvet.zip
Archive:  velvet.zip
  creating:  velvet/
  inflating:  velvet/Log
  inflating:  velvet/Sequences
  inflating:  velvet/Roadmaps
  inflating:  velvet/PreGraph
  inflating:  velvet/Graph
  inflating:  velvet/contigs.fa
  inflating:  velvet/stats.txt
  inflating:  velvet/LastGraph
iu@bielinux[backup] █
```

[ 4:38午後 ]

[ 4:38午後 ]

[ 4:39午後 ]

# 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

```
iu@bielinux[backup] pwd [ 4:45午後 ]
/home/iu/Desktop/backup
① 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
② iu@bielinux[velvet] wc contigs.fa [ 4:46午後 ]
75235 75235 4077589 contigs.fa
③ iu@bielinux[velvet] grep -c ">" contigs.fa [ 4:46午後 ]
8398
iu@bielinux[velvet] [ 4:46午後 ]
```

# W18-3: 行数とコンティグ数

- ① velvet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

```
iu@bielinux[backup] pwd [ 4:45午後 ]
/home/iu/Desktop/backup
① 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
② iu@bielinux[velvet] wc contigs.fa [ 4:46午後 ]
75235 75235 4077589 contigs.fa
③ iu@bielinux[velvet] grep -c ">" contigs.fa [ 4:46午後 ]
8398
iu@bielinux[velvet] [ 4:46午後 ]

Contig # : 8,398
Total contig size : 3,710,829
Maximum contig size : 6,572
Minimum contig size : 261
N50 contig size : 506
```



# 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
iu@bielinux[velvet] pwd
/home/iu/Desktop/backup/velvet
iu@bielinux[velvet] ls
contigs.fa LastGraph PreGraph Sequences
Graph Log Roadmaps stats.txt
iu@bielinux[velvet] wc contigs.fa
75235 75235 4077589 contigs.fa
iu@bielinux[velvet] grep -c ">" contigs.fa
8398
① iu@bielinux[velvet] fastaLengthFilter.py contigs.fa 300 > hoge.fa
iu@bielinux[velvet] grep -c ">" hoge.fa
2449
② iu@bielinux[velvet]
```

[ 4:45午後]

[ 4:46午後]

[ 4:46午後]

[ 4:46午後]

[ 4:46午後]

[ 5:01午後]

[ 5:02午後]





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

# W19-2: Select Tools

Preprocessing Start

step-1  
Preprocessing

Mapping / de novo Assembly

step-2

**Workflow**

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

**JOB STATUS**

step1. Preprocessing

step1. Mapping

step1. de novo Assembly

step2-All status

**HELP**

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

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

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

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

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

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

BACK NEXT

# 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

The screenshot shows the 'Setting for De Novo Assembly' page in the DDBJ pipeline. The breadcrumb trail at the top indicates the current step: 'Set Ass. Options'. The left sidebar contains navigation menus for ACCOUNT, ANALYSIS, and JOB STATUS. The main content area is titled 'Setting for De Novo Assembly' and includes a 'platanus' section with the following steps:

- Step1) Assembly :** Construct contigs using the algorithm based on the de Bruijn graph.  
Command: `platanus assemble -t 15 -m 120 -o out [ ] -f PE1.fastq PE2.fastq`
- Step2) Scaffold :** Map paired-end (mate-pair) reads on contigs and construct scaffolds.  
Command: `platanus scaffold -t 8 -o out [ ]  
-c out_contig.fa -b out_contigBubble.fa -IP1 PE1.fastq PE2.fastq (-OP2 MP1.fastq MP2.fastq)`
- Step3) Gap Close :** Map paired-end (mate-pair) reads on scaffolds and assemble reads on gaps and close gaps.  
Command: `platanus gap_close -t 8 -o out [ ]  
-c out_scaffold.fa -IP1 PE1.fastq PE2.fastq (-OP2 MP1.fastq MP2.fastq)`
- Step5) Create assembled sequences in FASTA file from pileupped reads to submit WGS division of DDBJ.**  
Set filtered length for contigs  
 perl lengthfilter.pl pileupFile [300] x out\_WGS.txt

At the bottom right, there are 'BACK' and 'NEXT' buttons. A red arrow with the number '1' points to the 'NEXT' button.

# 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見える。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

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

- Job info:** ID 21211, Tool (Version) Platanus (1.2.2).
- Download:** QC.1.trimmed.fastq.gz (QC.1.trimmed.fastq.gz), Read length N.A. bp, Alias L.hokka.
- Download modified queries:** QC.1.trimmed.fastq.gz (Original size 189.4 MB), QC.2.trimmed.fastq.gz (Original size 189.6 MB).
- Download wgs file:** out\_WGS.fasta.gz (Original size 2.3 MB).
- Assembly statistics:** Contig #: 117, Total contig size: 2,356,019, Maximum contig size: 257,728, Minimum contig size: 101, N50 contig size: 92,304.
- Time:** Wait time 0:0:9, Start time 2016-01-20 18:33:36, End time 2016-01-21 10:10:06.
- Command Log:** A table with columns: Command, Start time, End time, Log1, Log2, Result, MD5. The first row is highlighted with a red box and a red arrow labeled '1'. The second row has a red arrow labeled '2' pointing to the 'View' link.

**Command Log Table:**

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



# 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

**ANALYSIS**

Data setup

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

step-1

- Preprocessing
- Mapping / de novo Assembly

step-2

**Workflow**

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

**JOB STATUS**

step1. Preprocessing

step1. Mapping

step1. de novo Assembly

step2-All status

**HELP**

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

**Job info**

ID: 21211

Tool (Version): Platanus (1.2.2)

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

**Download modified queries**

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

**Download wgs file**

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

**Assembly statistics**

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

**Time**

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

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

BACK Top of page

# 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

**ANALYSIS**

Data setup

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

step-1

- Preprocessing
- Mapping / *de novo* Assembly

step-2

**Workflow**

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

**JOB STATUS**

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

**HELP**

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

**Job info**

ID: 21211

Tool (Version): Platanus (1.2.2)

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

**Download modified queries**

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

**Download wgs file**

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

**Assembly statistics**

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

**Time**

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

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

# W20-3: Velvetと比較

左表で調べた範囲のVelvet実行結果(W12-10)よりも、特にk値を意識せずに行った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

(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	ウェブ資料
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	1000000	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



Contig # : 117

Total contig size : 2,356,019

Maximum contig size : 257,728

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

http://p.ddbj.nig.ac.jp/pipeline/DetailView.do?query\_set\_ Detail view

### ANALYSIS

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

### JOB STATUS

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

### HELP

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

### Job info

ID: 21211

Tool (Version): Platanus (1.2.2)

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

#### Download modified queries

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

#### Download wgs file

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

#### Assembly statistics

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

#### Time

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

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

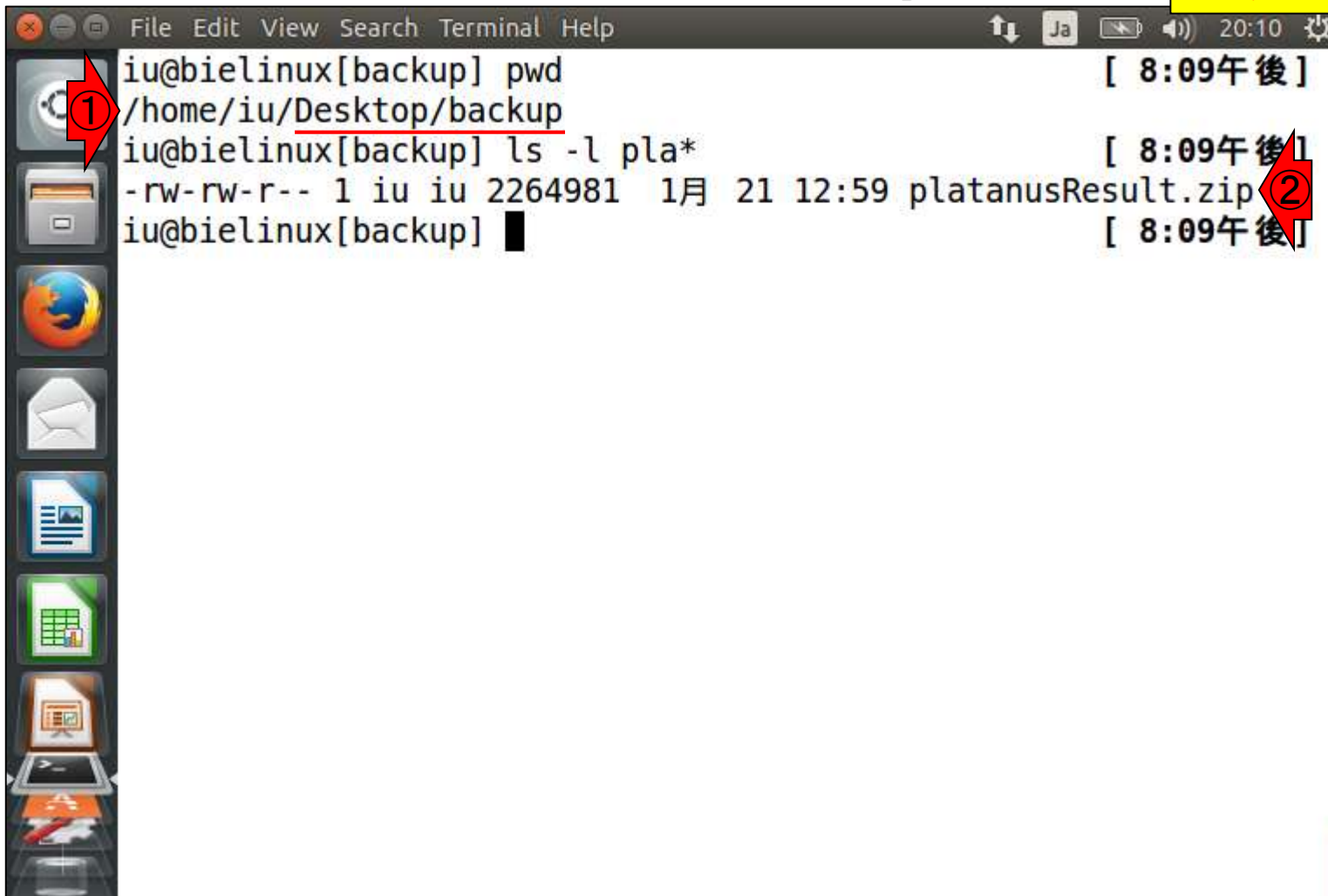
BACK Top of page





# W20-5: Bio-Linuxで解凍

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



```
iu@bielinux[backup] pwd [ 8:09午後 ]
/home/iu/Desktop/backup
iu@bielinux[backup] ls -l pla* [ 8:09午後 ]
-rw-rw-r-- 1 iu iu 2264981 1月 21 12:59 platanusResult.zip [ 8:09午後 ]
iu@bielinux[backup] █ [ 8:09午後 ]
```

# 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

```
iu@bielinux[backup] pwd [ 8:09午後 ]
/home/iu/Desktop/backup
iu@bielinux[backup] ls -l pla* [ 8:09午後 ]
-rw-rw-r-- 1 iu iu 2264981 1月 21 12:59 platanusResult.zip
iu@bielinux[backup] unzip -q platanusResult.zip [ 8:09午後 ]
iu@bielinux[backup] ls -ld pla* [ 8:13午後 ]
drwxrwxr-x 2 iu iu 4096 1月 21 10:09 platanusResult
-rw-rw-r-- 1 iu iu 2264981 1月 21 12:59 platanusResult.zip
iu@bielinux[backup] [ 8:14午後 ]
```

# 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

```
iu@bielinux[backup] pwd [ 8:09午後 ]
/home/iu/Desktop/backup
iu@bielinux[backup] ls -l pla* [ 8:09午後 ]
-rw-rw-r-- 1 iu iu 2264981 1月 21 12:59 platanusResult.zip
iu@bielinux[backup] unzip -q platanusResult.zip [ 8:09午後 ]
iu@bielinux[backup] ls -ld pla* [ 8:13午後 ]
drwxrwxr-x 2 iu iu 4096 1月 21 10:09 platanusResult
-rw-rw-r-- 1 iu iu 2264981 1月 21 12:59 platanusResult.zip
① iu@bielinux[backup] cd platanusResult [ 8:14午後 ]
iu@bielinux[platanusResult] pwd [ 8:21午後 ]
/home/iu/Desktop/backup/platanusResult
② iu@bielinux[platanusResult] ls [ 8:21午後 ]
out_32merFrq.tsv out_lib1_insFreq.tsv
out_contigBubble.fa out_scaffoldBubble.fa
out_contig.fa out_scaffoldComponent.tsv
③ out_gapClosed.fa out_scaffold.fa
iu@bielinux[platanusResult] [ 8:21午後 ]
```



# 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

```
iu@bielinux[platanusResult] pwd
/home/iu/Desktop/backup/platanusResult
iu@bielinux[platanusResult] ls -l *.fa
-rw-rw-r-- 1 iu iu 325 1月 21 10:08 out_contigBubble.fa
-rw-rw-r-- 1 iu iu 2436380 1月 21 10:08 out_contig.fa
-rw-rw-r-- 1 iu iu 2387680 1月 21 10:09 out_gapClosed.fa
-rw-rw-r-- 1 iu iu 0 1月 21 10:09 out_scaffoldBubble.fa
-rw-rw-r-- 1 iu iu 2389243 1月 21 10:09 out_scaffold.fa
iu@bielinux[platanusResult] grep -c ">" *.fa
out_contigBubble.fa:1
out_contig.fa:349
out_gapClosed.fa:117
out_scaffoldBubble.fa:0
out_scaffold.fa:117
iu@bielinux[platanusResult] █
```

```
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

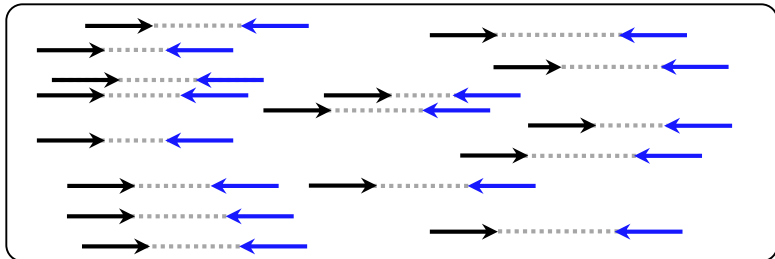
```
iu@bielinux[platanusResult] pwd [ 8:27午後 ]
/home/iu/Desktop/backup/platanusResult
iu@bielinux[platanusResult] ls -l *.fa [ 8:27午後 ]
-rw-rw-r-- 1 iu iu 325 1月 21 10:08 out_contigBubble.fa
-rw-rw-r-- 1 iu iu 2436380 1月 21 10:08 out_contig.fa
-rw-rw-r-- 1 iu iu 2387680 1月 21 10:09 out_gapClosed.fa
-rw-rw-r-- 1 iu iu 0 1月 21 10:09 out_scaffoldBubble.fa
-rw-rw-r-- 1 iu iu 2389243 1月 21 10:09 out_scaffold.fa
iu@bielinux[platanusResult] grep -c ">" *.fa [ 8:27午後 ]
out_contigBubble.fa:1
out_contig.fa:349
out_gapClosed.fa:117
out_scaffoldBubble.fa:0
out_scaffold.fa:117
iu@bielinux[platanusResult] █ [ 8:27午後 ]
```



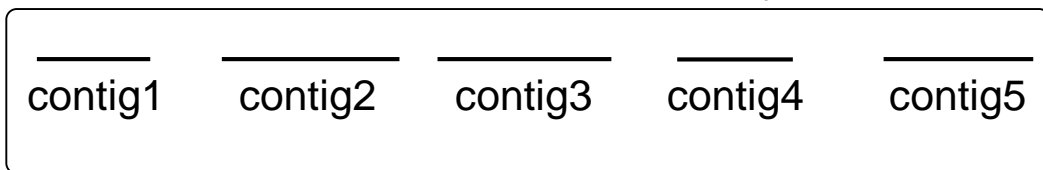
おさらい: *de novo*アセンブリの手順は大まかに3つのステップからなる

# *de novo*アセンブリ

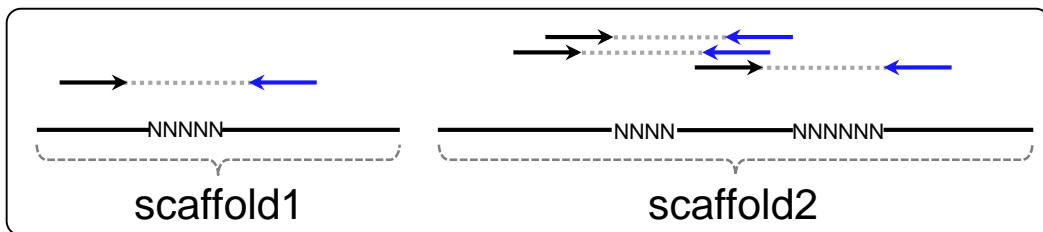
入力: paired-end FASTQファイル



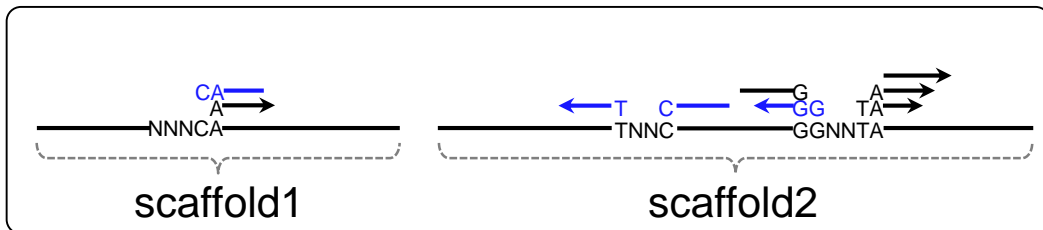
Step1: Assembly



Step2: Scaffold

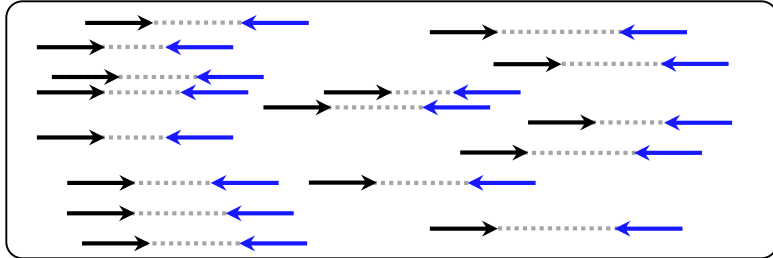


Step3: Gap close

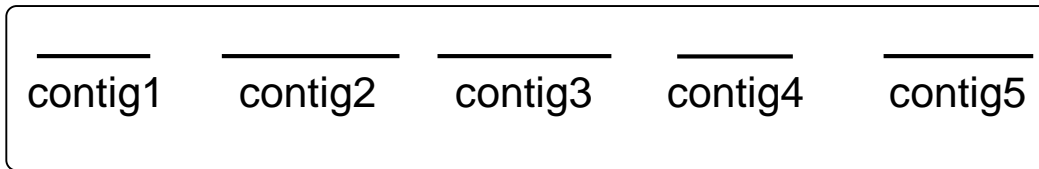


# de novoアセンブリ

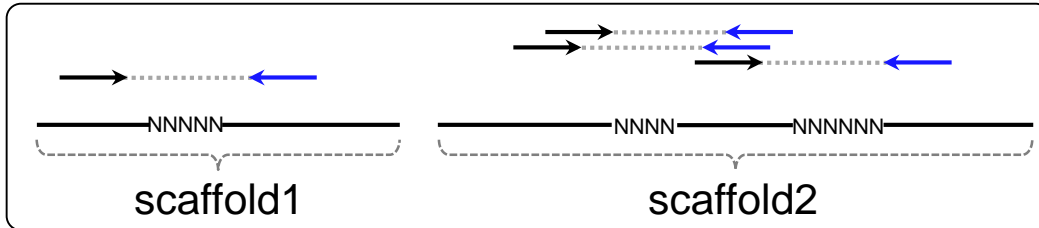
入力: paired-end FASTQファイル



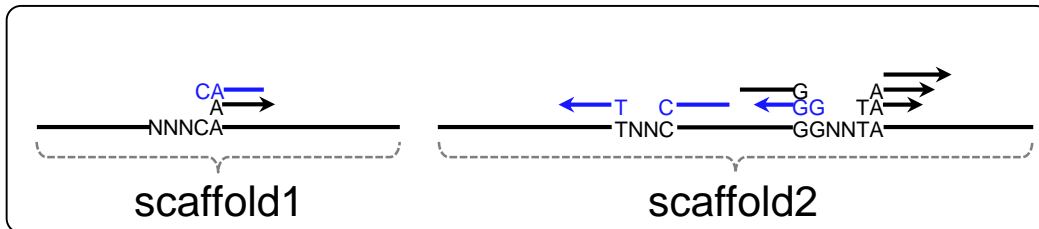
Step1: Assembly → out\_contig.fa (349配列) ①



Step2: Scaffold → out\_scaffold.fa (117配列) ②



Step3: Gap close → out\_gapClosed.fa (117配列) ③



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



# 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)と似た個数を得られたことがわかる

```
iu@bielinux[platanusResult] pwd [ 8:27午後 ]
/home/iu/Desktop/backup/platanusResult
iu@bielinux[platanusResult] ls -l *.fa [ 8:27午後 ]
-rw-rw-r-- 1 iu iu 325 1月 21 10:08 out_contigBubble.fa
-rw-rw-r-- 1 iu iu 2436380 1月 21 10:08 out_contig.fa
-rw-rw-r-- 1 iu iu 2387680 1月 21 10:09 out_gapClosed.fa
-rw-rw-r-- 1 iu iu 0 1月 21 10:09 out_scaffoldBubble.fa
-rw-rw-r-- 1 iu iu 2389243 1月 21 10:09 out_scaffold.fa
iu@bielinux[platanusResult] grep -c ">" *.fa [ 8:27午後 ]
out_contigBubble.fa:1
out_contig.fa:349
out_gapClosed.fa:117
out_scaffoldBubble.fa:0
out_scaffold.fa:117
① iu@bielinux[platanusResult] fastaLengthFilter.py out_gapClosed.
fa 300 > hoge.fa
② iu@bielinux[platanusResult] grep -c ">" hoge.fa [ 8:31午後 ]
52
iu@bielinux[platanusResult] █ [ 8:31午後 ]
```





# 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

```
iu@bielinux[platanusResult] pwd [ 8
/home/iu/Desktop/backup/platanusResult
iu@bielinux[platanusResult] ls -l *.fa [ 8
-rw-rw-r-- 1 iu iu 2347176 6月 16 20:31 hoge.fa
-rw-rw-r-- 1 iu iu 325 1月 21 10:08 out_contigBub
-rw-rw-r-- 1 iu iu 2436380 1月 21 10:08 out_contig.fa
-rw-rw-r-- 1 iu iu 2387680 1月 21 10:09 out_gapClosed.fa
-rw-rw-r-- 1 iu iu 0 1月 21 10:09 out_scaffoldBubble.fa
-rw-rw-r-- 1 iu iu 2389243 1月 21 10:09 out_scaffold.fa
① iu@bielinux[platanusResult] grep -v ">" out_gapClosed.fa | wc
29506 29506 2385525
② iu@bielinux[platanusResult] grep -v ">" hoge.fa | wc
52 52 2346552
iu@bielinux[platanusResult] █ [ 8:39午後]
```

# 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)
- 롱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




# 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が挿入されたことを確認したい。テンプレートとしては、「解析 | 一般 | [GC含量\(GC contents\)](#)」を用

```
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)
```



# 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

```
iu@bielinux[platanusResult] pwd [ 9:48午前 ]
/home/iu/Desktop/backup/platanusResult
iu@bielinux[platanusResult] ls *.fa [ 9:48午前 ]
hoge.fa out_contig.fa out_scaffoldBubble.fa
out_contigBubble.fa out_gapClosed.fa out_scaffold.fa
iu@bielinux[platanusResult] R -q [ 9:48午前 ]
> library(Biostrings) [ 9:48午前 ]
```





# 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

```
File Edit View Search Terminal Help
The following objects are masked from 'package:base':
  anyDuplicated, append, as.data.frame, as.vector, cbind, col
names,
  do.call, duplicated, eval, evalq, Filter, Find, get, inters
ect,
  is.unsorted, lapply, Map, mapply, match, mget, order, paste
, pmax,
  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
>
```

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

## • 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)
```

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

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

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

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

```
File Edit View Search Terminal Help
pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, repeat,
p.int,
rownames, sapply, setdiff, sort, table, tapply, union, unique,
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
>
```



# 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が認めたい。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 <- 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_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_contig.fa", format="fasta")
hoge <- alphabetFrequency(fasta)
obj <- is.element(colnames(hoge), c("A", "C", "G", "T", "N"))
colSums(hoge)[obj]
sum(hoge)
```



```
File Edit View Search Terminal Help
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
>
```





# 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が挿入されたことを確認したい。テンプレートとしては、「解析 | 一般 | [GC含量\(GC contents\)](#)」を用いた

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

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

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

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

```
File Edit View Search Terminal Help
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
>
```



# W21-2: ACGTカウン

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のカウン数を調べて、本当にscaffoldを認識したい。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 <- 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)
```

```
File Edit View Search Terminal Help
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
```





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

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

q(save="no")
```



```
File Edit View Search Terminal Help
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]

# 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)
- 롱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



# NGS連載第7回は

## (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、モジュール  
(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  
本日まで (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 2016/06/17) **NEW**
- 書籍 | 日本乳酸菌学会誌 | [第6回ゲノムアセンブリ](#) (last modified 2016/05/12)
- 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) **①**
- 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 | [任意の文字列を行の最後に挿入](#) (last modified 2014/07/17)
- 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)

## 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](#) (2016.06.17版、約200p) **②**
  - Macintosh用

### Linuxコマンド

- [ant-cache](#) (パッケージ調査)

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**

### 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

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

# おさらい(NGS連載第3回)

全部読んで説明

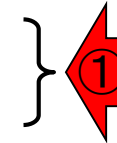
## ■ 3大公共DB(NGS分野)の提供ファイル形式

- 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)。



# 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: Pacific Biosciences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、シーケンス生データ。
  - 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/) (<http://pacbiofileformats.readthedocs.io/>) (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)

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



# おさらい

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

## ■ 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 (DRR024501 sub\_1.fastq.gz)
- reverse側 (DRR024501 sub\_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



# 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: Pacific Biosciences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、シーケンス生データ。
  - 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/) (<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)



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。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

## 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…



# 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) ②
- 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

① PacBio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



# 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

## ■ 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)



# この後の展開は…

## ■ 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)
- EMBL-EBI ENA (ENA): FASTQ形式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
- 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)。

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) の提供ファイル形式

- 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
- 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)。

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) の提供ファイル形式

- 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
- 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)。

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)
- 롱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

The screenshot shows the DRA Search web interface for run DRR054113. At the top, there are download buttons for FASTQ (marked with a red arrow ②) and SRA. Below this is a 'Run Detail' table with the following information:

Run Detail	
Alias	DRR054113
Instrument model	
Date of run	
Run center	
Number of spots	163,482
Number of bases	360,244,590

Below the table is a 'READS (joined)' section with a table of read statistics:

READS (joined)	qual	Time	Count	Download Link
>DRR054113.1		01/27/2016 02:32午後	2,392,259	<a href="#">DRR054113.fastq.bz2</a> (marked with red arrow ③)
CCTATGCTGTCAGCATTGATTGCTAGTTGAT		01/27/2016 02:32午後	2,901,836	<a href="#">DRR054114.fastq.bz2</a>
GCTACTAGTCTTGAGTCTGCCTGACTGATTGA		01/27/2016 02:33午後	3,858,646	<a href="#">DRR054115.fastq.bz2</a>
GTACGGCACGCATGTAGTACTGGTGCATGACC		01/27/2016 02:33午後	4,455,155	<a href="#">DRR054116.fastq.bz2</a>
>DRR054113.2				
TCATATACTCGGCACAATGTGTGTCGATCGTAAAGGGATGTCATTGTGTAGTATTGATTCTATATGTCGAGCATCAGCG				
TTCTACTGCTGAGATGATATATTCTGAGTATTATGGTTATGTATTTTACGCTGAACCTGGATTATGTCGTGGACGGACGT				
TGTACGGATTTCTAACTGTTAGTATCGAGCATTGATCGTCCGATGGATTGATAGTGCTTCCGTTGAGTCGTAATGATTGTT				
CAGTTAGTCGATCAGTTCTGGATCAGTGATTTTGTAGGCGAAGATTATGAATCTTTACGATCCTTATGGCTGAGTTGAA				
TGATCTGCTGCTAGTGTCTGTATGTTTCGTATGATCACATGACGATACGTGATATTTATTATTGTCTACGCATCGATTGAG				

On the right side of the interface, there is a 'Navigation' menu with links for 'Submissi...', 'Study DRP002401', and 'Experiment DRX022185'. The 'FASTQ' link under the experiment is highlighted with a red arrow ②.

# 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

② W3-1: DRR054113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
```

• 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！



```
iu@bielinux[iu] cd ~/Documents
iu@bielinux[Documents] pwd
/home/iu/Documents
iu@bielinux[Documents] mkdir DRR054113
iu@bielinux[Documents] cd DRR054113
iu@bielinux[DRR054113] pwd
/home/iu/Documents/DRR054113
iu@bielinux[DRR054113] wget -cq ftp://ftp.ddbj.nig.ac.jp/ddbj_d
atabase/dra/fastq/DRA002/DRA002643/DRX022185/DRR054113.fastq.bz
2
iu@bielinux[DRR054113] ls -l
total 2340
-rw-rw-r-- 1 iu iu 2392259  3月 22 12:32 DRR054113.fastq.bz2
iu@bielinux[DRR054113] ls -lh
total 2.3M
-rw-rw-r-- 1 iu iu 2.3M  3月 22 12:32 DRR054113.fastq.bz2
iu@bielinux[DRR054113]
```



[12:32午後]  
[12:32午後]  
[12:32午後]

[12:32午後]  
[12:33午後]  
[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

```
iu@bielinux[DRR054113] pwd
/home/iu/Documents/DRR054113
iu@bielinux[DRR054113] ls -l
total 2340
-rw-rw-r-- 1 iu iu 2392259  3月 22 12:32 DRR054113.fastq.bz2
iu@bielinux[DRR054113] fastqc2 -v
FastQC v0.11.4
iu@bielinux[DRR054113] fastqc2 -q DRR054113.fastq.bz2 --outdir=
/home/iu/Desktop/mac_share
iu@bielinux[DRR054113] ls -l ~/Desktop/mac_share
total 751
-rwxrwxrwx 1 iu iu 392406  3月 22 12:35 DRR054113_fastqc.html
-rwxrwxrwx 1 iu iu 375781  3月 22 12:35 DRR054113_fastqc.zip
iu@bielinux[DRR054113]
```



# 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

The screenshot shows a Windows desktop environment. On the left, a File Explorer window is open to a 'share' folder. Two red arrows labeled '1' and '2' point to the 'share' folder icon and the files 'DRR054113\_fastqc.html' and 'DRR054113\_fastqc.zip' respectively. On the right, a web browser window displays the 'FastQC Report' for the file 'DRR054113.fastq.bz2'. The report includes a 'Summary' section with a list of metrics and a 'Basic Statistics' table.











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

# 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 
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	915 
Sequences flagged as poor quality	0
Sequence length	923-8076 
%GC	38

### Per base sequence quality

Quality scores across all bases (Sanger / Illumina 1.9 encoding)



# W3-3: 結果を眺める

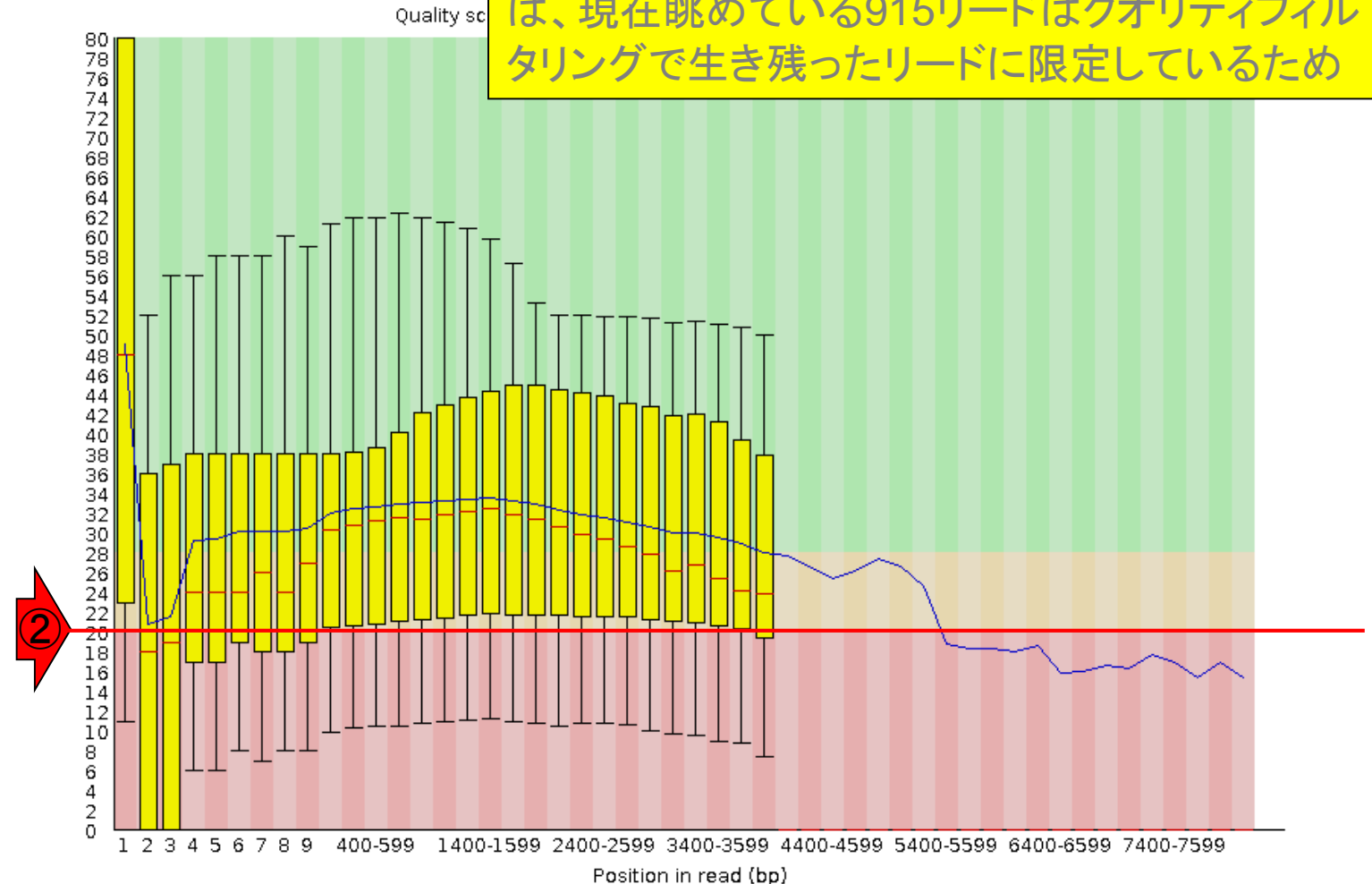
① Per base sequence quality. この図の縦軸はクオリティスコア。② 赤線のスコア20を超えているかどうか1つの目安。Illumina HiSeq2000 (第4回W8)やMiSeq (第6回W4)とは傾向が異なる。第7回W6では詳述しているが、PacBioの公称スペックよりも全体的なクオリティが高い。この理由は、現在眺めている915リードはクオリティフィルタリングで生き残ったリードに限定しているため

## 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

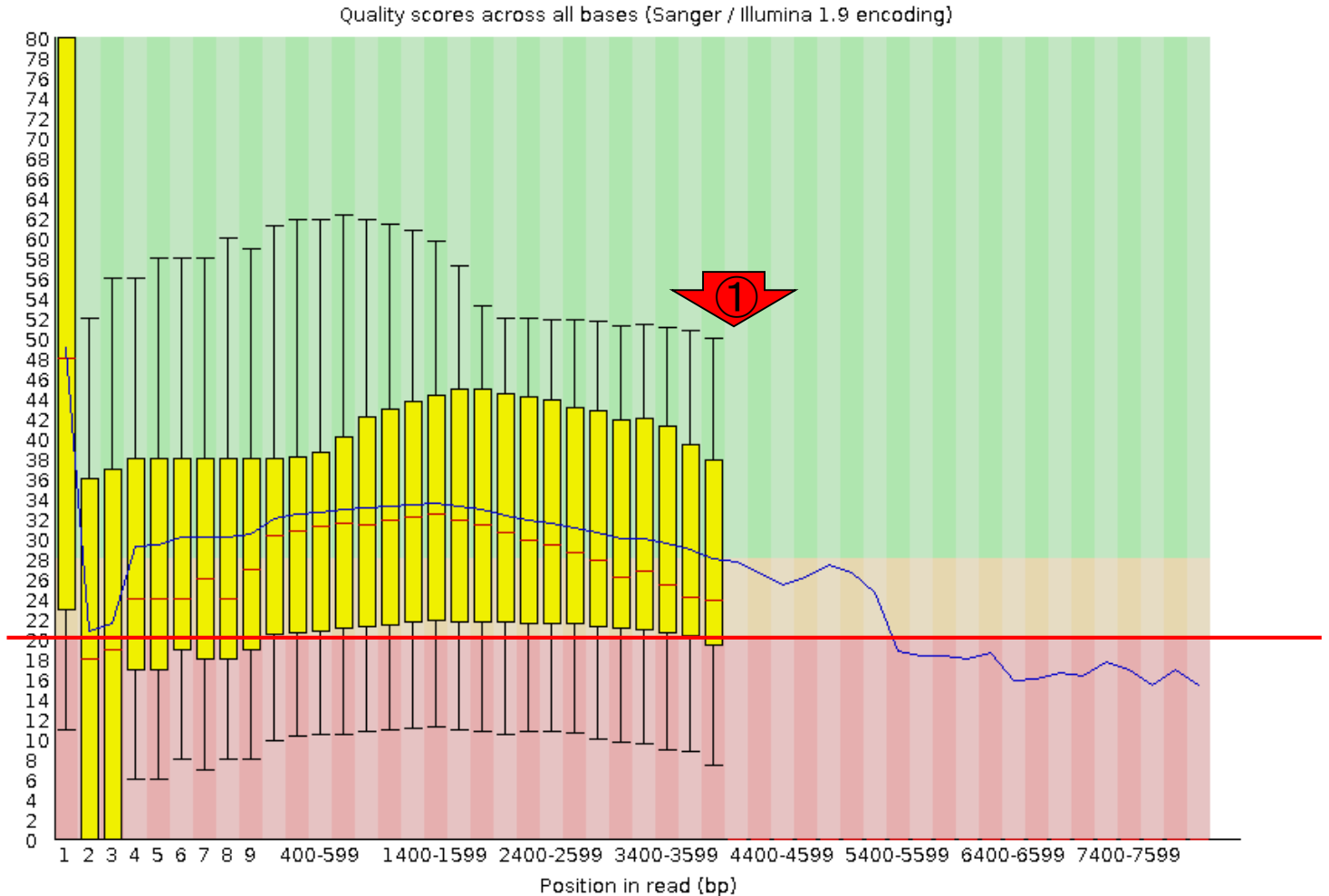
①横軸のリードポジションが4000 bpあたりで黄色の縦棒がなくなっている。この理由は、4000 bp以上のリードが少数だからだと思われる。それは②の配列長分布(Sequence Length Distribution)でも確認できる

## 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

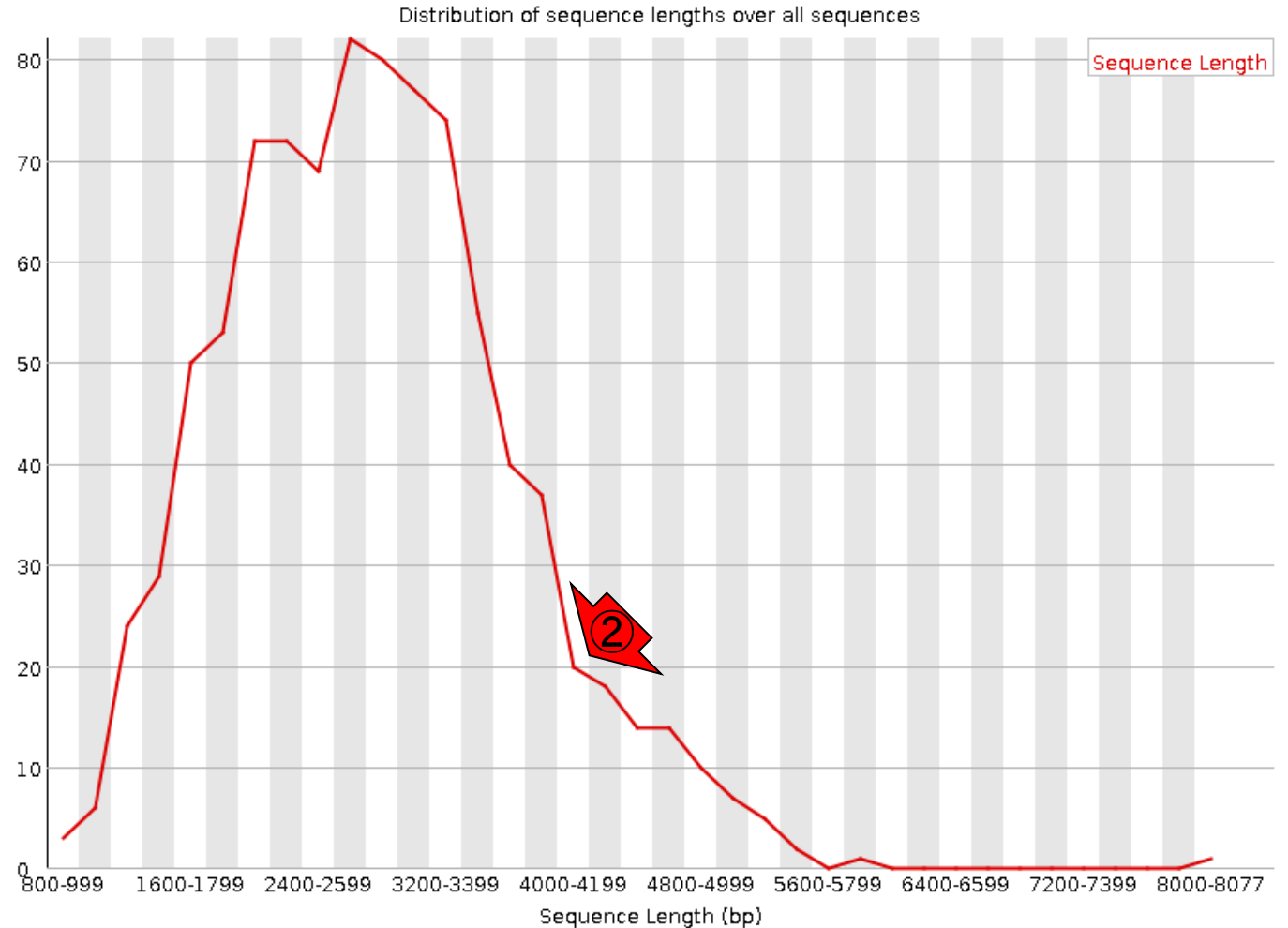
①配列長分布(Sequence Length Distribution)。②このあたりで黄色の縦棒がなくなっているのので、おそらく20リードが黄色の縦棒の有無の閾値なのだろう

## 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)
- 롱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



# NGS連載第7回の...

## (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  
本います。(2015/4

- 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Q](#)
- 書籍 | 日本乳酸菌学会誌 | [第6回ゲノムアセンブリ](#) (last modified 2014/07/17)
- 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 2016/06/03)
- 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 | [任意の文字列を行の最後に挿入](#) (last modified 2014/07/17)
- 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)

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います



## 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
  - Windows
  - Macintosh

### Linuxコマンド

- apt-cache (パッケージ検索)

### ② SRA Toolkit (ver. 2.5.7)のインストールと利用

- W4-1: [SRA Toolkit](#) (ver. 2.5.7)のインストールと利用  
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

The screenshot shows the NCBI SRA Toolkit download page. The browser address bar shows the URL: <http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software>. The page title is "Sequence Read Archive". The navigation menu includes "Main", "Browse", "Search", "Download", "Submit", "Documentation", "Software", "Trace Archive", "Trace Assembly", "Trace Home", and "Trace BLAST". The "Download" section is active, showing "Toolkit Documentation" and "XML Schema". The main heading is "SRA Toolkit". Below the heading, there is a link for documentation: "For Toolkit documentation [click here](#)". A red arrow with the number 4 points to this link. The first main item is "1. NCBI SRA Toolkit latest release (December 23 2015, version 2.5.7 release) compiled binaries and [md5 checksums](#)\*:". A red arrow with the number 1 points to this item. Underneath, there is a list of operating systems: "CentOS Linux 64 bit architecture", "Ubuntu Linux 64 bit architecture", "MacOS 64 bit architecture", and "MS Windows 64 bit architecture". A red box highlights this list, with a red arrow and the number 3 pointing to it. A red arrow with the number 2 points to the "Ubuntu Linux 64 bit architecture" link. Below this list is the text: "• [vdb-view Windows Installer](#) is a spreadsheet-like browser for viewing SRA and vdb objects - Windows only". The second main item is "2. NCBI Decryption Tools latest release binaries and [md5 checksums](#)\*:". Underneath, there is a list of operating systems: "CentOS Linux 64 bit architecture", "CentOS Linux 32 bit architecture", "Ubuntu Linux 64 bit architecture", "Ubuntu Linux 32 bit architecture", "MacOS 64 bit architecture", "MacOS 32 bit architecture", "MS Windows 64 bit architecture", and "MS Windows 32 bit architecture". The third main item is "3. Latest Source Code:", followed by a list: "1. [NGS Software Development Kit](#) – November 24 2015, version 1.2.3 release", "2. [NCBI VDB Software Development Kit](#) – December 23 2015, version 2.5.7 release", and "3. [NCBI SRA Toolkit](#) – December 23 2015, version 2.5.7 release".

# 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

The screenshot shows the NCBI Sequence Read Archive website. The browser address bar displays [http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit\\_doc](http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc). The page title is "SRA Toolkit Documentation". The navigation menu includes "Main", "Browse", "Search", "Download", "Submit", "Documentation", "Software", "Trace Archive", "Trace Assembly", "Trace Home", and "Trace BLAST". The "Download" section is active, showing "Toolkit Documentation" and "XML Schema".

**SRA Toolkit Documentation**

[SRA Toolkit Installation and Configuration Guide](#) ③

[Protected Data Usage Guide](#) ②

**Frequently Used Tools:**

[fastq-dump](#): Convert SRA data into fastq format ①

[prefetch](#): Allows command-line downloading of SRA, dbGaP, and ADSP data

[sam-dump](#): Convert SRA data to sam format

[sra-pileup](#): Generate pileup statistics on aligned SRA data

[vdb-config](#): Display and modify VDB configuration information

[vdb-decrypt](#): Decrypt non-SRA dbGaP data ("phenotype data")

**Additional Tools:**

[abi-dump](#): Convert SRA data into ABI format (csfasta / qual)

[illumina-dump](#): Convert SRA data into Illumina native formats (qseq, etc.)

[sff-dump](#): Convert SRA data to sff format

[sra-stat](#): Generate statistics about SRA data (quality distribution, etc.)

[vdb-dump](#): Output the native VDB format of SRA data.

[vdb-encrypt](#): Encrypt non-SRA dbGaP data ("phenotype data")

[vdb-validate](#): Validate the integrity of downloaded SRA data

# 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

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](mailto: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](http://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"
```

Unpack the Toolkit:

1. For Linux, use tar:

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

# 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

```
File Edit View Search Terminal Help
iu@bielinux[DRR054113] pwd
/home/iu/Documents/DRR054113
iu@bielinux[DRR054113] apt-cache -n search SRA [ 8:16午後 ]
libratom-0-0 - library for serialising LV2 atoms to/from Turtle
libratom-dev - library for serialising LV2 atoms to/from Turtle
- development files
libratom-doc - library for serialising LV2 atoms to/from Turtle
- documentation
sra-toolkit - utilities for the NCBI Sequence Read Archive
sra-toolkit-libs-dev - Development files for the NCBI SRA Toolkit's libraries
sra-toolkit-libs0 - Libraries for the SRA Toolkit
iu@bielinux[DRR054113] [ 8:16午後 ]
```





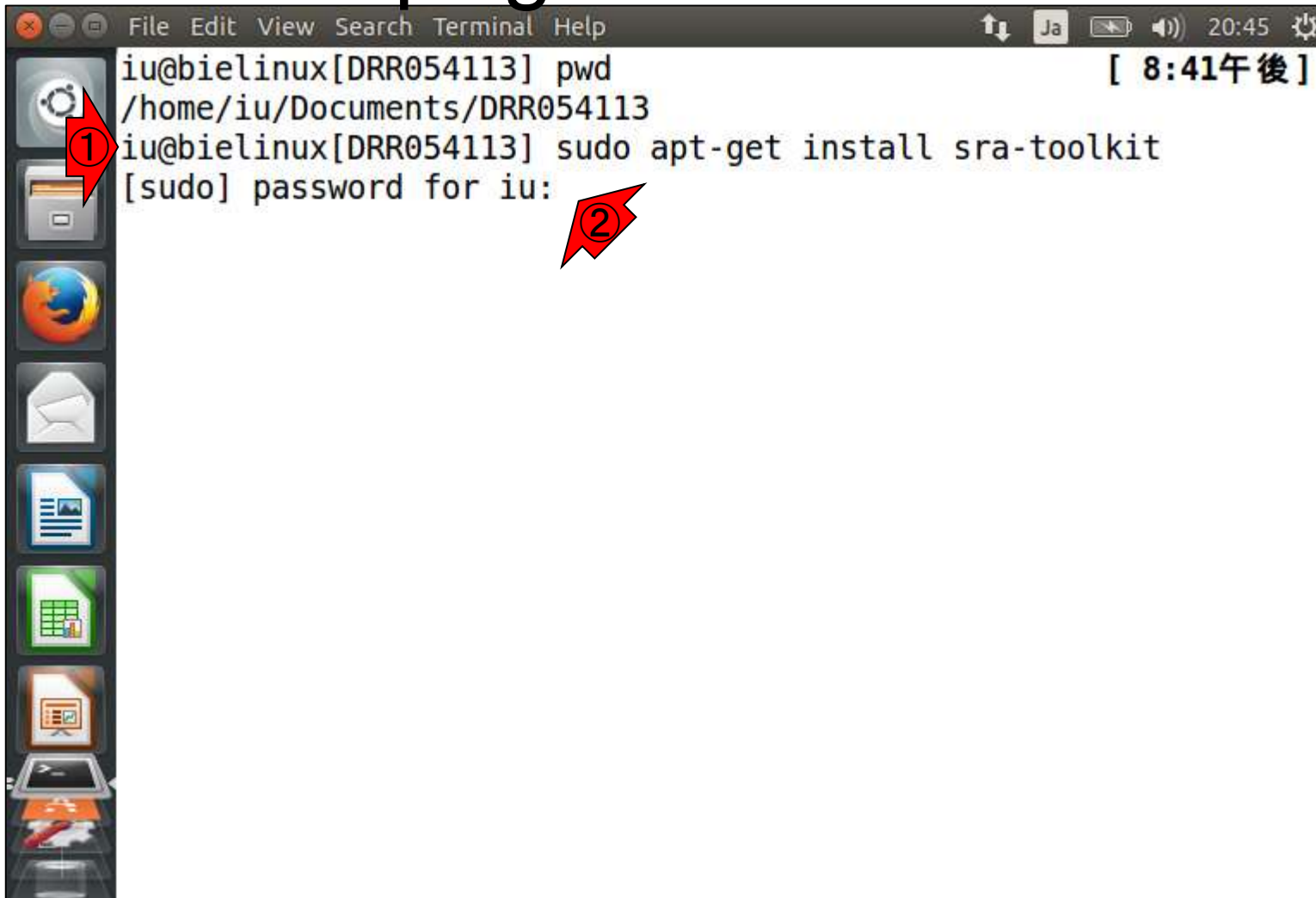
# W4-2: apt-cache

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

```
File Edit View Search Terminal Help
iu@bielinux[DRR054113] pwd
/home/iu/Documents/DRR054113
① iu@bielinux[DRR054113] apt-cache -n search SRA [ 8:16午後 ]
libratom-0-0 - library for serialising LV2 atoms to/from Turtle
libratom-dev - library for serialising LV2 atoms to/from Turtle
- development files
libratom-doc - library for serialising LV2 atoms to/from Turtle
- documentation
sra-toolkit - utilities for the NCBI Sequence Read Archive
sra-toolkit-libs-dev - Development files for the NCBI SRA Toolkit's libraries
sra-toolkit-libs0 - Libraries for the SRA Toolkit
② iu@bielinux[DRR054113] apt-cache -n search SRA | wc [ 8:16午後 ]
      6      58     418
iu@bielinux[DRR054113] [ 8:29午後 ]
```

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



The image shows a terminal window with the following text and annotations:

```
iu@bielinux[DRR054113] pwd  
/home/iu/Documents/DRR054113  
① iu@bielinux[DRR054113] sudo apt-get install sra-toolkit  
[sudo] password for iu: ②
```

The terminal window has a title bar with "File Edit View Search Terminal Help" and system icons for volume, network, and time (20:45). The system language is set to Japanese (Ja). The time is 8:41 PM. The terminal shows the user's current directory, then the command to install sra-toolkit, and the password prompt. Red arrows with numbers 1 and 2 point to the command and the password prompt respectively.

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

```
iu@bielinux[~/Documents/DRR054113] 20:48
Reading state information... Done
The following packages were automatically installed and are no longer required:
linux-headers-3.13.0-55 linux-headers-3.13.0-55-generic
linux-headers-3.13.0-68 linux-headers-3.13.0-68-generic
linux-headers-3.13.0-71 linux-headers-3.13.0-71-generic
linux-image-3.13.0-55-generic linux-image-3.13.0-68-generic
linux-image-3.13.0-71-generic linux-image-extra-3.13.0-55-generic
linux-image-extra-3.13.0-68-generic linux-image-extra-3.13.0-71-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
sra-toolkit-libs0
The following NEW packages will be installed:
sra-toolkit sra-toolkit-libs0
0 upgraded, 2 newly installed, 0 to remove and 138 not upgraded.
Need to get 2,311 kB of archives.
After this operation, 6,065 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```





# W4-3: apt-get

```
used.
Do you want to continue? [Y/n] y
Get:1 http://jp.archive.ubuntu.com/ubuntu/pool/lib/sra-toolkit-libs0 amd64 2.1.7a-1ubuntu2 [924 kB]
Get:2 http://jp.archive.ubuntu.com/ubuntu/pool/lib/sra-toolkit amd64 2.1.7a-1ubuntu2 [1,387 kB]
Fetched 2,311 kB in 0s (5,838 kB/s)
Selecting previously unselected package sra-toolkit-libs0.
(Reading database ... 439956 files and 409619 directories currently installed.)
Preparing to unpack .../sra-toolkit-libs0_2.1.7a-1ubuntu2_amd64.deb ...
Unpacking sra-toolkit-libs0 (2.1.7a-1ubuntu2) ...
Selecting previously unselected package sra-toolkit.
Preparing to unpack .../sra-toolkit_2.1.7a-1ubuntu2_amd64.deb ...
Unpacking sra-toolkit (2.1.7a-1ubuntu2) ...
Setting up sra-toolkit-libs0 (2.1.7a-1ubuntu2) ...
Setting up sra-toolkit (2.1.7a-1ubuntu2) ...
Processing triggers for libc-bin (2.19-0ubuntu6.7) ...
iu@bielinux[DRR054113] █ [ 8:48午後 ]
```

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

```
File Edit View Search Terminal Help
1 iu@bielinux[DRR054113] pwd
  /home/iu/Documents/DRR054113
2 iu@bielinux[DRR054113] fastq-dump [12:31午後]

Usage:
fastq-dump [options] [ -A ] <accession>
fastq-dump [options] <path [path...]>

Use option --help for more information

fastq-dump : 2.1.7

3 iu@bielinux[DRR054113] where fastq-dump [12:31午後]
  /usr/bin/fastq-dump
iu@bielinux[DRR054113] [12:31午後]
```

# 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



### 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](mailto:sra-tools@ncbi.nlm.nih.gov)

The following guide will outline the steps for the installation and configuration of the SRA Toolkit. For information on the usage of individual tools in the SRA Toolkit, see the SRA Toolkit User Guide.

The NCBI SRA Toolkit can be used to process data in the .sra format (Note that the .sra format may be compiled with GC content). For more information, see the SRA Toolkit User Guide. We recommend using these procedures.

### Downloading and installing the SRA Toolkit

Download the Toolkit from the SRA Toolkit website.

1. If you are using a web browser, click on the supported platform link to download the Toolkit.
2. If you are instead using a command-line utility, use the following command to download the Toolkit to the directory: `"/ftp-trace.ncbi.nlm.nih.gov/sra/sdk/1.18.0/linux64"`  

```
wget "/ftp-trace.ncbi.nlm.nih.gov/sra/sdk/1.18.0/linux64/sratoolkit.current-centos_linux64.tar.gz"
```

### Unpack the Toolkit:

1. For Linux, use tar:  

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



### 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.

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 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)
- 롱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

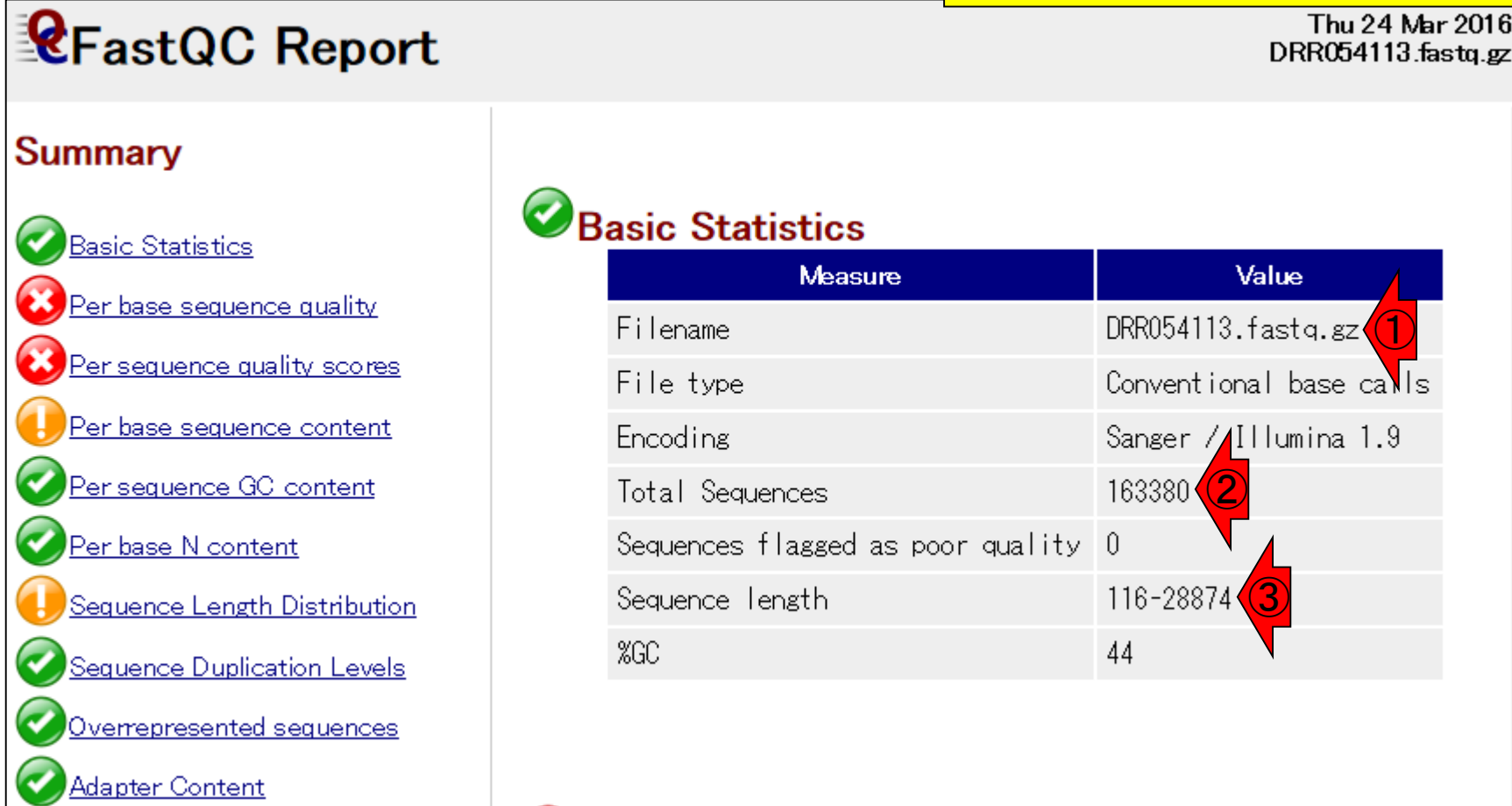
```
File Edit View Search Terminal Help
/home/iu/Documents/DRR054113
iu@bielinux[DRR054113] ls
DRR054113_DRA.fastq.bz2  DRR054113.fastq
DRR054113_DRA.fastq.gz  DRR054113.sra
iu@bielinux[DRR054113] rm -f *.fastq
iu@bielinux[DRR054113] ls -l
total 1389816
-rw-rw-r-- 1 iu iu    2392259  3月 22 12:32 DRR054113_DRA.fastq.bz2
-rw-rw-r-- 1 iu iu    2720482  3月 22 12:38 DRR054113_DRA.fastq.gz
-rw-rw-r-- 1 iu iu 1418046334  3月 22 15:23 DRR054113.sra
iu@bielinux[DRR054113] fastq-dump ./DRR054113.sra [ 5:14午後]
Written 163380 spots for ./DRR054113.sra
Written 163380 spots total
iu@bielinux[DRR054113] ls -l [ 5:15午後]
total 2102944
-rw-rw-r-- 1 iu iu    2392259  3月 22 12:32 DRR054113_DRA.fastq.bz2
-rw-rw-r-- 1 iu iu    2720482  3月 22 12:38 DRR054113_DRA.fastq.gz
-rw-rw-r-- 1 iu iu  730238332  3月 23 17:15 DRR054113.fastq
-rw-rw-r-- 1 iu iu 1418046334  3月 22 15:23 DRR054113.sra
iu@bielinux[DRR054113] wc DRR054113.fastq [ 5:15午後]
 653520   980280 730238332 DRR054113.fastq
iu@bielinux[DRR054113] █ [ 5:19午後]
```





# 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



The screenshot shows a FastQC report for the file DRR054113.fastq.gz. The report is dated Thu 24 Mar 2016. On the left, a 'Summary' section lists various metrics with status icons: Basic Statistics (green check), Per base sequence quality (red X), Per sequence quality scores (red X), Per base sequence content (orange exclamation), Per sequence GC content (green check), Per base N content (green check), Sequence Length Distribution (orange exclamation), Sequence Duplication Levels (green check), Overrepresented sequences (green check), and Adapter Content (green check). The main content area is titled 'Basic Statistics' with a green check icon. It contains a table with two columns: 'Measure' and 'Value'. The table rows are: Filename (DRR054113.fastq.gz), File type (Conventional base calls), Encoding (Sanger / Illumina 1.9), Total Sequences (163380), Sequences flagged as poor quality (0), Sequence length (116-28874), and %GC (44). Red arrows with circled numbers 1, 2, and 3 point to the Filename, Total Sequences, and Sequence length rows, respectively.

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
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	163380
Sequences flagged as poor quality	0
Sequence length	116-28874
%GC	44

# W6-3: 結果を眺める

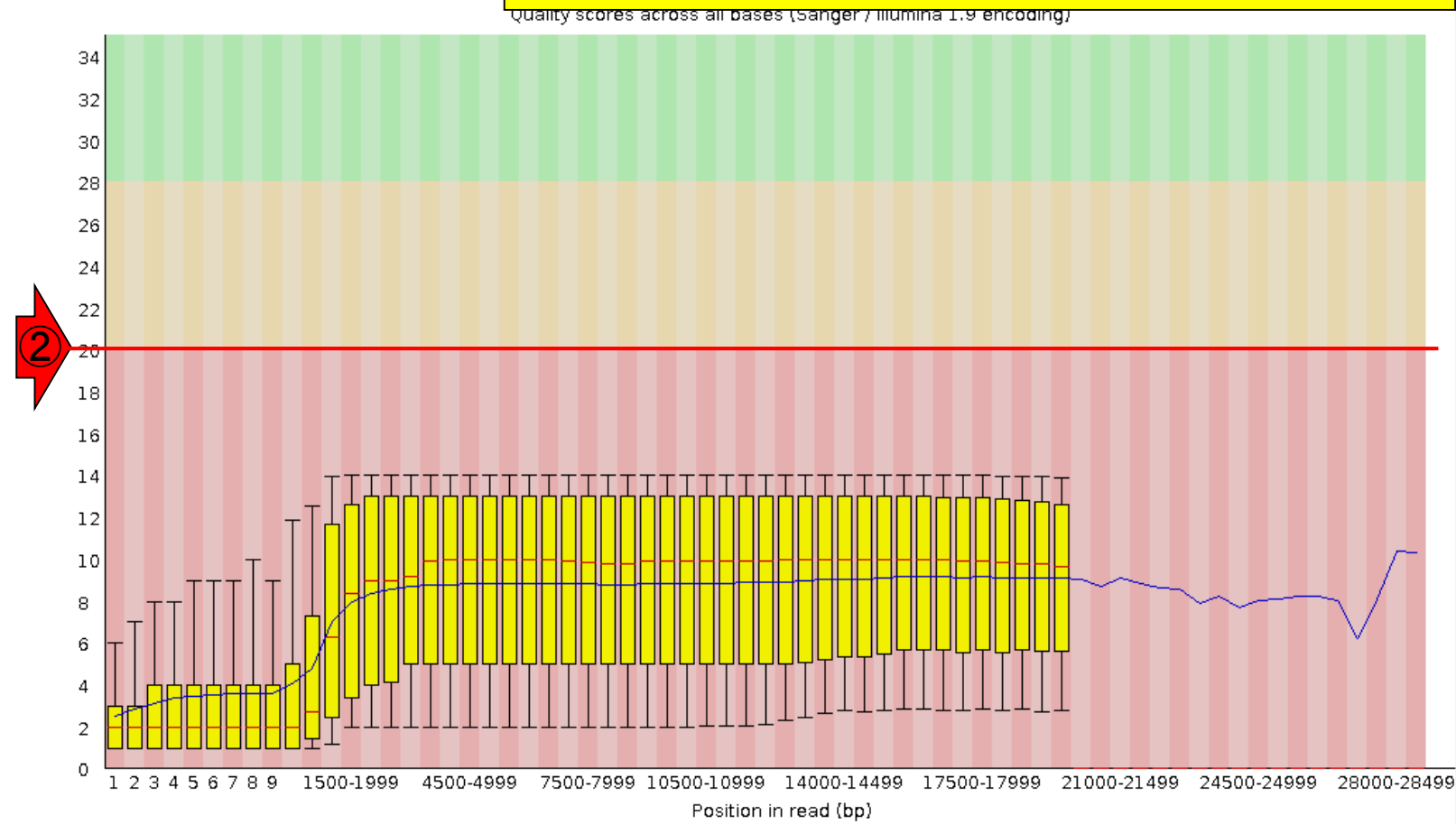
① Per base sequence quality. この図の縦軸はクオリティスコア。② 赤線のスコア20を基準としてみると、DRAから直接ダウンロードした915リードからなるFASTQファイルのFastQC実行結果(W3-3)と比べて、明らかに低くなっていることがわかる。クオリティスコアは、平均で10弱程度

## 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

```
iu@bielinux[DRR054113] R -q [ 4:32午後]
> -10*log10(0.01) #エラー率1%のときのクオリティスコア
[1] 20
> -10*log10(0.10) #エラー率10%のときのクオリティスコア
[1] 10
① > -10*log10(0.13) #エラー率13%のときのクオリティスコア
[1] 8.860566
> q(save="no")
iu@bielinux[DRR054113] [ 4:33午後]
```

# 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)
- 롱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

## PacBioのファイル形式とデータ解析の概要

- W1-1: Pacific Biosciences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、シーケンス生データ。
  - 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)



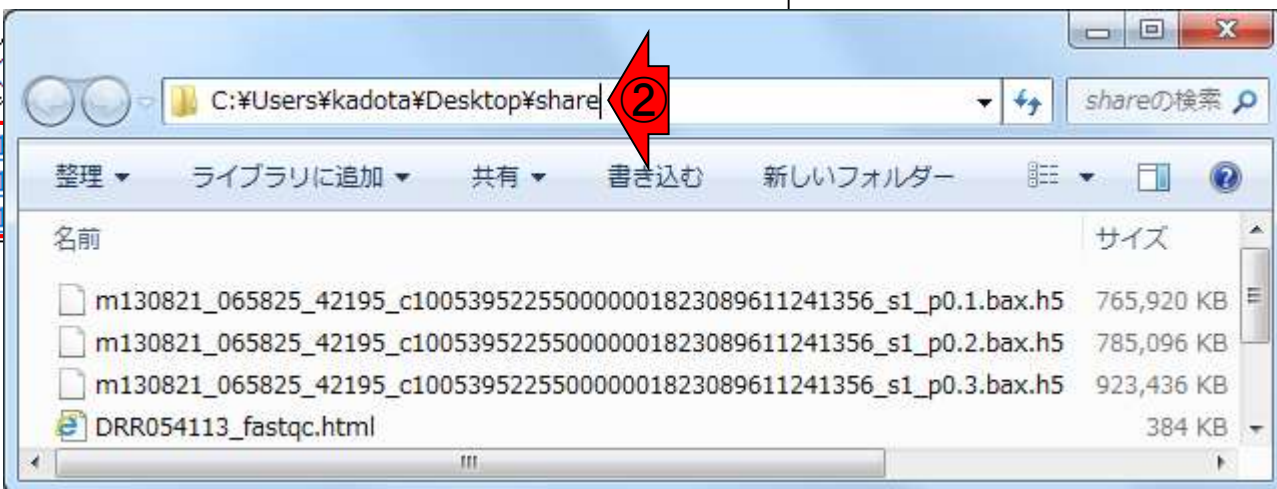
おさらい。DDBJ Pipeline上では、PacBio用の *de novo* アセンブリ用プログラムHGAPを利用可能。しかし、①HGAPはbax.h5ファイルのみしか受け付けない

# W7-1: bax.h5ファイル準備

## PacBioのファイル形式とデータ解析の概要

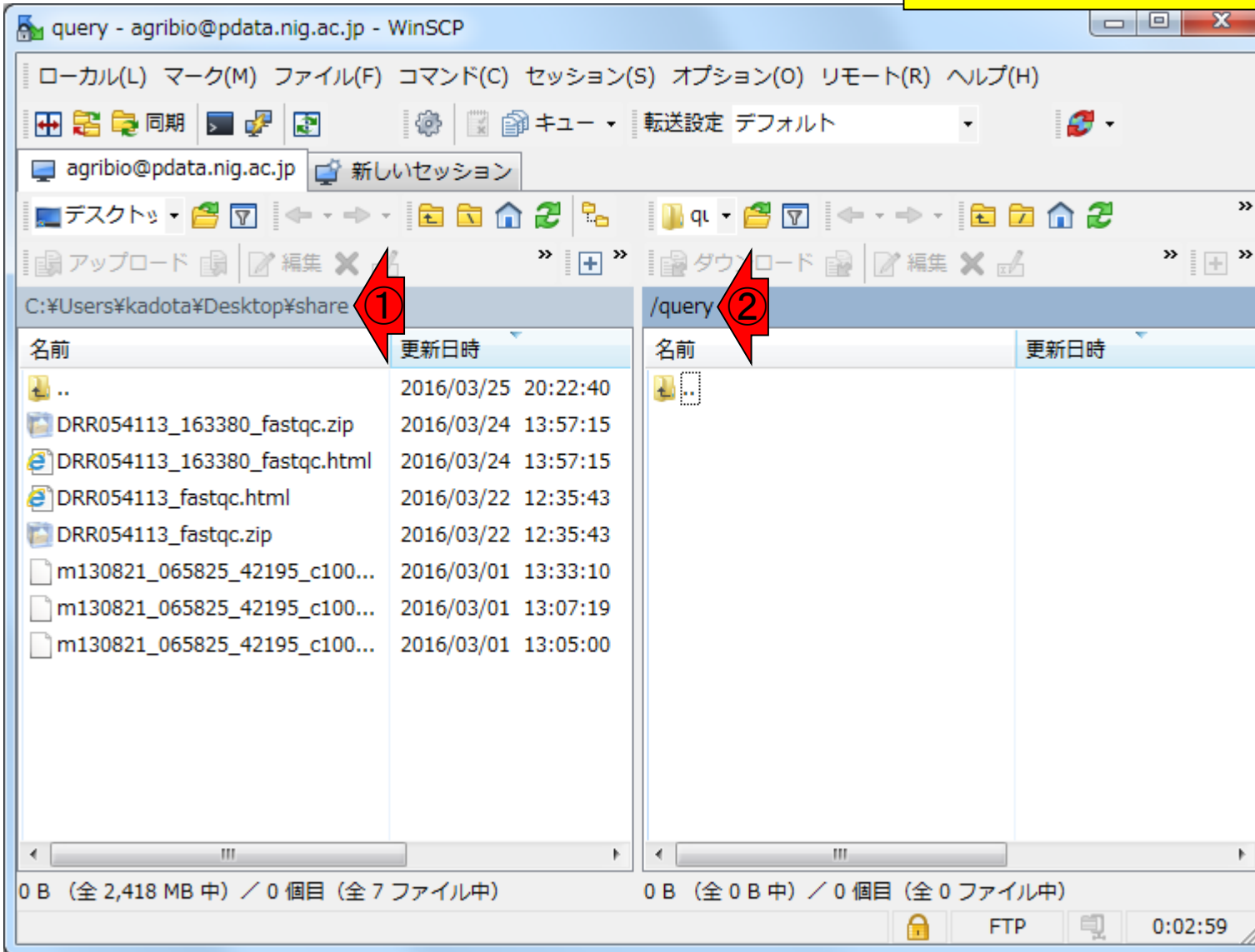
- W1-1: PacificBiosciencesのYouTubeサイト
  - [Introduction to SMRT Sequencing](#)
  - [Single Molecule Real Time Sequencing](#)
- W2-1: PacBioデータ(原著論文中のDRR IDだが削除されている)
  - [DRR024500: Tanizawa et al., BMC Genomics, 2015](#)
- W2-3: [DRR024501](#) -> [DRP002401](#) -> [DRX022185](#)
- W2-4: [DRR024501](#) -> [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、シーケンス生データ。
  - PacBio RSIIの後継機である [Sequel](#) の出力ファイル
  - 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ファイル合計)
  - [m130821\\_065825\\_42195\\_c100539522550000001823089611241356\\_s1\\_p0.1.bax.h5](#)
  - [m130821\\_065825\\_42195\\_c100539522550000001823089611241356\\_s1\\_p0.2.bax.h5](#)
  - [m130821\\_065825\\_42195\\_c100539522550000001823089611241356\\_s1\\_p0.3.bax.h5](#)

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！



# W7-2: アップロード

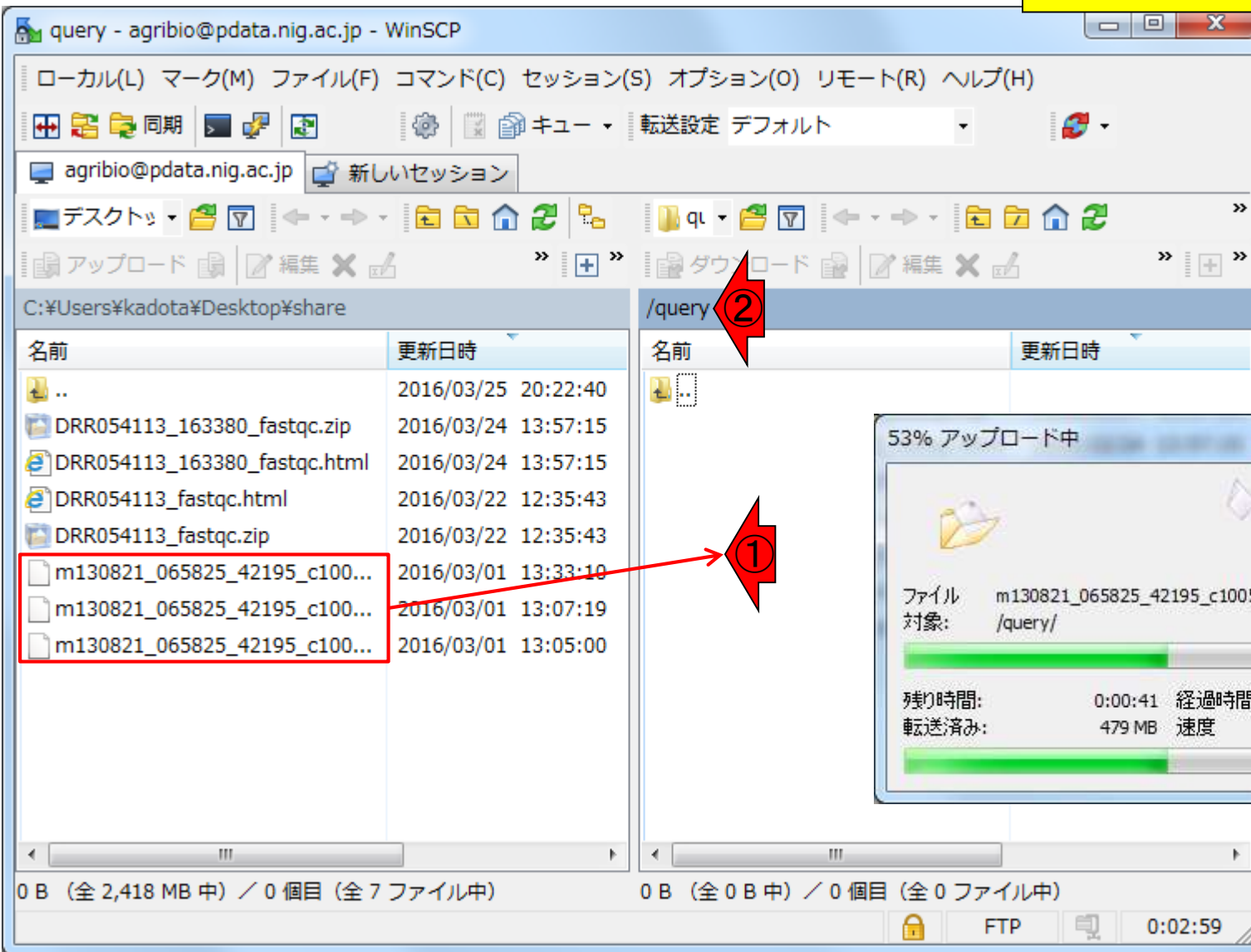
WinSCPを用いてDDBJ Pipelineの実体である遺伝研スパコンにログインし、①「デスクトップ - share」、②queryフォルダに移動した状態





# 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



# W8-1: HGAP実行

Preprocessing Start

step-1  
Preprocessing

Mapping / de novo Assembly

step-2

**Workflow**

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

**JOB STATUS**

step1. Preprocessing

step1. Mapping

step1. de novo Assembly

step2-All status

**HELP**

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

Tool	Help	Version	Base space	Color space	Paired-end	MSS (WGS)	Comment
<input type="checkbox"/> SOAPdenovo		2.04-r240	✓		✓		
<input type="checkbox"/> ABySS		1.3.2	✓		✓		Maximum K-mer value is 64.
<input type="checkbox"/> Velvet		1.2.10	✓		✓	✓	We severe recommend when performing Velvet, total length of those reads is up to 22G bp.Maximum K-mer value is 64.
<input type="checkbox"/> Trinity		2.1.1	✓		✓		RNA-Seq De novo Assembly
<input type="checkbox"/> Platanus		1.2.2	✓		✓		
<input checked="" type="checkbox"/> HGAP		Protocol3 (v 2.2.0)					HGAP Pipeline for PacBio Sequence based on SMRT Analysis v2.2.0. For bax.h5 file only. (Beta version)

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

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

BACK NEXT

# W8-1: HGAP実行

Generating Query Sets from Query Read Files

Single analysis  
Layout of single sequence.

5' 3'  
Linker(1) Target Linker(2)

	Run ACCESSION	Read length	Quality Score
<input checked="" type="checkbox"/>	m130821_065825_42195_c10053952255000001823089611241356_s1_p0.3.bax.h5	bp	
<input checked="" type="checkbox"/>	m130821_065825_42195_c10053952255000001823089611241356_s1_p0.2.bax.h5	bp	
<input checked="" type="checkbox"/>	m130821_065825_42195_c10053952255000001823089611241356_s1_p0.1.bax.h5	bp	

confirm

QUERY SET

# W8-2: 2つのパラメータ

Setting for De Novo Assembly

hgap

Set optional parameters for HGAP pipeline

Select UGE-node to run :

- month\_fat (32 CPUs and 320GB memory)
- month\_medium (32 CPUs and 256GB memory)

Same results will be generated with either option.  
You can check the CPU and memory usage at [NIG-SC Website](#).

1: The approximate genome size, in base pairs.(Must be a value between 1 and 150000000)

GenomeSize =

2: The minimum length of reads (in base pairs) to use as seeds for pre-assembly.

Minimum Seed Length :

Automatic Estimation

If the coverage exceeds 30X, the Minimum Seed Read Length that results in at least 30X coverage by the longest subreads will be calculated automatically. If the coverage is less than 30X, the user-specified value will be used.

Use Manually Specified Value (regardless of the coverage)



# W8-2: パラメータの解説

The screenshot shows a GitHub Wiki page for 'HGAP in SMRT Analysis'. The page content is as follows:

**HGAP in SMRT Analysis**  
lhon edited this page on Jun 10 2014 · 2 revisions

This page contains information about the current release of HGAP. There have been multiple iterations of the HGAP implementation in SMRT Analysis, with performance improvements added to each iteration. In SMRT Analysis v2.0, we introduced, significantly speeding up HGAP execution. In most cases, HGAP.3 makes it the preferred protocol. In production environments, we recommend using the latest version of SMRT Analysis to ensure the best performance with HGAP.

SMRT Analysis v2.1 has a new implementation of HGAP that speeds up the process. This is found in the `RS_HGAP_Assembly.2` protocol. `RS_HGAP_Assembly.2` is used in SMRT Analysis v2.2.0 and later versions.

SMRT Analysis v2.2 contains a further improvement to HGAP, in which the final assembly stage is sped up, this new protocol is named `RS_HGAP_Assembly.3`. The `RS_HGAP_Assembly.2` and `RS_HGAP_Assembly.3` versions 2 and 3 is largely the same.

**Important parameters**

- 1. Genome Size**  
To accurately determine the Minimum Seed Read Length and the coverage of trimmed preassembled reads going into the assembly step, it is important to adjust the target genome size as accurately as possible.
- 2. Automatic Minimum Seed Read Length calculation**  
The Minimum Seed Read Length that results in at least 30X target genome coverage by the longest subreads is being calculated automatically (the default option). To use the user-selected Minimum Seed Read Length, the default option has to be **deselected**. If less than 30X coverage is being used for the HGAP process, the algorithm will use the user-selected Minimum Seed Length (6kb default), so lowering the default setting to 500bp is required to allow all-vs-all PreAssembly at lower than 30X coverage.

**Genome Size**  
At the moment, HGAP in SMRT Analysis supports genomes up to 130 MB; further improvements to scaling the workflow will enable support for larger genomes.

Older versions of SMRT Analysis may have lower genome size limits. SMRT Analysis 2.0 was limited to a 10 Mb genome size. We do not recommend using older versions of SMRT Analysis since they can have significant performance limitations; please upgrade if possible.

**Usage notes**  
For microbial assemblies we have seen improved assembly results using the latest workflows

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

# W8-3: HGAP実行

PDF原稿p104左下

http://p.ddbj.nig.ac.jp/pipeline/SettingAssembly.do

Setting for De Novo Ass...

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

Select Query Files → Select Tools → Set QuerySet → **Set Ass. Options** → Confirmation → Running Status

## Setting for De Novo Assembly

BACK NEXT

### hgap

#### Set optional parameters for HGAP pipeline

Select UGE-node to run :

- month\_fat (32 CPUs and 320GB memory)
- month\_medium (32 CPUs and 256GB memory)

Same results will be generated with either option.  
You can check the CPU and memory usage at [NIG-SC Website](#).

1 : The approximate genome size, in base pairs.(Must be a value between 1 and 150000000)

GenomeSize =  x

2 : The minimum length of reads (in base pairs) to use as seeds for pre-assembly.

Minimum Seed Length :

- Automatic Estimation  
If the coverage exceeds 30X, the Minimum Seed Read Length that results in at least 30X coverage by the longest subreads will be calculated automatically. If the coverage is less than 30X, the user-specified value will be used.
- Use Manually Specified Value (regardless of the coverage)

BACK NEXT

# W8-3: HGAP実行

http://p.ddbj.nig.ac.jp/pipeline/Confirm.do

Run Confirmation

Select Query Files → Select Tools → Set QuerySet → Set Ass. 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.

## Run Confirmation

BACK RUN

Destination of mail  
When the request is completed, the system sends an email to this address.  
kadota@bi.a.u-tokyo.ac.jp \* Required  
Result files will be deleted 60 days after submission.

Assembly [hgap]

Query sets  
Query set1

PairedOrientation	RunAccession	RunAlias	RowLength	QualityScore1	QualityScore2
single	21144	L.hokkaidonensis.PacBio3			
single	21143	L.hokkaidonensis.PacBio2			
single	21142	L.hokkaidonensis.PacBio1			

Assembly commands  
hgap

### Set optional parameters for HGAP pipeline

Select UGE-node to run :

month\_fat (32 CPUs and 320GB memory)  
 month\_medium (32 CPUs and 256GB memory)

Same results will be generated with either option.  
 You can check the CPU and memory usage at [NIG-SC Website](#).

1 : The approximate genome size, in base pairs.(Must be a value between 1 and 150000000)

GenomeSize = 2500000

Web ページからのメッセージ

Do you really want to execute pipeline programs?

OK キャンセル

# W9-2: 結果を眺める

① *de novo* Assembly、② Job 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

**Job info**

ID: 21965

Tool (Version): HGAP (Protocol3(v 2.2.0))

RunAccession or Filename	Download
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5	<a href="#">m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5</a>
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5	<a href="#">m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5</a>
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5	<a href="#">m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5</a>

**Assembly statistics**

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

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	<a href="#">View</a>		<a href="#">Download(13.1 MB)</a>	<a href="#">MD5</a>



# W9-2: 結果を眺める

① Total contig sizeは、乳酸菌の一般的なゲノムサイズと近く妥当。② Maximum 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

The screenshot shows the DDBJ pipeline detail view for job ID 21965. The job was run using HGAP (Protocol3(v 2.2.0)). The assembly statistics are as follows:

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

The command used for the run is: `run HGAP through smrtpipe.py : GenomeSize=2500000,minSeedLength=6000`. The run started on 2016-03-28 at 20:14:25 and ended on 2016-03-29 at 19:12:37.

Contig # : 4

Total contig size : 2,433,614

Maximum contig size : 2,289,497

Minimum contig size : 11,372

N50 contig size : 2,289,497

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

# W9-3: ダウンロード

The screenshot shows the DDBJ pipeline detail view for job ID 21965. The job info section indicates the tool used is HGAP (Protocol3(v 2.2.0)). Below this, a table lists the commands executed during the job. A red box highlights the first row of this table, and a red arrow points to the 'Download(13.1 MB)' link in the 'Result' column.

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	<a href="#">View</a>		<a href="#">Download(13.1 MB)</a>	<a href="#">MD5</a>



# 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るならば、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