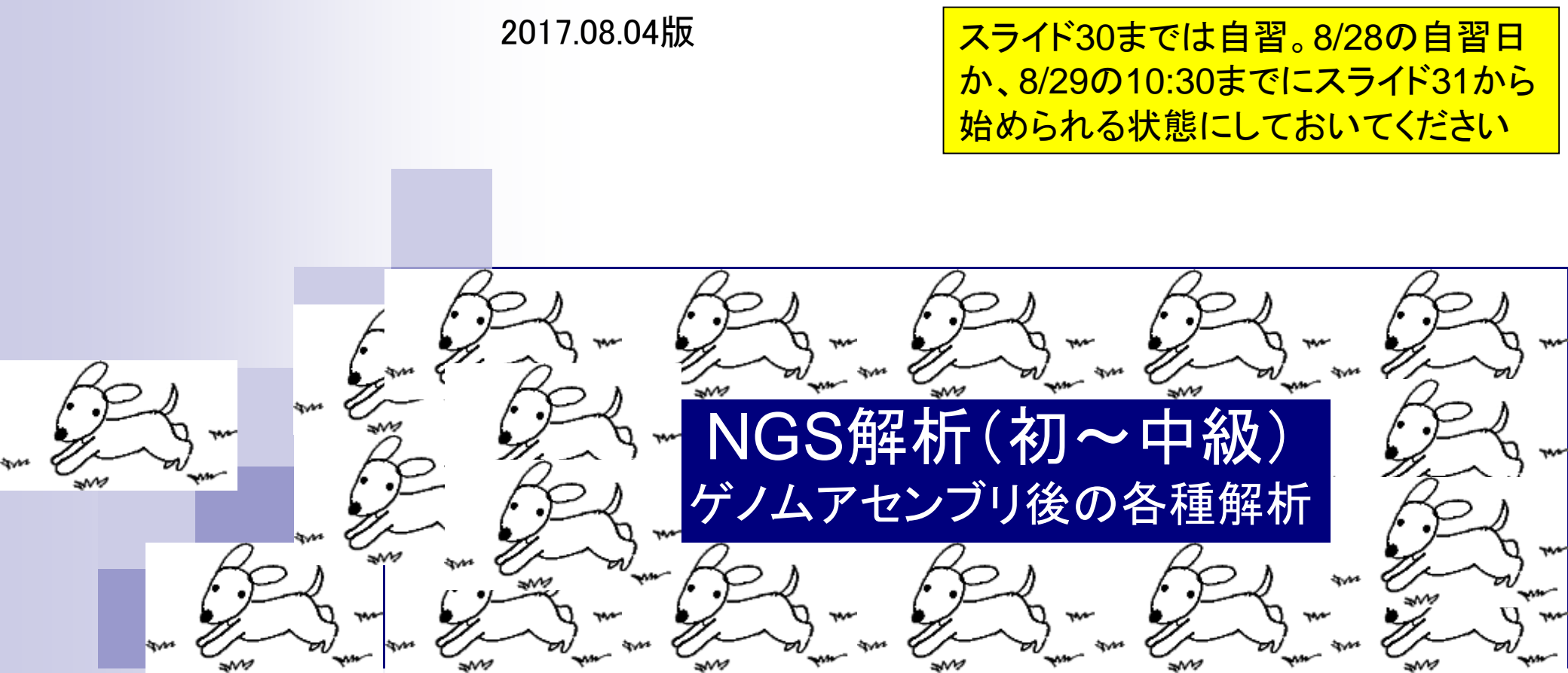


スライド30までは自習。8/28の自習日か、8/29の10:30までにスライド31から始められる状態にしておいてください



NGS解析(初～中級) ゲノムアセンブリ後の各種解析

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

門田幸二(かどた こうじ)

kadota@iu.a.u-tokyo.ac.jp

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

Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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



予習事項

①乳酸菌ゲノム配列決定論文のPacBioデータを用いて、*de novo*アセンブリ後の各種解析を行う。②主目的は、LinuxおよびNGS解析周辺のスキルアップ。昨年度からの継続性も重視

■ 平成28年度NGSハンズオン講習会

- 日本乳酸菌学会誌NGS連載第7回のウェブ資料W9-3までの内容を実施


■ 平成29年度NGSハンズオン講習会

- 8/29-30は、第7回の残り内容と第8回の内容が中心



予習事項

乳酸菌NGS連載第4回までは自習。第5-7回途中までは、**赤枠**内の平成28年度講習会の8/1-3で実施済み。特に、①2016年8月3日の講習会資料のスライド121以降を眺めておき、どんなデータのアセンブリ後の解析をやろうとしているかは把握しておきましょう

日	時間	部	内容	資料	
8月1日 (月)	10:30-18:15	第3部 NGS解析(中~上級) (農学生命情報科学特論II)	Linux環境でのデータ解析: JavaやRの利用法	<ul style="list-style-type: none"> 日本乳酸菌学会誌のNGS連載第4回の復習(特にFastQCとFaQCs) 乳酸菌連載第5回(W13-2まで) paired-endファイルのアダプター除去(FaQCs) Javaプログラムの設定と実行(Rockhopper2) Linux環境でのRの利用法(対話モードとバッチモード) 	(東京大学) (PDF:11.7MB) 統合TV
8月2日 (火)	10:30-18:15		Linux環境でのデータ解析: マッピング、トリミング、アセンブリ	<ul style="list-style-type: none"> NGS連載第5回(残り)、第6回(W10-6まで) RパッケージQuasRを用いたRNA-seqデータのマッピング 末端塩基のトリミング(Biostringsとfastx_trimmer) トリミング前後のde novoアセンブルとマッピング結果の評価 Illumina MiSeqデータの特徴と前処理(FastQCとFaQCs) de novoゲノムアセンブリ(Velvet) 	講義資料 (PDF:11.9MB) 統合TV
8月3日 (水)	10:30-18:15		クラウド環境との連携、ロングリードデータの解析	<ul style="list-style-type: none"> NGS連載第6回(残り)、ゲノムサイズ推定(KmerGenie) 配列長によるフィルタリング(Pythonプログラム実行と改変) DDBJ Pipeline (VelvetとPlatanus; エアーハンズオン) ロングリード(PacBio)データと公共DB ファイル形式(sra, FASTQ, bax.h5)、SRA Toolkit、FastQC DDBJ Pipeline (HGAP; エアーハンズオン) 	 講義資料 (PDF:10.3MB) 統合TV
8月4日 (木)	10:30-18:15		トランスクリプトームアセンブリ、発現量推定	<ul style="list-style-type: none"> de novoトランスクリプトームアセンブリ(TrinityとBridger) 前処理(アダプター除去やトリミング)との組合せによる性能比較 発現量推定(TIGAR2) 	講義資料 (PDF:9.75MB) 統合TV

予習事項

(実質的に同じなので)もちろん①乳酸菌NGS連載第7回の、②ウェブ資料のW9あたりまでを眺めておくのでも構いません

(Rで)塩基配列解析

(last modified 2017/06/04, since 2010)

このウェブページのR関連部分は、[インストール](#) | についての推奨手順 ([Windows2015.04.0](#) [Macintosh2015.04.03版](#))に従ってフリーソフトRと必要なパッケージをインストール済みであります。初心者の方は[基本的な利用法](#)([Windows2015.04.03版](#)と[Macintosh2015.04.03版](#))ウェブページを体系的にまとめた[書籍](#)もあります。(2015/04/03)

What's new?

- [解析 | 菌叢解析](#) | について
- [平成29年度NGSハンズオン](#) (月)14時00分~6月23日
- [平成28年度NGSハンズオン](#)
- [Galaxyのウェブサイト](#)のR
- [日本乳酸菌学会誌](#)のNGS

- 書籍 | [トランスクリプトーム解析](#) | [4.3.4 他の実験デザイン\(3群間\)](#) (last modified 2014/04/28)
- 書籍 | [日本乳酸菌学会誌](#) | について (last modified 2017/03/13)
- 書籍 | [日本乳酸菌学会誌](#) | [第1回イントロダクション](#) (last modified 2016/12/22)
- 書籍 | [日本乳酸菌学会誌](#) | [第2回GUI環境からコマンドライン環境へ](#) (last modified 2015/11/26)
- 書籍 | [日本乳酸菌学会誌](#) | [第3回Linux環境構築からNGSデータ取得まで](#) (last modified 2015/12/07)
- 書籍 | [日本乳酸菌学会誌](#) | [第4回クオリティコントロールとプログラムのインストール](#) (last modified 2016/06/03)
- 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 2016/07/05)
- 書籍 | [日本乳酸菌学会誌](#) | [第6回ゲノムアセンブリ](#) (last modified 2016/06/17)
- 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 2017/01/26)
- 書籍 | [日本乳酸菌学会誌](#) | [第8回アセンブリ後の解析](#) (last modified 2016/11/10)
- 書籍 | [日本乳酸菌学会誌](#) | [第9回ゲノムアノテーションとその可視化、DDBJへの登録](#) (last modified 2017/03/13)
- 書籍 | [日本乳酸菌学会誌](#) | [第10回DDBJへの塩基](#)
- 書籍 | [日本乳酸菌学会誌](#) | [第11回統合データ解析](#)
- イントロ | 一般 | [ランダムに行を抽出](#) (last modified 2016/11/10)

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](#)(2016-06.20版; 約15MB)
 - [Macintosh用](#)

Linuxコマンド

- [apt-cache](#) (パッケージ調査)
- [apt-get](#) (パッケージのインストールやアップデート)

W9-2: 結果を眺める

第7回W9-2で、DDBJ Pipeline上で実行したHGAPアセンブリ結果のページを眺めている。① Total contig sizeは、乳酸菌の一般的なゲノムサイズ(2-3Mb)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。③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)

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	m130821_065825_42195_c100539522550000001823089611241356_s1_p0.3.bax.h5
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5	m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5	m130821_065825_42195_c100539522550000001823089611241356_s1_p0.1.bax.h5

Download modified queries

The modified query file does not exist, because of the following reasons:

- The file is expired. (about 1 months)
- Job is waiting for execution queue.
- Error in query file.

Download wgs file

- out_WGS.fasta.gz (Original size 2.4 MB)

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

Time

Wait time	Start time	End time
0: 0:11	2016-03-28 20:14:25	2016-03-29 19:13:20

Command

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

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

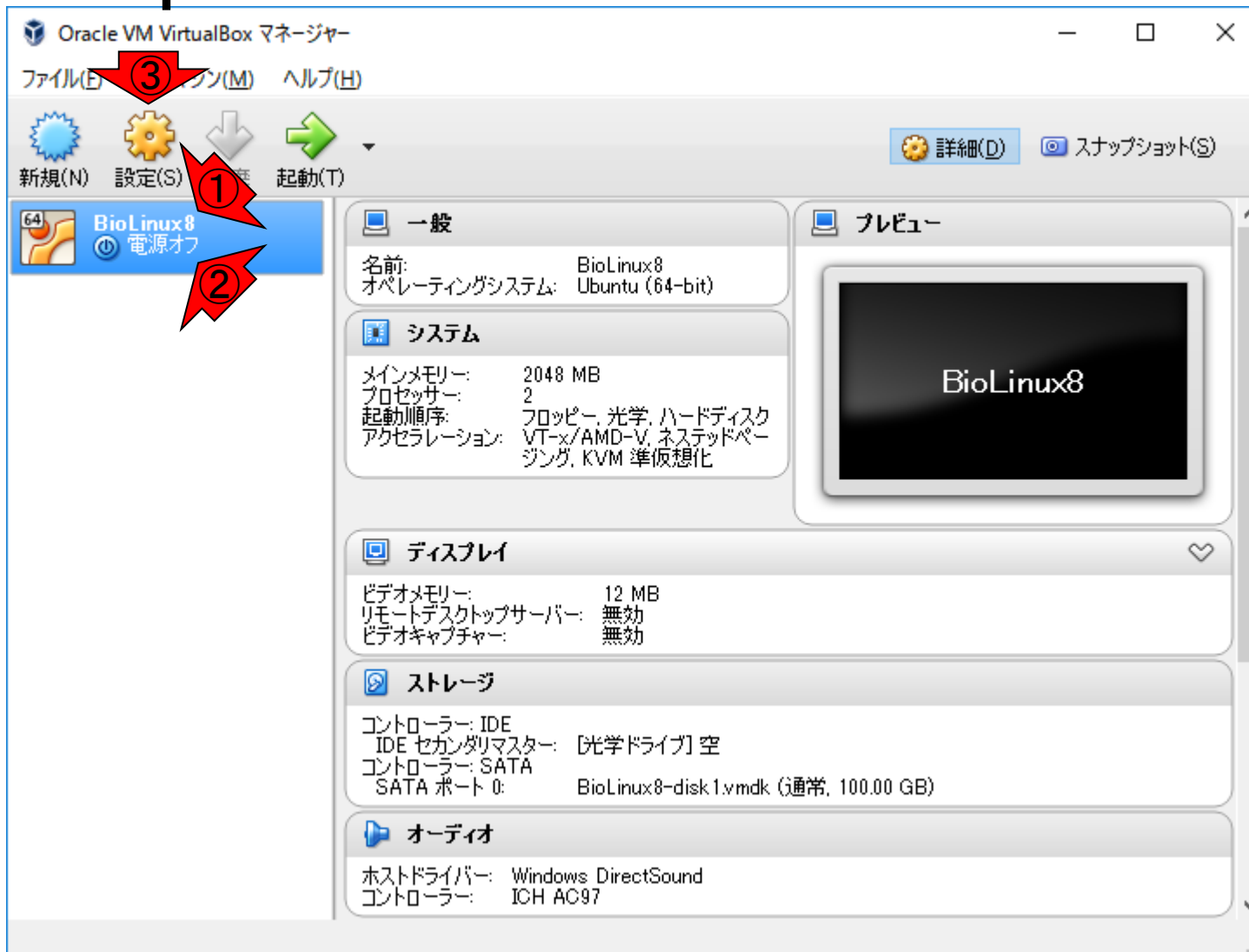
Contig # : 4
 Total contig size : 2,433,614
 Maximum contig size : 2,289,497
 Minimum contig size : 11,372
 N50 contig size : 2,289,497

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

①のBioLinux8という名前をNGS20170829に変更する。②電源オフになっている状態で、③設定

Tips: 名前の変更



Tips: 名前の変更

The screenshot shows the Oracle VM VirtualBox Manager interface. The main window is titled "Oracle VM VirtualBox マネージャー" and contains a menu bar (ファイル(E), 仮想マシン(M), ヘルプ(H)) and a toolbar with icons for 新規(N), 設定(S), 破棄, and 起動(T). Below the toolbar are buttons for 詳細(D) and スナップショット(S).

The "BioLinux8 - 設定" window is open, showing the "一般" (General) tab. The left sidebar lists various settings categories: 一般, システム, ディスプレイ, ストレージ, オーディオ, ネットワーク, シリアルポート, USB, 共有フォルダー, and ユーザーインターフェース.

In the "一般" tab, there are sub-tabs: 基本(B), 高度(A), 説明(D), and 暗号化(R). The "基本(B)" sub-tab is active, showing the following fields:

- 名前(N): BioLinux8 (This field is highlighted with a red circle and the number 1, indicating it is the focus of the tip.)
- タイプ(T): Linux
- バージョン(V): Ubuntu (64-bit)

At the bottom right of the "BioLinux8 - 設定" window are "OK" and "Cancel" buttons.

Tips: 名前の変更

The screenshot shows the Oracle VM VirtualBox Manager interface. The main window is titled "Oracle VM VirtualBox マネージャー" and contains a menu bar (ファイル(E), 仮想マシン(M), ヘルプ(H)) and a toolbar with icons for 新規(N), 設定(S), 破棄, and 起動(T). Below the toolbar are buttons for 詳細(D) and スナップショット(S).

The "BioLinux8 - 設定" window is open, showing the "一般" (General) tab. The "名前(N)" field contains "NGS20170829" and is highlighted with a red arrow and the number 1. The "タイプ(T)" dropdown is set to "Linux" and the "バージョン(V)" dropdown is set to "Ubuntu (64-bit)".

At the bottom right of the settings window, the "OK" button is highlighted with a red arrow and the number 2, and the "Cancel" button is also visible.

①無事NGS20170829になりました。この部分はただの識別用。8/31、および9/1のovaファイルの名前と被るときや自分好みの名前に変えたいときにご利用ください

Tips: 名前の変更

Oracle VM VirtualBox マネージャー

ファイル(E) 仮想マシン(M) ヘルプ(H)

新規(N) 設定(S) 破壊 起動(T)

詳細(D) スナップショット(S)

NGS20170829 電源オフ

①

一般

名前: NGS20170829
オペレーティングシステム: Ubuntu (64-bit)

システム

メインメモリ: 2048 MB
プロセッサ: 2
起動順序: フロッピー, 光学, ハードディスク
アクセラレーション: VT-x/AMD-V, ネステッドページング, KVM 準仮想化

ディスプレイ

ビデオメモリ: 12 MB
リモートデスクトップサーバー: 無効
ビデオキャプチャー: 無効

ストレージ

コントローラー: IDE
IDE セカンダリマスター: [光学ドライブ] 空
コントローラー: SATA
SATA ポート 0: BioLinux8-disk1.vmdk (通常, 100.00 GB)

オーディオ

ホストドライバー: Windows DirectSound
コントローラー: ICH AC97

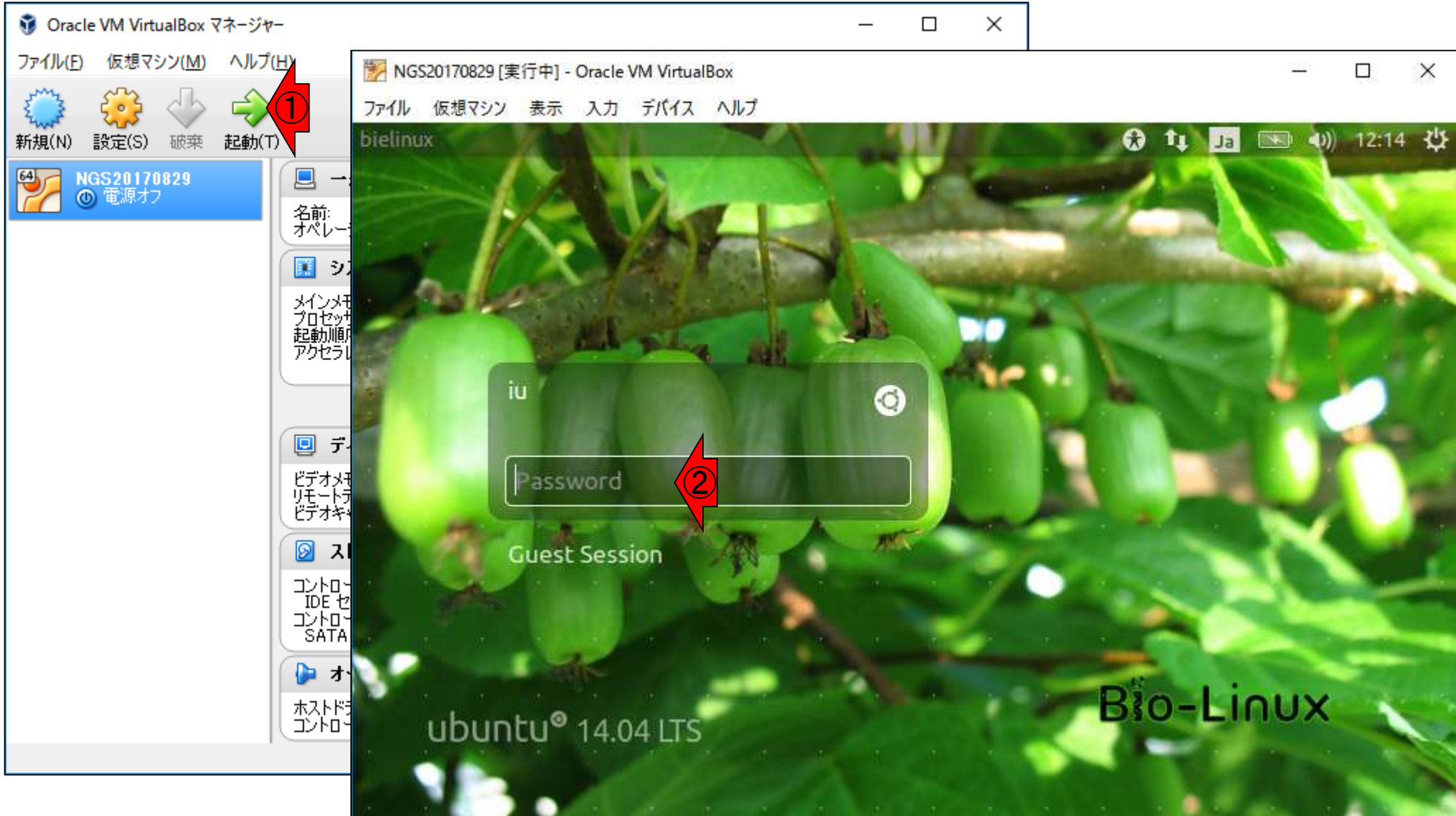
プレビュー

NGS20170829

①

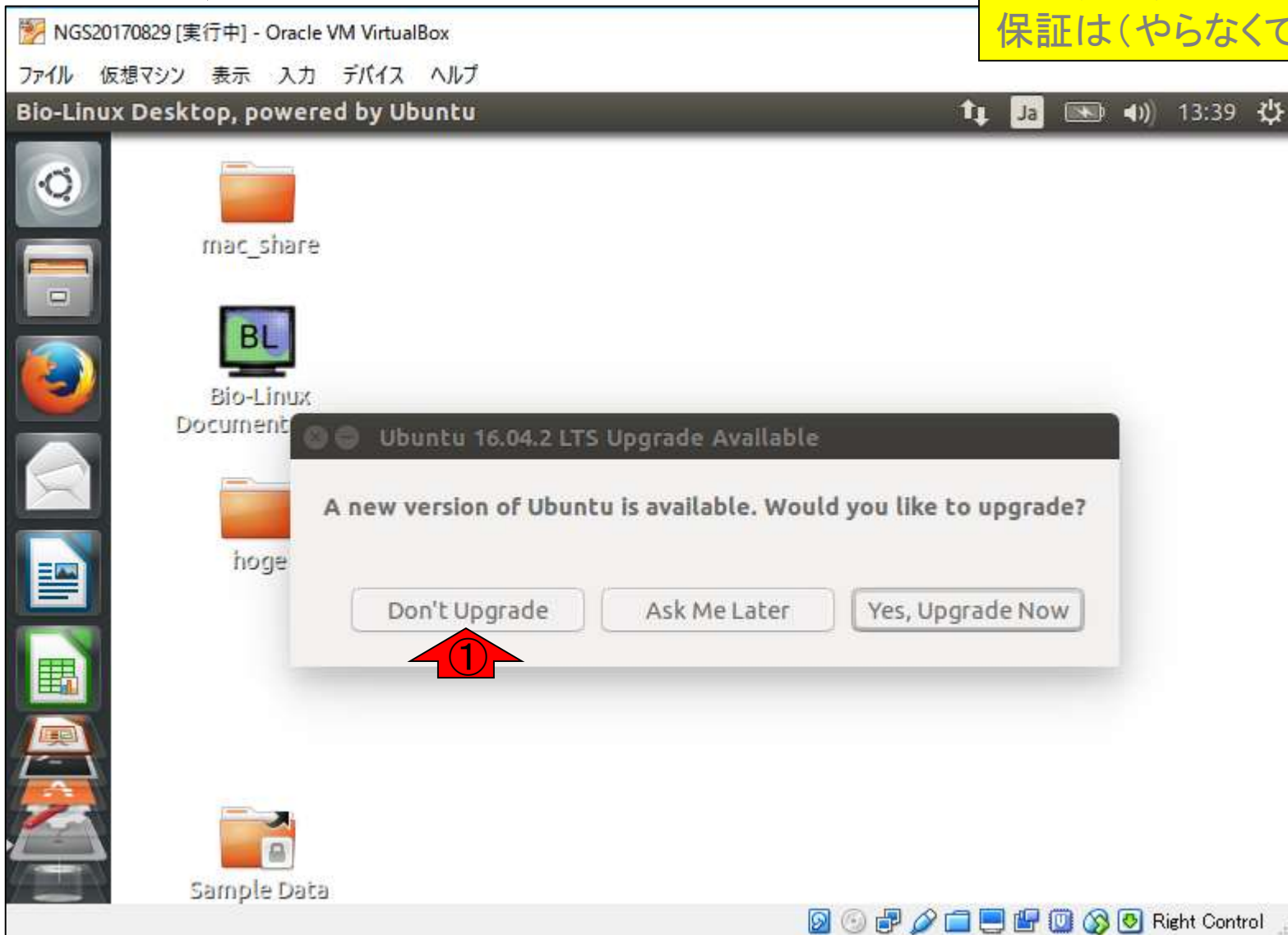
起動

①でBio-Linuxを起動。②パスワードは pass1409 です (連載第3回W2あたり)

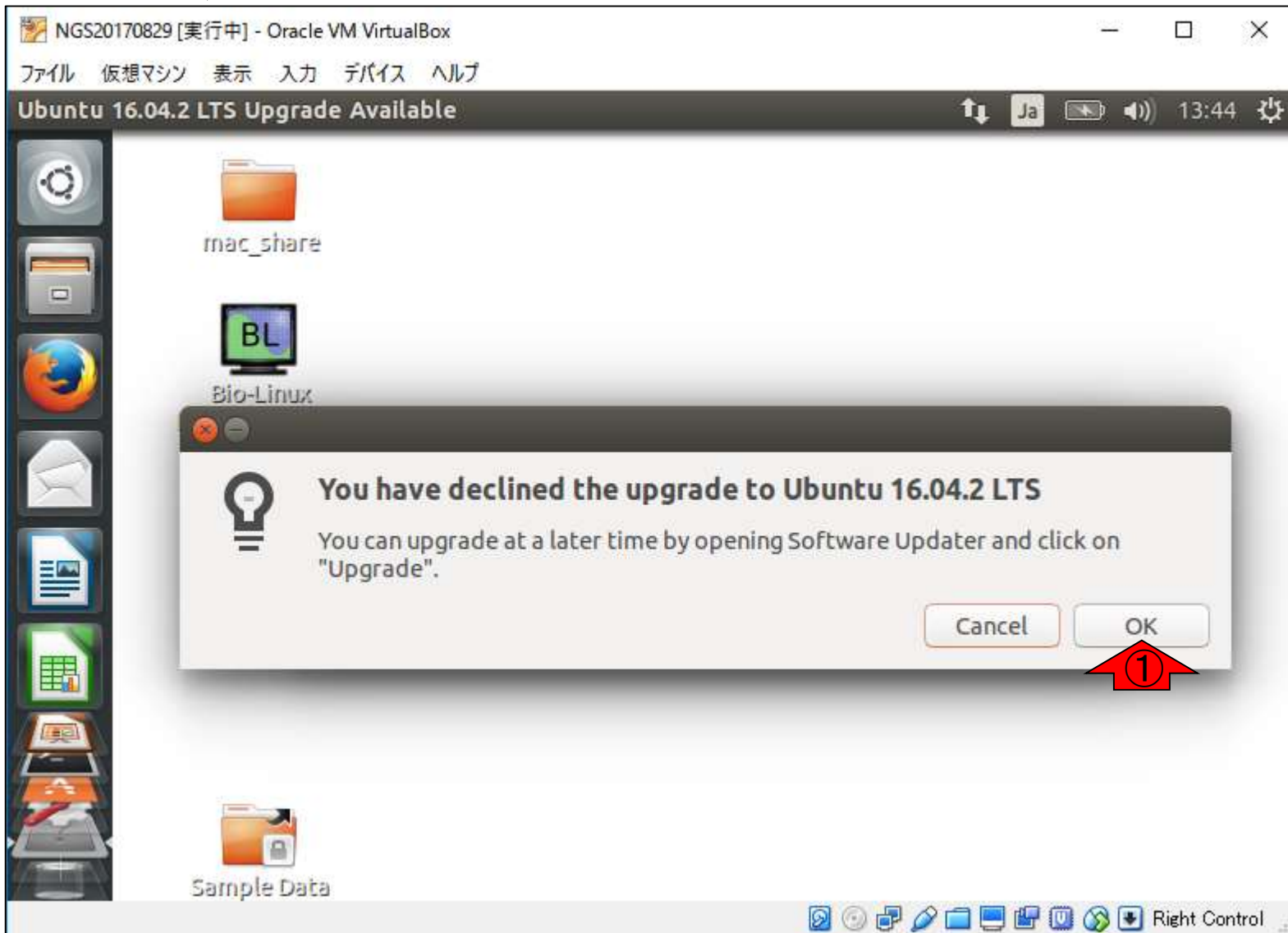


起動後

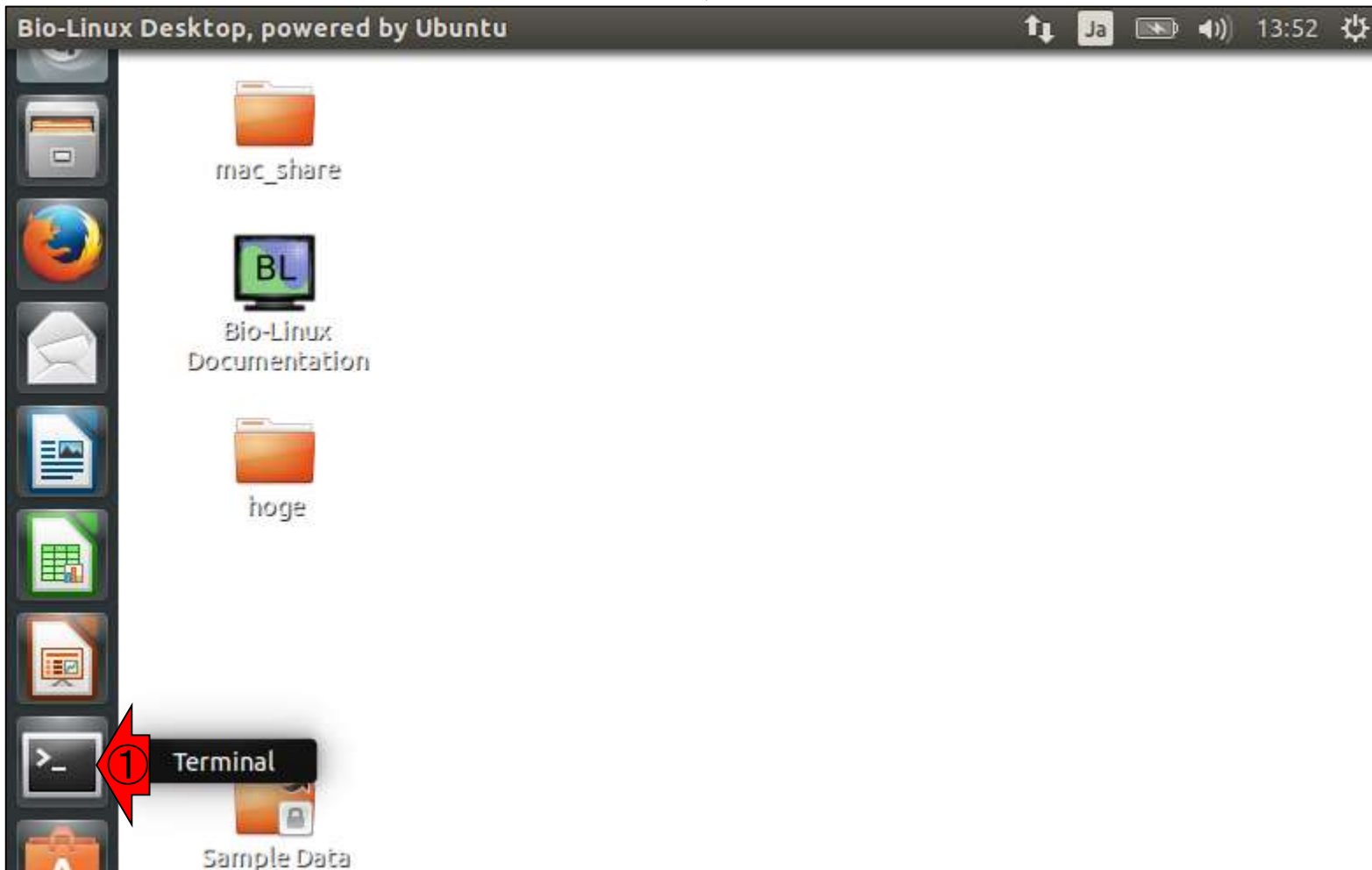
(もしこのようなポップアップが出たとしたら)①Don't Upgrade。やってもいいとは思いますが、やったことがないのでその後の保証は(やらなくてもですがw)できません



起動後



ターミナルを起動



ターミナル起動後



backupがないのはダメ

①lsで確認。②backupという名前のディレクトリがないヒトは平成29年度講習会用として指定されたovaファイルを正しくインポートできていないので、指示通りにやり直してください

```
iu@bielinux[~]
iu@bielinux[iu] ls
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu]
```

[2:08午後]

[2:09午後]

W9-3: ダウンロード1

PacBio用 *de novo* アセンブリプログラムHGAP実行結果である、①result.zipというzip圧縮ファイルを共有フォルダ(ホストOS側はDesktop/share)にダウンロード。②MD5については、連載第3回W12で説明した。③リンク先のMD5チェックサム値

The screenshot shows the DDBJ pipeline web interface. The main content area displays job information for ID 21965, using the HGAP tool. Below this is a table with columns: Command, Start time, End time, Log1, Log2, Result, and MD5. The first row contains the command 'run HGAP through smrtpipe.py : GenomeSize=2500000,minSeedLength=6000', start time '2016-03-28 20:14:25', end time '2016-03-29 19:12:37', a 'View' link in Log1, a 'Download(13.1 MB)' link in Result, and an 'MD5' link in MD5. Below the table, the 'Assembly statistics' section shows 'Contig # : 4', 'Total contig size : 2,433,614', and 'Maximum contig size : 2,289,497'. The MD5 value '5cf4ed21fd476edce6625afaec577815' is displayed next to 'result.zip'. A 'Time' table shows wait time '0: 0:11', start time '2016-03-28 20:14:25', and end time '2016-03-29 19:13:20'. At the bottom, a smaller version of the command table is visible. Red boxes and arrows are used to highlight the command table, the 'Download(13.1 MB)' link (labeled 1), the 'MD5' link (labeled 2), and the MD5 value '5cf4ed21fd476edce6625afaec577815' (labeled 3).

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

Assembly statistics

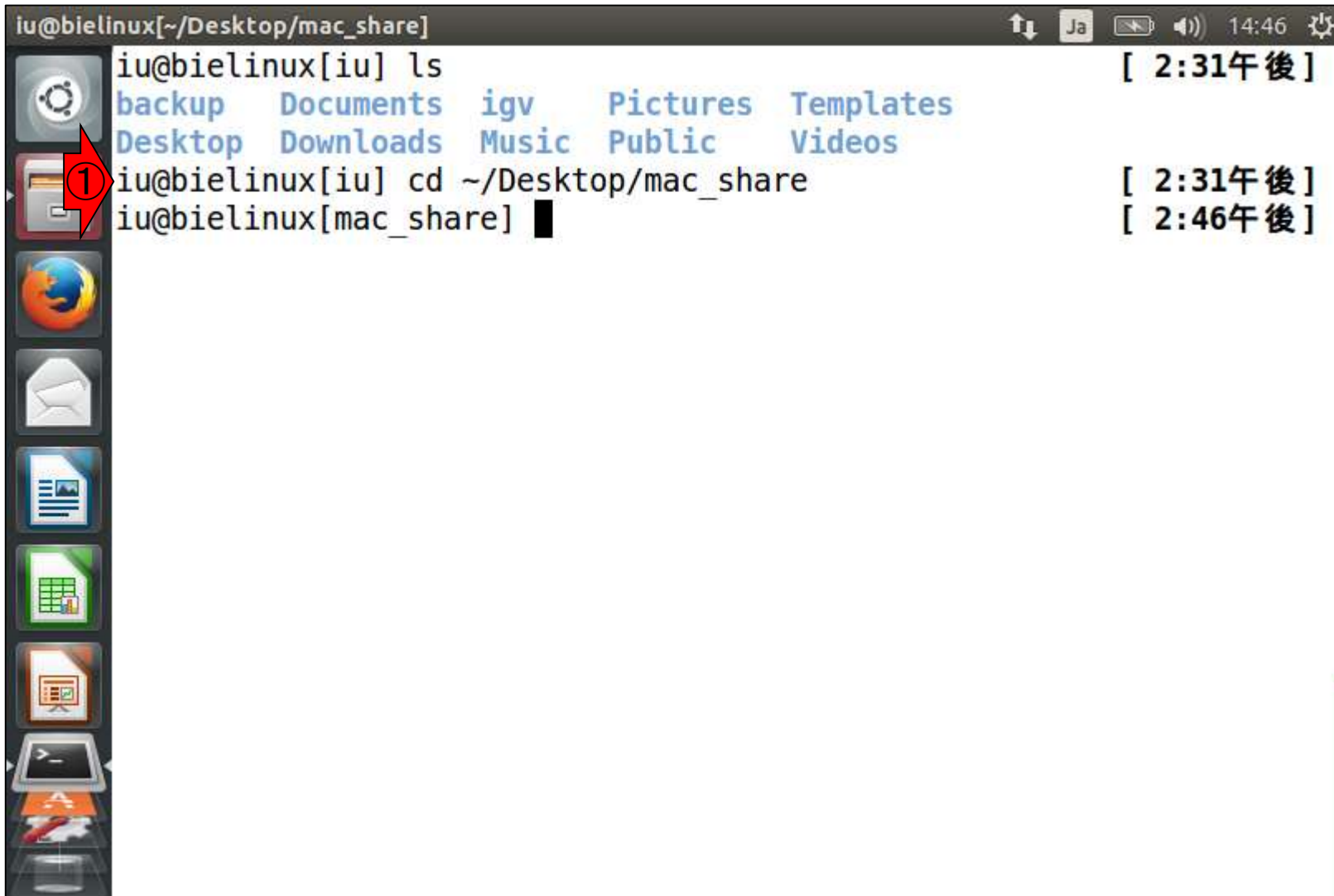
Contig # : 4
Total contig size : 2,433,614
Maximum contig size : 2,289,497

MD5: 5cf4ed21fd476edce6625afaec577815 result.zip

Wait time	Start time	End time
0: 0:11	2016-03-28 20:14:25	2016-03-29 19:13:20

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

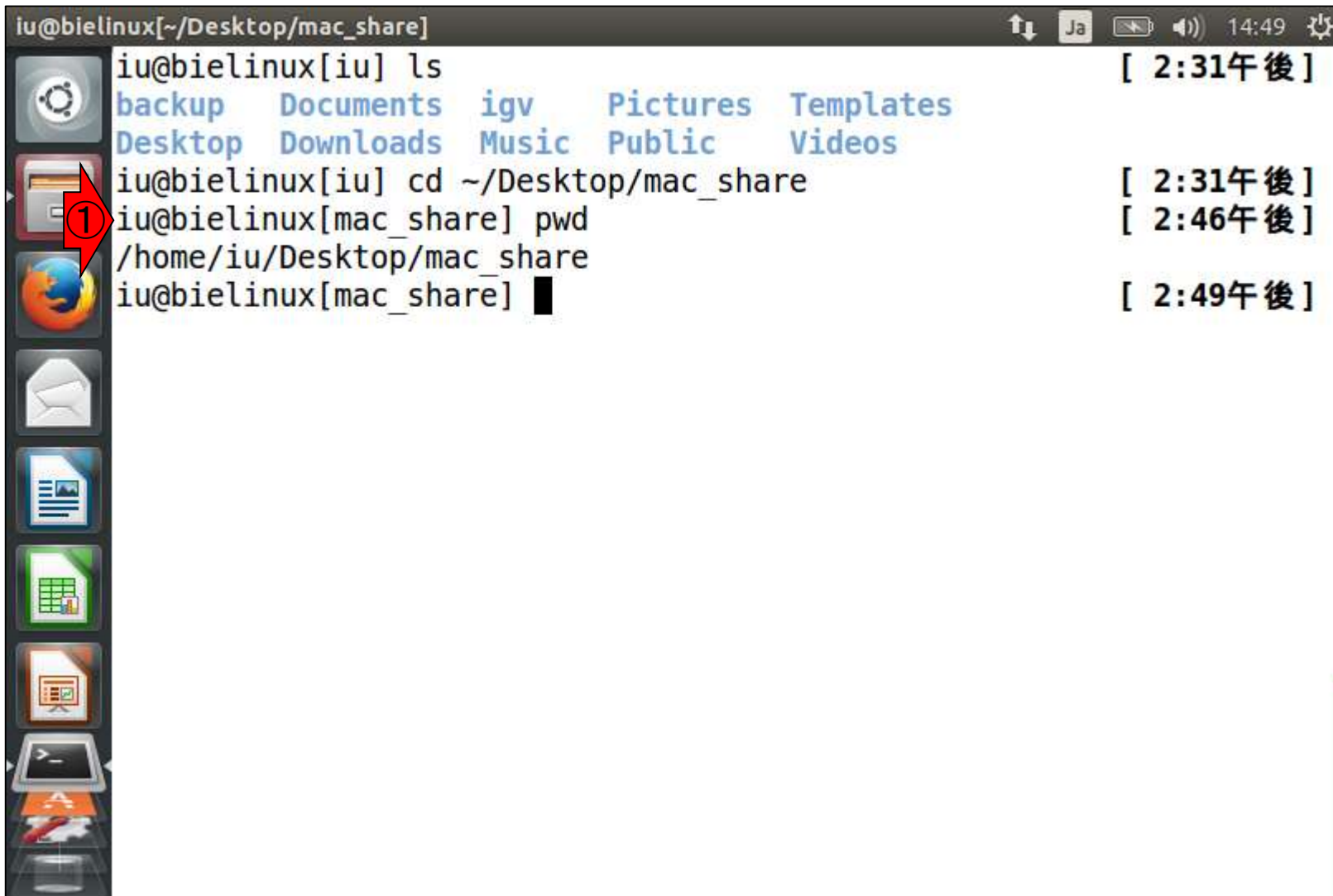
W9-3:ダウンロード2



The image shows a terminal window on a Linux system. The prompt is `iu@bielinux[~/Desktop/mac_share]`. The user has entered the `ls` command, which lists the contents of the current directory: `backup Documents igv Pictures Templates Desktop Downloads Music Public Videos`. The time shown is [2:31午後]. The user then enters the `cd ~/Desktop/mac_share` command, and the prompt changes to `iu@bielinux[mac_share]`. The time shown is [2:46午後]. A red arrow with the number 1 points to the `cd` command line. The terminal window has a sidebar on the left with various application icons, including a file manager icon that is highlighted with a red arrow and the number 1.

```
iu@bielinux[~/Desktop/mac_share] [ 2:31午後]
iu@bielinux[iu] ls
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 2:31午後]
iu@bielinux[mac_share] [ 2:46午後]
```


W9-3: ダウンロード3



```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] pwd
```

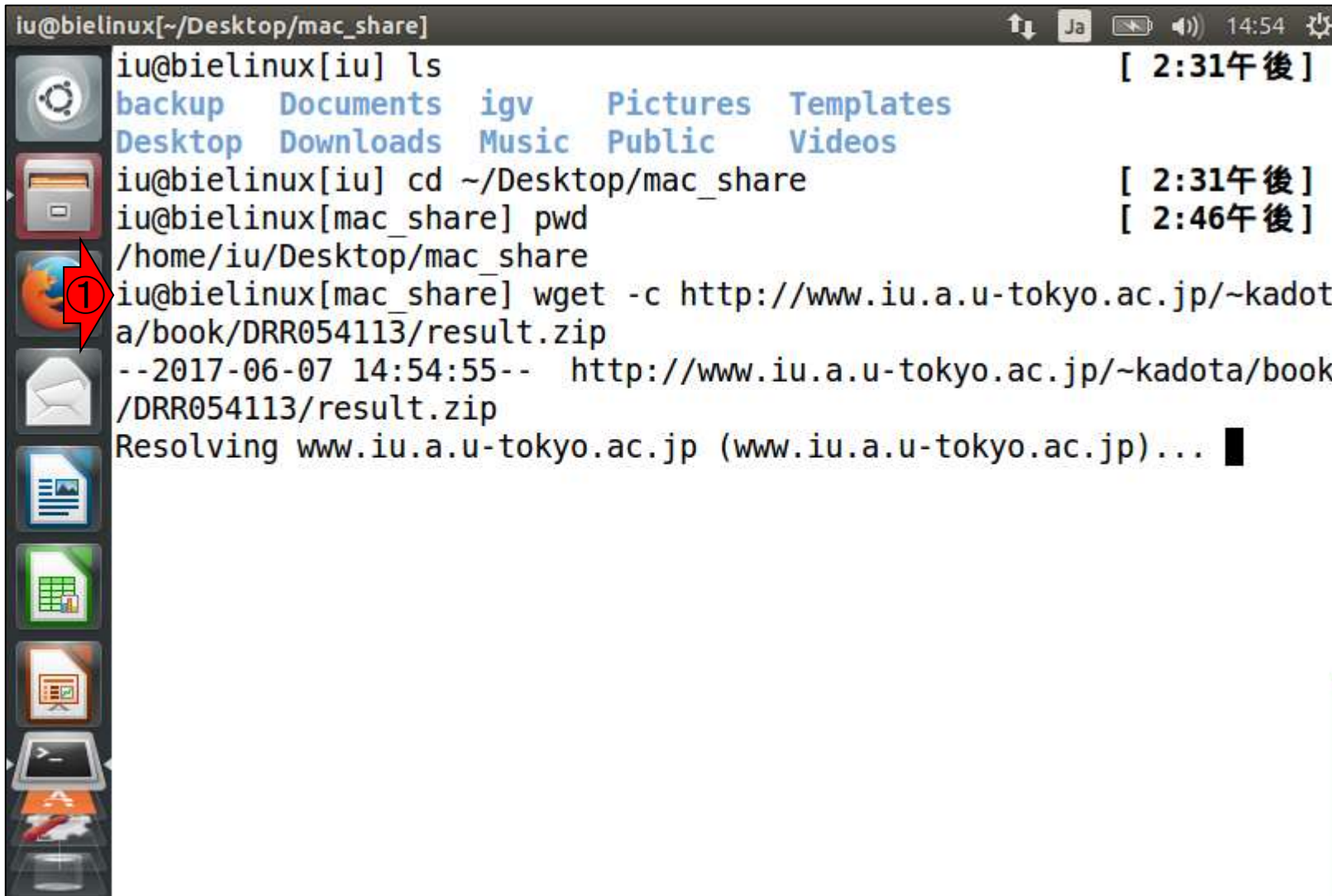
W9-3: ダウンロード4

①wgetコマンド(初出はたしか第3回W11-1)でアセンブリ結果ファイル(result.zip)をダウンロード。ここでは、②オプションを-cのみとして、途中経過を表示させている

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 2:31午後 ]
iu@bielinux[mac_share] pwd [ 2:46午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadot
a/book/DRR054113/result.zip
```



W9-3:ダウンロード5



```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls [ 2:31午後]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 2:31午後]
iu@bielinux[mac_share] pwd [ 2:46午後]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota
a/book/DRR054113/result.zip
--2017-06-07 14:54:55-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)...
```

①ときどきネットワークの不調で、こんな感じで失敗することがあります。そんなときは…

W9-3: ダウンロード6

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls [ 2:31午後]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 2:31午後]
iu@bielinux[mac_share] pwd [ 2:46午後]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota
a/book/DRR054113/result.zip
--2017-06-07 14:54:55-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... failed
: Name or service not known.
wget: unable to resolve host address 'www.iu.a.u-tokyo.ac.jp'
iu@bielinux[mac_share] [ 2:55午後]
```



W9-3: ダウンロード7

数分後にwgetをリトライするのもアリですが、
①のような感じで、②~/backupディレクトリ内
にあるresult.zipを、③カレントディレクトリにコ
ピーして、wgetしたつもりになってください

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/book/DRR054113/result.zip
--2017-06-07 17:08:36-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... failed
: Name or service not known.
wget: unable to resolve host address 'www.iu.a.u-tokyo.ac.jp'
iu@bielinux[mac_share] cp ~/backup/result.zip .
```

① (red arrow pointing to the terminal icon in the sidebar)
② (red arrow pointing to the file path ~/backup/result.zip)
③ (red arrow pointing to the dot in the cp command)

①ls -lで確認。②約13MBのresult.zipが確かに存在します

W9-3:ダウンロード8

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls [ 5:08午後]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 5:08午後]
iu@bielinux[mac_share] pwd [ 5:08午後]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/book/DRR054113/result.zip
--2017-06-07 17:08:36-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... failed
: Name or service not known.
wget: unable to resolve host address 'www.iu.a.u-tokyo.ac.jp'
iu@bielinux[mac_share] cp ~/backup/result.zip . [ 5:08午後]
iu@bielinux[mac_share] ls -l [ 5:16午後]
total 12822
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] [ 5:17午後]
```



①md5sumでresult.zipの②MD5チェックサム値(第3回W12)を表示。この値と…

W9-3:ダウンロード9

```
iu@bielinux[~/Desktop/mac_share] [ 5:08午後]
iu@bielinux[iu] ls [ 5:08午後]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 5:08午後]
iu@bielinux[mac_share] pwd [ 5:08午後]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota
a/book/DRR054113/result.zip
--2017-06-07 17:08:36-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... failed
: Name or service not known.
wget: unable to resolve host address 'www.iu.a.u-tokyo.ac.jp'
iu@bielinux[mac_share] cp ~/backup/result.zip . [ 5:08午後]
iu@bielinux[mac_share] ls -l [ 5:16午後]
total 12822
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] md5sum result.zip [ 5:17午後]
5cf4ed21fd476edce6625afaec577815 result.zip
iu@bielinux[mac_share] [ 5:26午後]
```



W9-3: ダウンロード9

(今回はダウンロード元がDDBJ Pipelineではなかったが、そうだったとして) ①をクリックして見られる②MD5チェックサム値と比較し、同じであればファイルの中身は同じである(ファイルが壊れていない)と判断する

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

All status

HELP

HELP 07

TUTORIAL

Contact Us

DDBJ Read Annotation Pipeline Development Team

Detail view

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

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	m130821_065825_42195_c100539522550000001823089611241356
m130821_065825_42195_c100539522550000001823089611241356_s1_p0.2.bax.h5	m130821_065825_42195_c100539522550000001823089611241356

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

Download wgs file

- out_WGS.fasta.gz (Original size 2.4 MB)

Assembly statistics

Contig # : 4

Total contig size : 2,433,614

Maximum contig size : 2,289,497

Minimum contig size : 44,272

Time

Wait time	Start time	End time
0: 0:11	2016-03-28 20:14:25	2016-03-29 19:13:20

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

5cf4ed21fd476edce6625afaec577815 result.zip

ゲストOS (Bio-Linux) 上の①のディレクトリは、ホストOSのデスクトップ上にあるshareフォルダとつながっています。そのような設定を行ったからです

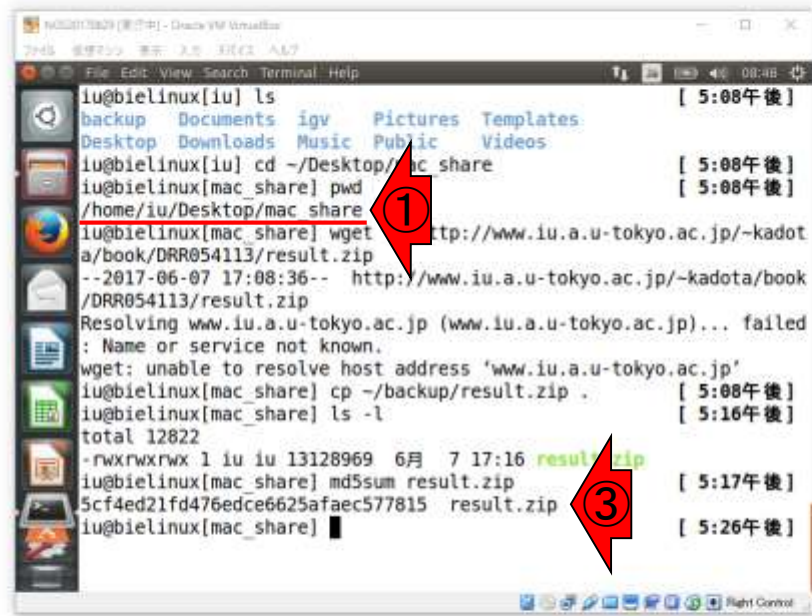
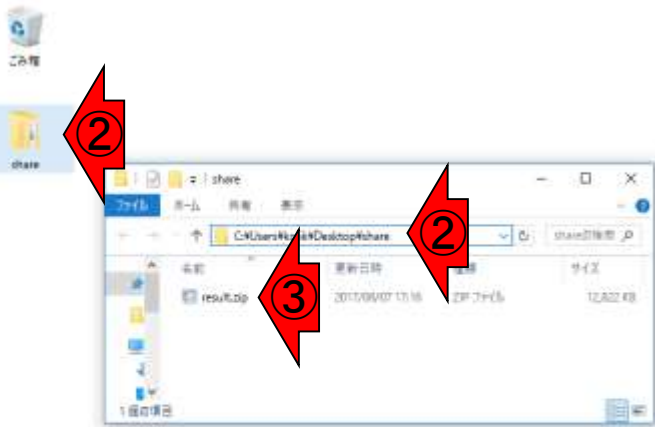
共有フォルダ

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[iu] ls [ 5:08午後 ]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] cd ~/Desktop/mac_share [ 5:08午後 ]
iu@bielinux[mac_share] pwd [ 5:08午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota
a/book/DRR054113/result.zip
--2017-06-07 17:08:36-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/DRR054113/result.zip
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... failed
: Name or service not known.
wget: unable to resolve host address 'www.iu.a.u-tokyo.ac.jp'
iu@bielinux[mac_share] cp ~/backup/result.zip . [ 5:08午後 ]
iu@bielinux[mac_share] ls -l [ 5:16午後 ]
total 12822
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] md5sum result.zip [ 5:17午後 ]
5cf4ed21fd476edce6625afaec577815 result.zip
iu@bielinux[mac_share] [ 5:26午後 ]
```



共有フォルダ

ゲストOS (Bio-Linux) 上の①のディレクトリは、②ホストOS (この場合Windows 10) のデスクトップ上にあるshareフォルダとつながっています。つまり共有フォルダです。③確かに同じresult.zipファイルが見えていますね。この(程度の話)についてこれるヒトが予習をやったと言えるヒト



Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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



W9-4: 解凍して概観

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 9:57午前 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] unzip result.zip [ 9:57午前 ]
Archive: result.zip
  creating: result/
  inflating: result/corrected.fastq
  inflating: result/smrtpipe.log
  inflating: result/polished_assembly.fastq
  inflating: result/polished_assembly.fasta
iu@bielinux[mac_share] [ 9:57午前 ]
```



①lsで中身を確認。②resultディレクトリの中には、③赤枠で示す計4つのファイルが存在する

W9-4: 解凍して概観

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 9:57午前 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] unzip result.zip [ 9:57午前 ]
Archive: result.zip
  creating: result/
  inflating: result/corrected.fastq
  inflating: result/smrtpipe.log
  inflating: result/polished_assembly.fastq
  inflating: result/polished_assembly.fasta
① iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
②
result:
total 75356
-rwxrwxrwx 1 iu iu 69756928 3月 29 2016 corrected.fastq
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 64556 3月 29 2016 smrtpipe.log
③
iu@bielinux[mac_share] [10:00午前]
```

W9-4: 解凍して概観

①がPacBio用の *de novo* アセンブリプログラムHGAPの主な実行結果。拡張子が .fastaなので、multi-FASTA形式ファイル

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] pwd [ 9:57午前 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] unzip result.zip [ 9:57午前 ]
Archive: result.zip
  creating: result/
  inflating: result/corrected.fastq
  inflating: result/smrtpipe.log
  inflating: result/polished_assembly.fastq
  inflating: result/polished_assembly.fasta
iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip

result:
total 75356
-rwxrwxrwx 1 iu iu 69756928 3月 29 2016 corrected.fastq
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 64556 3月 29 2016 smrtpipe.log
iu@bielinux[mac_share] [10:00午前]
```



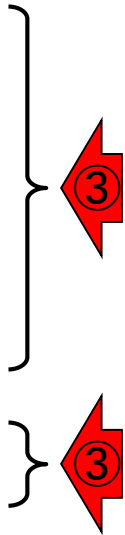
①

Tips: ls -ld

①ls -ldとすれば、ディレクトリの中身を非表示にできる(第6回W11-2)。②ls -lとの違いは③一目瞭然

```
iu@bielinux[~/Desktop/mac_share]
iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] unzip result.zip [ 9:57午前 ]
Archive: result.zip
  creating: result/
  inflating: result/corrected.fastq
  inflating: result/smrtpipe.log
  inflating: result/polished_assembly.fastq
  inflating: result/polished_assembly.fasta
② iu@bielinux[mac_share] ls -l result* [ 9:57午前 ]
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip

result:
total 75356
-rwxrwxrwx 1 iu iu 69756928 3月 29 2016 corrected.fastq
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 64556 3月 29 2016 smrtpipe.log
① iu@bielinux[mac_share] ls -ld result* [10:00午前 ]
drwxrwxrwx 1 iu iu 4096 3月 29 2016 result
-rwxrwxrwx 1 iu iu 13128969 6月 7 17:16 result.zip
iu@bielinux[mac_share] [10:01午前 ]
```



第7回原稿p104右下

今は①赤枠あたりです。このあと②赤下線部分あたりの話をしていきます

①
ノムサイズ (2.5MB; 2,500,000 bp) を与えたが、Illumina MiSeq データから得られた推定ゲノムサイズ (約 2.4MB; 2,400,000 bp) を与えてみてもいいだろう³⁾ [W8-3]。PacBio データは、ランダムな位置でシーケンシングエラーが起こる。そのため、②で短いリードをマップして、ポジションごとに出現する塩基の多数決ルールを適用することでエラー補正ができる。X に相当する推定ゲノムサイズを与えることで、coverage の計算を行うことができる。

②
2つめのパラメータである Minimum Seed Length は、基本的にデフォルトの 6,000 bp、および Automatic Estimation でよい。PacBio の真骨頂は、リピート配列を超えうる長さのリードを得られる点にある。それゆえ、①で選択するシードは、安定的にアセンブルできる 25X を超える範囲で、できるだけ長いサブリードに限定するほうがよい。それを自動的に算出するのが Automatic Estimation である。6,000 bp という値は、25X という条件を満たす最短サブリード長が 6,000 bp 未満だった場合に

HGAP 実行結果の概観と前処理

DDBJ Pipeline で HGAP (Protocol3; ver. 2.2.0) を実行した結果 (result.zip) には、計 4 つのファイルが含まれる [W9-4]。エンドユーザが欲しい最終結果ファイルは、polished_assembly.fasta である。Linux の基本コマンド [W9-5] や R [W9-6] を駆使して、このファイルの全体像を大まかに把握し、ある程度予想を立てる。具体的には、最も長い 2,289,497 bp のコンティグは、乳酸菌の平均ゲノムサイズ (約 2.5MB) に近いことから、染色体 (chromosome) だろうと予想した。それ以外の 3 つのコンティグ (86,892 bp, 45,853 bp, and 11,372 bp) は、染色体の一部、プラスミド、ミスアセンブルのいずれかであろう。もしコンティグが環状であれば、プラスミドである可能性が高い。そこでまずは、コンティグごとに環状かどうかのチェックを行う。

W9-5: コンティグ数

①resultディレクトリに移動し、②コンティグ数を表示。FASTA形式ファイルなので">"を含む行数がコンティグ数に相当する。③description行を表示。こんな感じの記述内容か~と思うだけ。④行数は、40,567行

```
iu@bielinux[mac_share] pwd [12:25午後]
/home/iu/Desktop/mac_share
① iu@bielinux[mac_share] cd result [12:26午後]
iu@bielinux[result] pwd [12:26午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l [12:26午後]
total 75356
-rwxrwxrwx 1 iu iu 69756928 3月 29 19:12 corrected.fastq
-rwxrwxrwx 1 iu iu 2474245 3月 29 19:12 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 19:12 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 64556 3月 29 19:12 smrtpipe.log
② iu@bielinux[result] grep -c ">" polished_assembly.fasta
4
③ iu@bielinux[result] grep ">" polished_assembly.fasta [12:26午後]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
④ iu@bielinux[result] wc polished_assembly.fasta [12:26午後]
 40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] █ [12:35午後]
```

W10-2のイントロ1

①cdでホームディレクトリに移動。②lsで確認。これから作成予定のbinというディレクトリがないことを確認しているだけ

```
iu@bielinux[~]
iu@bielinux[mac_share] pwd [ 2:16午後 ]
/home/iu/Desktop/mac_share
① iu@bielinux[mac_share] cd [ 2:16午後 ]
iu@bielinux[iu] pwd [ 2:16午後 ]
/home/iu
② iu@bielinux[iu] ls [ 2:16午後 ]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
iu@bielinux[iu] [ 2:16午後 ]
```

W10-2のイントロ2

```
iu@bielinux[~/bin]
iu@bielinux[mac_share] pwd [ 2:16午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] cd [ 2:16午後 ]
iu@bielinux[iu] pwd [ 2:16午後 ]
/home/iu
iu@bielinux[iu] ls [ 2:16午後 ]
backup Documents igv Pictures Templates
Desktop Downloads Music Public Videos
① iu@bielinux[iu] mkdir bin [ 2:16午後 ]
iu@bielinux[iu] ls [ 2:17午後 ]
backup Desktop Downloads Music Public Videos
bin Documents igv Pictures Templates
② iu@bielinux[iu] cd bin [ 2:17午後 ]
iu@bielinux[bin] pwd [ 2:17午後 ]
/home/iu/bin
iu@bielinux[bin] █ [ 2:17午後 ]
```

W10-2のイントロ3

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)を①wgetでダウンロード。このプログラムは配列長順にソートした結果を返す

```
iu@bielinux[~/bin]
iu@bielinux[bin] pwd
/home/iu/bin
iu@bielinux[bin] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/fastaLengthFilter.py
```

[2:20午後]



W10-2のイントロ3

```
iu@bielinux[~/bin]
iu@bielinux[bin] pwd
/home/iu/bin
iu@bielinux[bin] wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/fastaLengthFilter.py
--2017-06-12 14:28:26-- http://www.iu.a.u-tokyo.ac.jp/~kadota/book
/fastaLengthFilter.py
Resolving www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)... 133.11
.224.26
Connecting to www.iu.a.u-tokyo.ac.jp (www.iu.a.u-tokyo.ac.jp)|133.1
1.224.26|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1071 (1.0K) [text/plain]
Saving to: 'fastaLengthFilter.py'

100%[=====>] 1,071      ---K/s   in 0.001s

2017-06-12 14:28:26 (1.11 MB/s) - 'fastaLengthFilter.py' saved [107
1/1071]

iu@bielinux[bin] █
```



W10-2のイントロ4

```
iu@bielinux[~/bin]
iu@bielinux[bin] pwd [ 2:34午後 ]
/home/iu/bin
① iu@bielinux[bin] ls -l [ 2:34午後 ]
total 4
-rw-rw-r-- 1 iu iu 1071 1月 8 2016 fastaLengthFilter.py
② iu@bielinux[bin] head fastaLengthFilter.py [ 2:34午後 ]
#!/usr/bin/env python
# coding:utf-8

import math
import sys

class Fasta():

    @staticmethod
    def read(inputfile):

iu@bielinux[bin] [ 2:34午後 ]
```

W10-2のイントロ5

①chmod 755として、fastaLengthFilter.pyプログラムの実行権限を付与。②ここがXとなっていることを確認。第4回W3-1

```
iu@bielinux[~/bin]
iu@bielinux[bin] pwd [ 2:57午後]
/home/iu/bin
iu@bielinux[bin] ls -l [ 2:57午後]
total 4
-rw-rw-r-- 1 iu iu 1071 1月 8 2016 fastaLengthFilter.py
iu@bielinux[bin] chmod 755 fastaLengthFilter.py [ 2:57午後]
iu@bielinux[bin] ls -l [ 2:57午後]
total 4
-rwxr-xr-x 1 iu iu 1071 1月 8 2016 fastaLengthFilter.py
iu@bielinux[bin] █ [ 2:57午後]
```



W10-2の本番1

①取り扱いたいアセンブリ結果ファイル
(polished_assembly.fasta; スライド34)のある②ディレクトリに移動して、③fastaLengthFilter.pyを実行

```
iu@bielinux[~/Desktop/mac_share/result]
② iu@bielinux[bin] cd ~/Desktop/mac_share/result [ 3:24午後]
iu@bielinux[result] pwd [ 3:25午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls [ 3:25午後]
corrected.fastq ① polished_assembly.fastq
polished_assembly.fasta smrtpipe.log
③ iu@bielinux[result] ~/bin/fastaLengthFilter.py polished_assembly.fasta 0 > LH_hgap.fasta [ 3:25午後]
iu@bielinux[result] █
```


①~/bin/fastaLengthFilter.pyのように書くことで、~/binに「パスを通す」作業をしていなくても実行できる。パスについては、第4回のW9-5やW15-5、第6回のW12-3など

W10-2の本番1

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[bin] cd ~/Desktop/mac_share/result [ 3:24午後]
iu@bielinux[result] pwd [ 3:25午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls [ 3:25午後]
corrected.fastq polished_assembly.fastq
polished_assembly.fasta smrtpipe.log
iu@bielinux[result] ~/bin/fastaLengthFilter.py polished_assembly.fa
sta 0 > LH_hgap.fa
iu@bielinux[result] █ [ 3:25午後]
```



W10-2の本番1

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[bin] cd ~/Desktop/mac_share/result [ 3:24午後]
iu@bielinux[result] pwd [ 3:25午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls [ 3:25午後]
corrected.fastq polished_assembly.fastq
polished_assembly.fasta smrtpipe.log
iu@bielinux[result] ~/bin/fastaLengthFilter.py polished_assembly.fa
sta 0 > LH_hgap.fa
iu@bielinux[result] [ 3:25午後]
```

fastaLengthFilter.pyの①入力と②出力。③出力ファイルのほうが若干サイズが小さくなっている

W10-2の本番2

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:56午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa [ 3:56午後]
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta [ 3:56午後]
iu@bielinux[result] █ [ 3:56午後]
```



W10-2の本番3

multi-FASTA形式の①入力ファイルと②出力ファイルのdescription行部分を表示。fastaLengthFilter.pyは②出力ファイルのdescription行部分をsequence...のようにシンプルに記載する。この文字数の違いもファイルサイズ削減に寄与している

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] █ [ 4:07午後]
```


W10-2の本番4

①入力ファイル(polished_assembly.fasta)は40,567行。②出力ファイル(LH_hgap.fa)は8行。この改行コードのバイト数の違いがファイルサイズの違いに大きく寄与している

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:56午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後 ]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後 ]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後 ]
40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後 ]
8 8 2433662 LH_hgap.fa
iu@bielinux[result] █ [ 4:24午後 ]
```



W10-2の本番4

赤下線部分のファイルサイズの違いは、主に① polished_assembly.fastaの塩基配列情報部分において、60塩基程度ごとに改行が入っているから。具体的には、改行コードが $(40567 - 8) = 40559$ 個分、つまり40559 bytes分だけ大きくなっている

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後]
40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後]
8 8 2433662 LH_hgap.fa
iu@bielinux[result] █ [ 4:24午後]
```



W10-2の本番4

①polished_assembly.fastaの2,474,245 bytesから、 $(40,567 - 8) = 40,559$ bytesを引くと、 $2,474,245 - 40,559 = 2,433,686$ bytesとなる。②LH_hgap.faの2,433,662 bytesと酷似しており妥当

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:56午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後 ]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後 ]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後 ]
40567 40567 2474245 ①lshed_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後 ]
8 8 2433662 ②_hgap.fa
iu@bielinux[result] █ [ 4:24午後 ]
```

W10-2の本番4

①polished_assembly.fastaの2,474,245 bytesから、 $(40,567 - 8) = 40,559$ bytesを引くと、 $2,474,245 - 40,559 = 2,433,686$ bytesとなる。②LH_hgap.faの2,433,662 bytesと酷似しており妥当。 $2,433,686 - ②2,433,662 = 24$ bytesの違いは、赤枠部分の文字数の違いに起因。③6文字×4で24 bytes



```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後]
40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後]
8 8 2433662 LH_hgap.fa
iu@bielinux[result] █ [ 4:24午後]
```

quiver
quiver
quiver
quiver



sequence1
sequence2
sequence3
sequence4



ここまでの話は...

あくまでもこの後の処理をやりやすくするための前処理の話。(何をもってやりやすいと思うかはヒトそれぞれだが)Linuxコマンドを駆使して解析を行う場合は、①LH_hgap.faのように「コンティグ1個あたり2行」で表すほうが取り扱いやすい



```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後]
>sequence1
>sequence2
>sequence3
>sequence4
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後]
40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後]
8 8 2433662 LH_hgap.fa
iu@bielinux[result] █ [ 4:24午後]
```



LH_hgap.fa

①LH_hgap.faは全部で8行からなる。②最初の2行分がsequence1、③最後の2行分(7-8行目)がsequence4の情報となる。fastaLengthFilter.pyを実行したのは、このようにすっきりさせるため

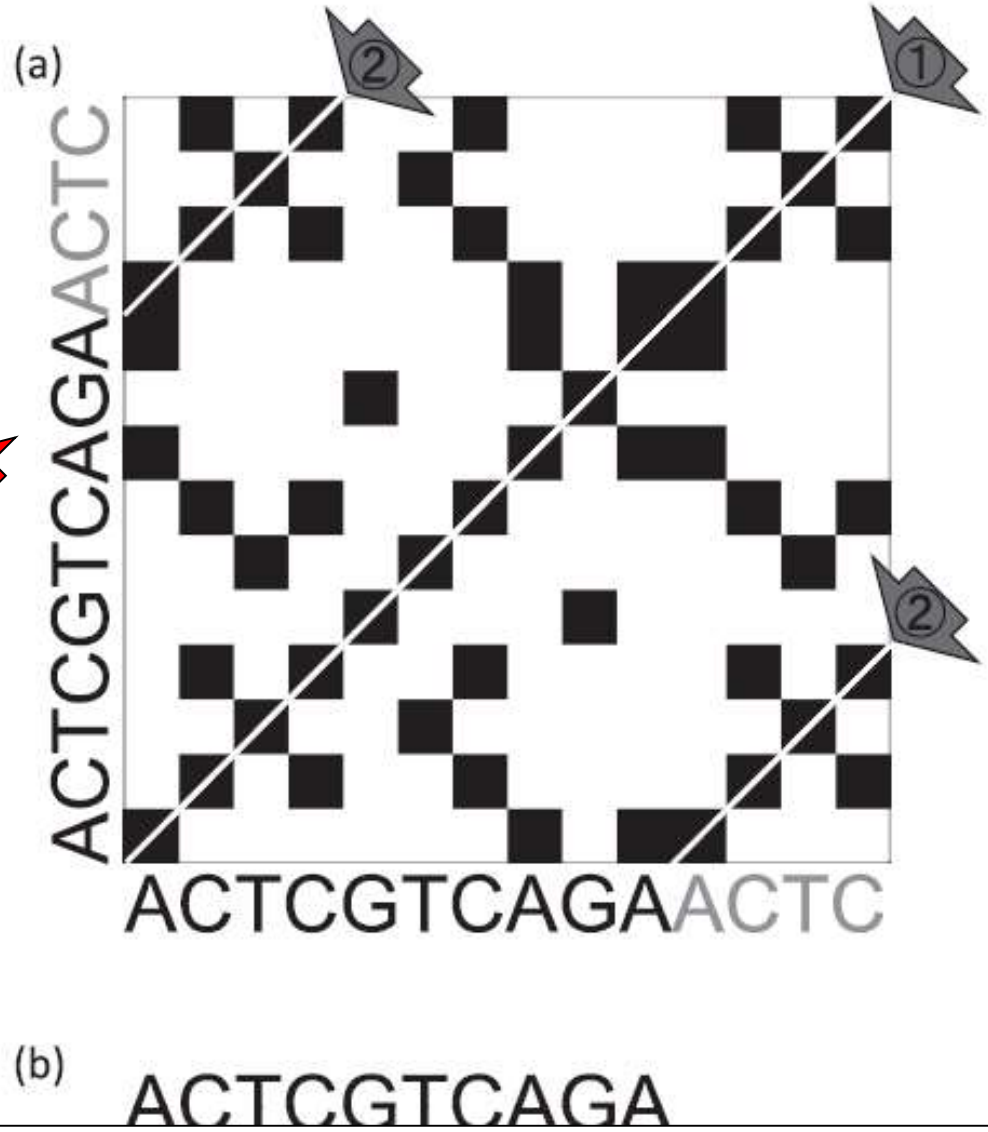
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:56午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.fasta *.fa [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
iu@bielinux[result] grep ">" polished_assembly.fasta [ 3:56午後 ]
>unitig_0|quiver
>unitig_2|quiver
>unitig_3|quiver
>unitig_1|quiver
iu@bielinux[result] grep ">" LH_hgap.fa [ 4:07午後 ]
>sequence1 ②
>sequence2
>sequence3
>sequence4 ③
iu@bielinux[result] wc polished_assembly.fasta [ 4:07午後 ]
40567 40567 2474245 polished_assembly.fasta
iu@bielinux[result] wc LH_hgap.fa [ 4:24午後 ]
8 8 2433662 LH_hgap.fa ①
iu@bielinux[result] █ [ 4:24午後 ]
```


第7回原稿p106左上

今は①赤枠のファイル分割の前処理部分の話。
前処理が終わったので、②Linuxコマンドを組み合わせたファイル分割を行っていきます

コンティグ数が少なくコンティグごとに作業を行う場合は、multi-FASTA ファイルを分割し、コンティグ数分だけ single-FASTA ファイルを作成しておいたほうが効率的な場合もある。ここでは、ファイル分割手段として R を用いるやり方、および自作プログラム (fastaLengthFilter.py; 第6回の W12) と Linux コマンドを組み合わせたやり方を示した [W10]。どちらが正解ということはなく、自分の感性に合う手段を用いればよい。

一般に、HGAP アセンブリ結果として得られるコンティグの末端部分のクオリティは、中央部分に比べて低い。HGAP 実行結果には、FASTA ファイル (polished_assembly.fasta) だけでなく FASTQ ファイル (polished_assembly.fastq) も含まれる。ここでは、FASTQ ファイルを入力として、コンティグごとのクオリティスコア分布を眺めておく。例えば 2 番目に短いコンティグ (45,853 bp; sequence3.fq) のスコア分布の場合 (図 1a; W11-9)、最初の 1,223 bp までと最後の 1,335 bp が連続してスコア 0 になっており、中央の 1,224 bp から 44,518 bp までの計 43,295 bp (=44,518 - 1,224 + 1) がスコア 1 以上になっていると判断できる [W11-12]。これは、コンティグが環状



W10-3: ファイル分割

sequence1は、最初の2行分に相当する。それゆえ、①headコマンドで最初の2行のみ抽出した結果をsequence1.faというファイル名で保存している。それ以外の②grepや③wcはただの確認

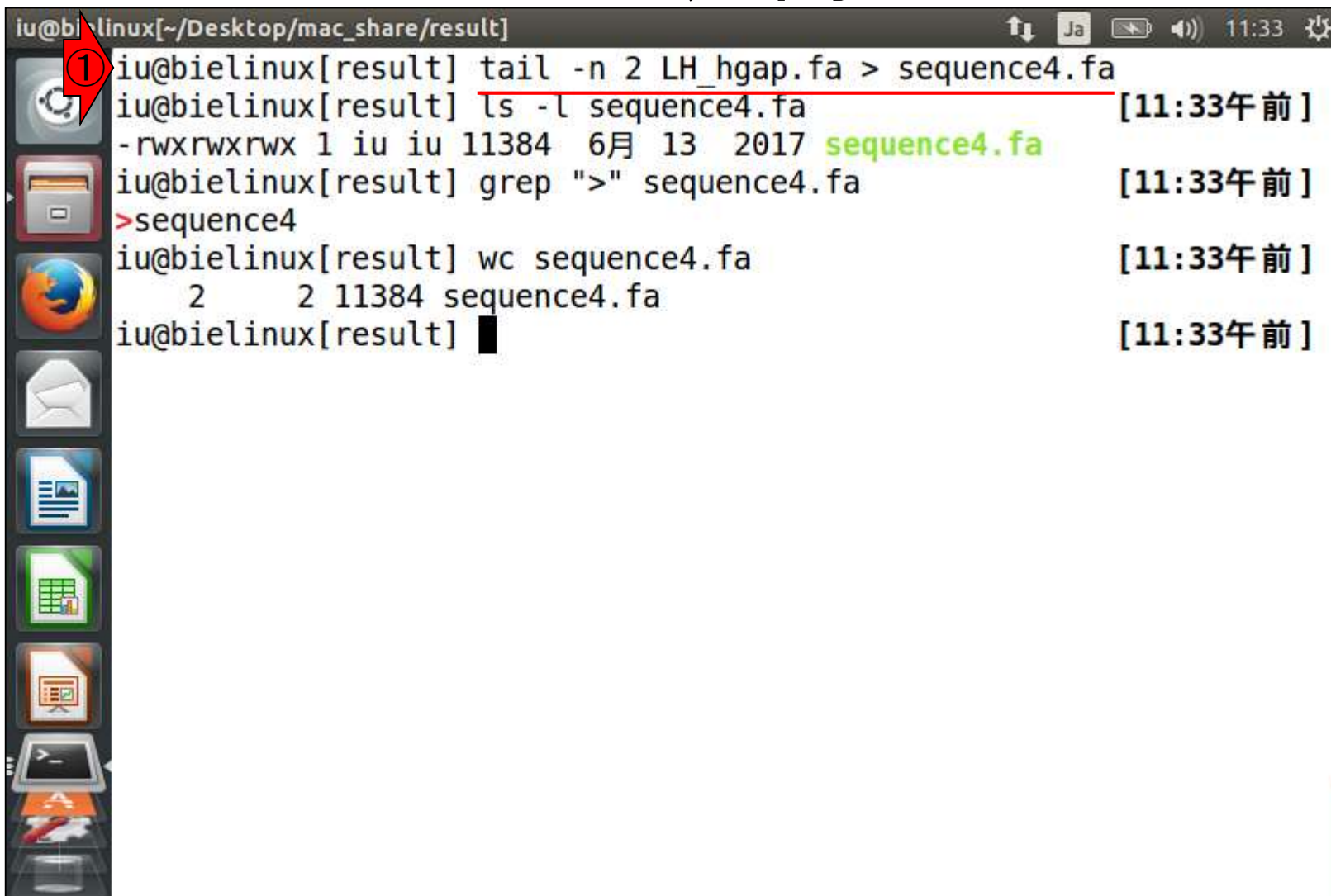
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [11:30午前]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH* [11:30午前]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
① iu@bielinux[result] head -n 2 LH_hgap.fa > sequence1.fa [11:30午前]
iu@bielinux[result] ls -l sequence1.fa [11:30午前]
-rwxrwxrwx 1 iu iu 2289509 6月 13 2017 sequence1.fa
② iu@bielinux[result] grep ">" sequence1.fa [11:30午前]
>sequence1
③ iu@bielinux[result] wc sequence1.fa [11:30午前]
      2      2 2289509 sequence1.fa
iu@bielinux[result] █ [11:30午前]
```


W10-4: ファイル分割2

①sequence2と②sequence3は、headとtailを組み合わせて目的の配列のみ抽出。連載第3回のW19-3にもあり

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [11:32午前]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH* [11:32午前]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
① iu@bielinux[result] head -n 4 LH_hgap.fa | tail -n 2 > sequence2.fa
iu@bielinux[result] ls -l sequence2.fa [11:32午前]
-rwxrwxrwx 1 iu iu 86904 6月 13 2017 sequence2.fa
iu@bielinux[result] grep ">" sequence2.fa [11:32午前]
>sequence2
iu@bielinux[result] wc sequence2.fa [11:32午前]
 2      2 86904 sequence2.fa
iu@bielinux[result] [11:32午前]
② iu@bielinux[result] head -n 6 LH_hgap.fa | tail -n 2 > sequence3.fa
iu@bielinux[result] ls -l sequence3.fa [11:33午前]
-rwxrwxrwx 1 iu iu 45865 6月 13 2017 sequence3.fa
iu@bielinux[result] grep ">" sequence3.fa [11:33午前]
>sequence3
iu@bielinux[result] wc sequence3.fa [11:33午前]
 2      2 45865 sequence3.fa
iu@bielinux[result] █ [11:33午前]
```

W10-4: ファイル分割2



```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] tail -n 2 LH_hgap.fa > sequence4.fa
iu@bielinux[result] ls -l sequence4.fa [11:33午前]
-rwxrwxrwx 1 iu iu 11384 6月 13 2017 sequence4.fa
iu@bielinux[result] grep ">" sequence4.fa [11:33午前]
>sequence4
iu@bielinux[result] wc sequence4.fa [11:33午前]
 2      2 11384 sequence4.fa
iu@bielinux[result] █ [11:33午前]
```

W10-5: lsで確認

①ls -lで配列ごとのファイルサイズを確認。②のように書いてもよい。[0-9]は任意の数字1字という意味。③このファイルサイズは、配列長とほぼ同じ。④例えばsequence1の2,289,509 bytesは

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence*
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 86904 6月 13 11:32 sequence2.fa
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
iu@bielinux[result] ls -l sequence[0-9].fa
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 86904 6月 13 11:32 sequence2.fa
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
iu@bielinux[result]
```



[11:37午前]

[11:38午前]

[11:38午前]

[11:39午前]

W9-2: 結果を眺め

スライド6の①Maximum contig sizeの2,289,497 bpとほぼ同じということです。sequence1の2,289,509 bytesは、「>sequence1」の計10文字と改行コード2文字分からなる、計12 bytes分の情報を余分に含むためと解釈

The screenshot shows the DDBJ pipeline detail view for job ID 21965. The job was executed using HGAP (Protocol3(v 2.2.0)). The assembly statistics section shows 4 contigs with a total size of 2,433,614 bp, a maximum size of 2,289,497 bp, and a minimum size of 11,372 bp. The N50 contig size is also 2,289,497 bp. The command used was 'run HGAP through smrtpipe.py' with parameters GenomeSize=2500000 and minSeedLength=6000. The job started on 2016-03-28 at 20:14:25 and ended on 2016-03-29 at 19:13:20.

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

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



Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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

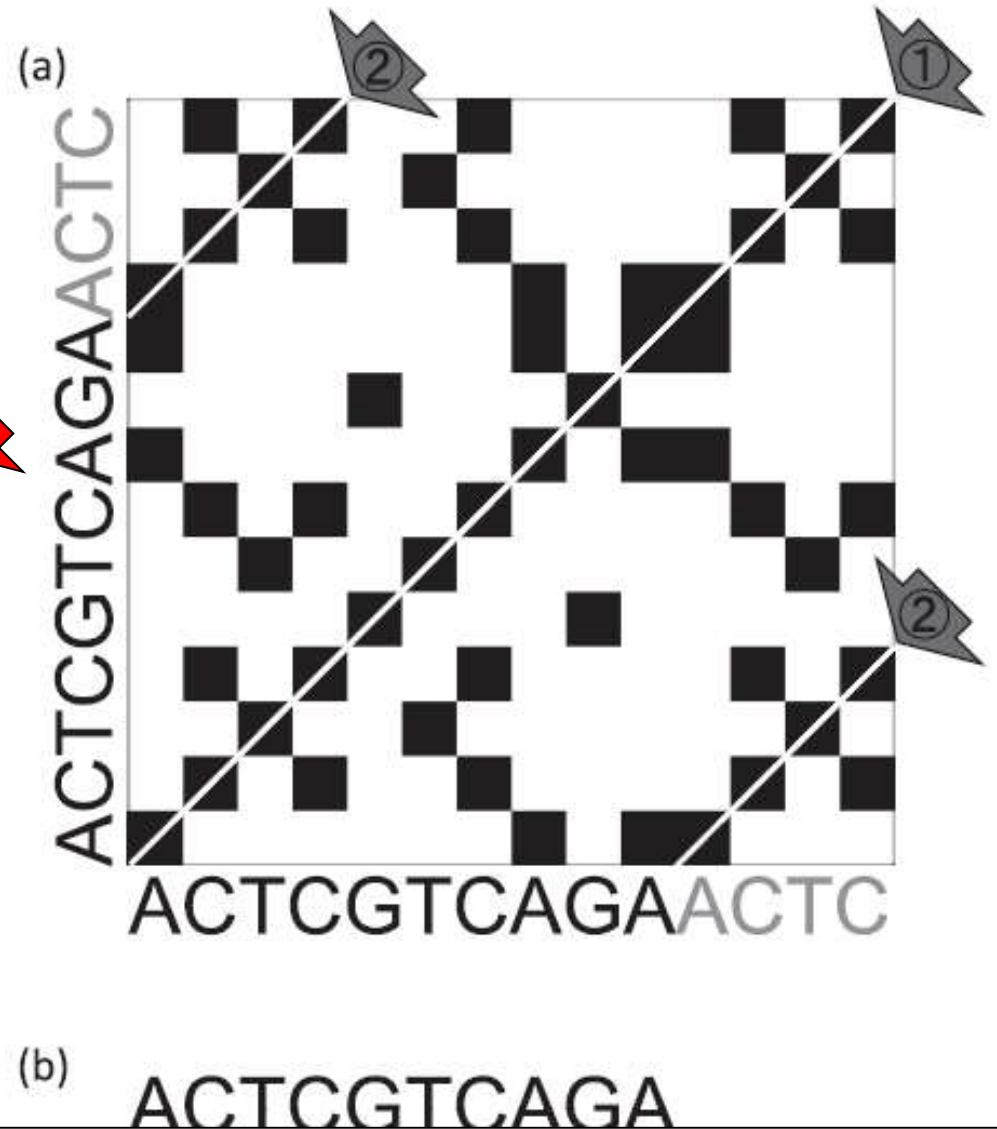


第7回原稿p106左上

①のあたりの話に移行。HGAP実行結果のFASTQファイルを眺めるところからです

コンティグ数が少なくコンティグごとに作業を行う場合は、multi-FASTA ファイルを分割し、コンティグ数分だけ single-FASTA ファイルを作成しておいたほうが効率的な場合もある。ここでは、ファイル分割手段として R を用いるやり方、および自作プログラム (fastaLengthFilter.py; 第6回の W12) と Linux コマンドを組み合わせたやり方を示した [W10]。どちらが正解ということはなく、自分の感性に合う手段を用いればよい。

一般に、HGAP アセンブリ結果として得られるコンティグの末端部分のクオリティは、中央部分に比べて低い。HGAP 実行結果には、FASTA ファイル (polished_assembly.fasta) だけでなく FASTQ ファイル (polished_assembly.fastq) も含まれる。ここでは、FASTQ ファイルを入力として、コンティグごとのクオリティスコア分布を眺めておく。例えば 2 番目に短いコンティグ (45,853 bp; sequence3.fq) のスコア分布の場合 (図 1a; W11-9)、最初の 1,223 bp までと最後の 1,335 bp が連続してスコア 0 になっており、中央の 1,224 bp から 44,518 bp までの計 43,295 bp (=44,518-1,224+1) がスコア 1 以上になっていると判断できる [W11-12]。これは、コンティグが環状



①lsで、②FASTAとFASTQのサイズ比が1:2になっているので妥当

W11-3のイントロ1

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:33午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l [ 4:33午後]
total 80110
-rwxrwxrwx 1 iu iu 69756928 3月 29 2016 corrected.fastq
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 86904 6月 13 11:32 sequence2.fa
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
-rwxrwxrwx 1 iu iu 64556 3月 29 2016 smrtpipe.log
iu@bielinux[result] [ 4:33午後]
```



W11-3のイントロ2

主に①FASTQファイルの行数を確認。②16行。4コンティグというのは既知。1コンティグあたり4行で表されているので、headとtailコマンドの組み合わせでどうにかなるのではと閃く

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:33午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l [ 4:33午後 ]
total 80110
-rwxrwxrwx 1 iu iu 69756928 3月 29 2016 corrected.fastq
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 86904 6月 13 11:32 sequence2.fa
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
-rwxrwxrwx 1 iu iu 64556 3月 29 2016 smrtpipe.log
iu@bielinux[result] wc LH* poli* [ 4:33午後 ]
      8          8 2433662 LH_hgap.fa
 40567  40567 2474245 polished_assembly.fasta
      16         16 4867312 polished_assembly.fastq
 40591  40591 9775219 total
iu@bielinux[result] [ 4:53午後 ]
```



W11-3: FASTQ分割1

①W10-3やW10-4のファイル分割のやり方と若干違うのは、このような記述の仕方でもOKであることを示すとともに、シェルスクリプト利用のイントロ(後述のスライド280~)

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:09午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l polished_assembly.fast* [ 5:09午後 ]
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
iu@bielinux[result] head -n 4 polished_assembly.fastq | tail -n 4 >
sequence1.fq
iu@bielinux[result] head -n 8 polished_assembly.fastq | tail -n 4 >
sequence2.fq
iu@bielinux[result] head -n 12 polished_assembly.fastq | tail -n 4
> sequence3.fq
iu@bielinux[result] head -n 16 polished_assembly.fastq | tail -n 4
> sequence4.fq
iu@bielinux[result] █ [ 5:09午後 ]
```



W11-3: FASTQ分割

①ファイル分割前は、分割後のFASTQファイルが配列長順になっているかどうかは不明であった。②lsでsequence*のfaとfqのサイズが、③約1:2になっていることを確認して配列長順になっていると確信

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:09午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l polished_assembly.fast* [ 5:09午後 ]
-rwxrwxrwx 1 iu iu 2474245 3月 29 2016 polished_assembly.fasta
-rwxrwxrwx 1 iu iu 4867312 3月 29 2016 polished_assembly.fastq
iu@bielinux[result] head -n 4 polished_assembly.fastq | tail -n 4 >
sequence1.fq
iu@bielinux[result] head -n 8 polished_assembly.fastq | tail -n 4 >
sequence2.fq
iu@bielinux[result] head -n 12 polished_assembly.fastq | tail -n 4
> sequence3.fq
iu@bielinux[result] head -n 16 polished_assembly.fastq | tail -n 4
> sequence4.fq
iu@bielinux[result] ls -l sequence* [ 5:09午後 ]
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa }
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq } ③
-rwxrwxrwx 1 iu iu 86904 6月 13 11:32 sequence2.fa }
-rwxrwxrwx 1 iu iu 173805 6月 13 17:09 sequence2.fq } ③
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa }
-rwxrwxrwx 1 iu iu 91727 6月 13 17:09 sequence3.fq } ③
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa }
-rwxrwxrwx 1 iu iu 22765 6月 13 17:09 sequence4.fq } ③
iu@bielinux[result] [ 5:12午後 ]
```


W11-3: FASTQ分割2

FASTQファイルのdescription部分の行頭は①@および②+なので、念のため両方で調べている。③行数は全部4行


```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:28午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls *.fq [ 5:28午後]
sequence1.fq sequence2.fq sequence3.fq sequence4.fq
① iu@bielinux[result] grep "^@" sequence*.fq [ 5:28午後]
sequence1.fq:@unitig_0|quiver
sequence2.fq:@unitig_2|quiver
sequence3.fq:@unitig_3|quiver
sequence4.fq:@unitig_1|quiver
② iu@bielinux[result] grep "^+" sequence*.fq [ 5:28午後]
sequence1.fq:+
sequence2.fq:+
sequence3.fq:+
sequence4.fq:+
③ iu@bielinux[result] wc sequence*.fq [ 5:28午後]
   4      4 4579015 sequence1.fq
   4      4  173805 sequence2.fq
   4      4   91727 sequence3.fq
   4      4   22765 sequence4.fq
  16     16 4867312 total
iu@bielinux[result] █ [ 5:28午後]
```

Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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



Tips: LinuxでR



National Bioscience Database Center

- 散在するデータベースを、まとめて、使い易く -

バイオサイエンスデータベースセンター

English サイトマップ

国立研究開発法人 科学技術振興機構
JST
文字サイズ変更 大 中 小

Search for... Search

- ホーム
- NBDCについて
- 研究開発
- 公募情報
- 採用情報
- イベント
- 人材支援
- アクセス
- リンク

Home > 人材支援 > 支援 > 講習会 > 平成28年度NGSハンズオン講習会カリキュラム



H28年度 NGSハンズオン講習会カリキュラム

H28年度日程・講義資料・動画等

カリキュラム (PDF: 72KB)

実施日	時間	部	講義内容	講師	資料
8月1日 (月)	10:30-18:15	第3部 NGS解析 (中～上級) (農学生命情報科学特論II)	Linux環境でのデータ解析: JavaやRの利用法	門田 幸二 (東京大学)	講義資料 (PDF:11.7MB) 統合TV
7月19日 (火)					
8月2日 (火)	10:30-18:15		Linux環境でのデータ解析: マッピング、トリミング、アセンブリ		講義資料 (PDF:11.9MB) 統合TV



W11-4: スコア分布

赤枠部分がFASTQファイル中の1文字表記のクオリティスコアを数値化 (PHREDスコアに変換) して保存するRコード。講習会用に
①ShortReadパッケージをインストール済み

W11-4: スコア分布 (スライド 70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))にて、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```
R -q
in_f <- "sequence4.fq"           #入力ファイル名を指定してin_fに格納
out_f1 <- "sequence4.png"        #出力ファイル名を指定してout_f1に格納
out_f2 <- "sequence4.txt"        #出力ファイル名を指定してout_f2に格納
param_fig <- c(700, 350)         #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード
library(ShortRead)               #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f)         #in_fで指定したファイルの読み込み
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix") #ASCIIコードのquality scoreをPHRED scoreに変換し
colnames(out) <- 1:ncol(out)      #列名を付与
rownames(out) <- as.character(id(fastq)) #行名を付与
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パ
par(mar=c(4, 4, 0, 0))           #下、左、上、右の順で余白(行)を指定
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
      type="p", xlab="position", ylab="PHRED score") #プロット
dev.off()                         #おまじない
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの
```

W11-5: 入出力の関係

これは、①sequence4.fq(一番短い11,372 bpのコンティグ)を入力ファイルとして、②2つのファイルを出力するコード

W11-4: スコア分布 (スライド 70)

「前処理 | クオリティチェック | [PHREDスコアに変換](#)」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```
R -q
in_f <- "sequence4.fq" #入力ファイル名を指定してin_fに格納
out_f1 <- "sequence4.png" #出力ファイル名を指定してout_f1に格納
out_f2 <- "sequence4.txt" #出力ファイル名を指定してout_f2に格納
param_fig <- c(700, 350) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード
library(ShortRead) #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f) #in_fで指定したファイルの読み込み
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix") #ASCIIコードのquality scoreをPHRED scoreに変換し
colnames(out) <- 1:ncol(out) #列名を付与
rownames(out) <- as.character(id(fastq)) #行名を付与
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パ
par(mar=c(4, 4, 0, 0)) #下、左、上、右の順で余白(行)を指定
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
      type="p", xlab="position", ylab="PHRED score") #プロット
dev.off() #おまじない
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの
```



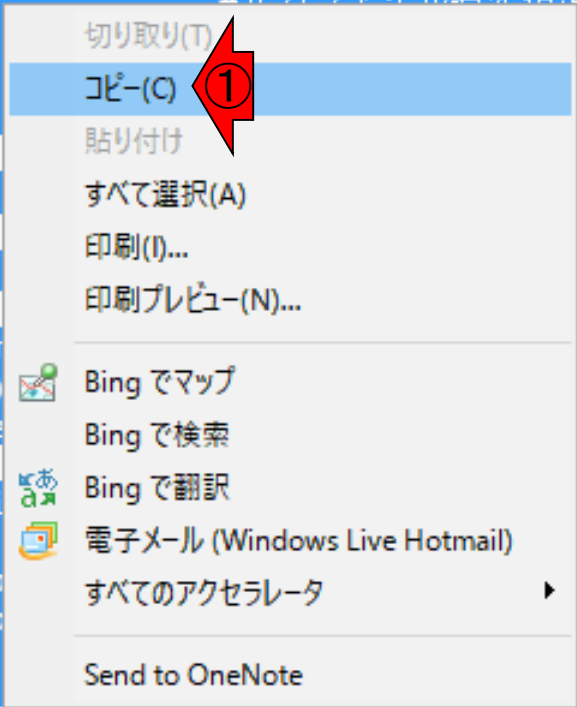
コピー

コードの枠内を全選択 (Windowsのヒトはトリプルクリックで全選択できます。CTRLキーを押しながらワンクリックでもOK)して、右クリックで①コピー

W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```
R -q
in_f <- "sequence4.fq"
out_f1 <- "sequence4.png"
out_f2 <- "sequence4.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix")
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2],
     par(mar=c(4, 4, 0, 0)))
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, type="p", xlab="position",
     dev.off())
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out))#保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F)#tmpの
```



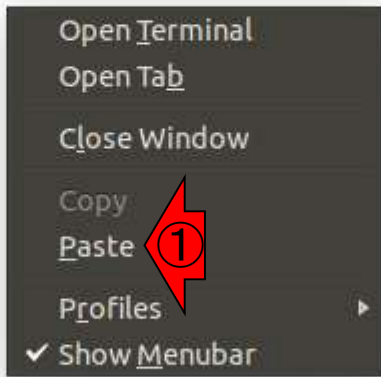
コピペ

- W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、pngファイル作成(描画)時に、cex=0.5は大きさを通常

```
R -q
in_f <- "sequence4.f
out_f1 <- "sequence4
out_f2 <- "sequence4
param_fig <- c(700,
#必要なパッケージをロー
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(i
#本番(PHREDスコアに変
out <- as(quality(fa
colnames(out) <- 1:n
rownames(out) <- as.
#ファイルに保存(pngフ
png(out_f1, pointsiz
par(mar=c(4, 4, 0, 0
plot(x=1:ncol(out),
type="p", xlab=
dev.off()
#ファイルに保存(テキス
tmp <- cbind(colname
write.table(tmp, out
```

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence4*
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
-rwxrwxrwx 1 iu iu 22765 6月 13 17:09 sequence4.fq
iu@bielinux[result]
```



W11-6: 実行結果

コピー実行結果後に①lsで確認(自動でここまでなっているはず)。②確かに指定した名前の2つのファイルが作成されていることがわかる

- W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的pngファイル作成(描画)時により、cex=0.5は大きさを通常

```
R -q
in_f <- "sequence4.f
out_f1 <- "sequence4
out_f2 <- "sequence4
param_fig <- c(700,
#必要なパッケージをロー
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(i
#本番(PHREDスコアに変
out <- as(quality(fa
colnames(out) <- 1:n
rownames(out) <- as.
#ファイルに保存(pngフ
png(out_f1, pointsiz
par(mar=c(4, 4, 0, 0
plot(x=1:ncol(out),
type="p", xlab=
dev.off()
#ファイルに保存(テキス
tmp <- cbind(colname
write.table(tmp, out
```

```
iu@bielinux[~/Desktop/mac_share/result]
) #出力ファイルの各種パラメータを指定
> par(mar=c(4, 4, 0, 0)) #下、左、上、右の順で余白
(行)を指定
> plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
+ type="p", xlab="position", ylab="PHRED score") #プロット
> dev.off() #おまじない
null device
1
> #ファイルに保存(テキストファイル)
> tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
> write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F,
, col.names=F) #tmpの中身を指定したファイル名で保存
> q(save="no")
iu@bielinux[result] pwd [ 1:49午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence4* [ 1:49午後 ]
-rwxrwxrwx 1 iu iu 11384 6月 13 11:33 sequence4.fa
-rwxrwxrwx 1 iu iu 22765 6月 13 17:09 sequence4.fq
-rwxrwxrwx 1 iu iu 24763 6月 14 2017 sequence4.png
-rwxrwxrwx 1 iu iu 86006 6月 14 2017 sequence4.txt
iu@bielinux[result]
```



余談1

Rも自動で終了し、pwdや①ls -l sequence4*まで実行されたのは、①コピーしたコードの最後のほうに、②のような感じで書きこんでいるからです

W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | [PHREDスコアに変換](#)」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```

param_fig <- c(700, 350) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード
library(ShortRead) #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f) #in_fで指定したファイルの読み込み
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix") #ASCIIコードのquality scoreをPHRED scoreに変換し
colnames(out) <- 1:ncol(out) #列名を付与
rownames(out) <- as.character(id(fastq)) #行名を付与
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種バ
par(mar=c(4, 4, 0, 0)) #下、左、上、右の順で余白(行)を指定
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
      type="p", xlab="position", ylab="PHRED score") #プロット
dev.off() #おまじない
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの
q(save="no")
pwd
ls -l sequence4*
    
```

①

②

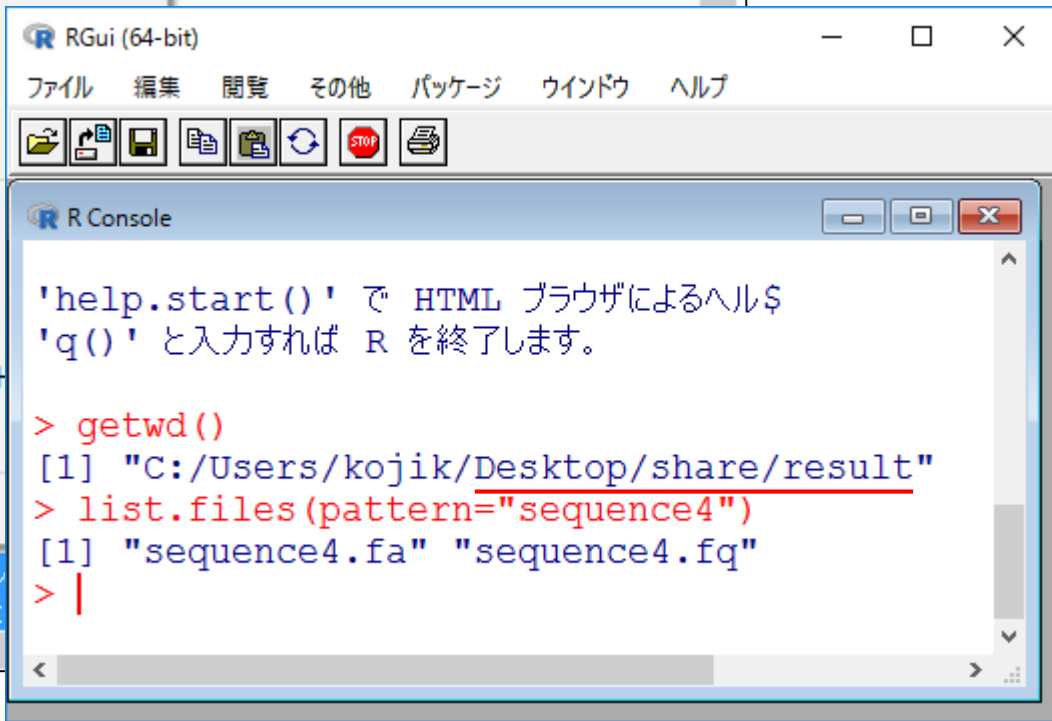
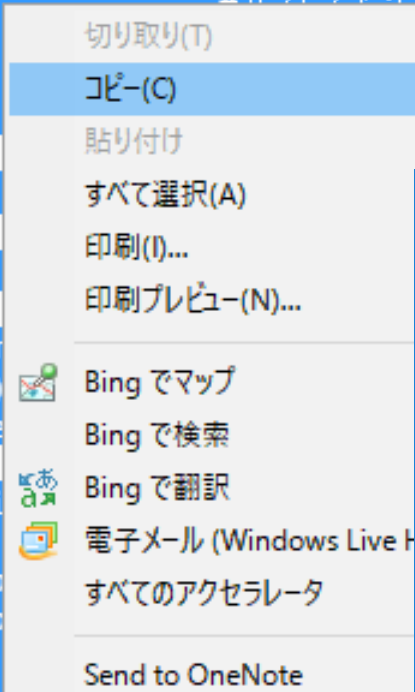
余談2

もちろんペーストは、①ShortReadパッケージがインストールされていれば、ホストOS(この場合はWindows10)上のRでやっても構いません。これが共有フォルダの便利なところ

• W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```
R -q
in_f <- "sequence4.fq"
out_f1 <- "sequence4.png"
out_f2 <- "sequence4.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix")
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2],
par(mar=c(4, 4, 0, 0)))
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, type="p", xlab="position",
dev.off())
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out))#保存し
write.table(tmp, out_f2, sep="\t", append=F, quot
```



W11-7: pngファイル

①pngファイルのほうは、②で横幅700 pixelと縦幅350 pixelにサイズ指定しているの横長になっている。横軸はコンティグのposition、縦軸はPHREDスコア。数値が大きいほどクオリティが高い。尚、Linux上で出力ファイル(sequence4.png)を開く場合は、左側の引き出しアイコンから辿っていくのもよいし、「eog sequence4.png」と打つのもよい

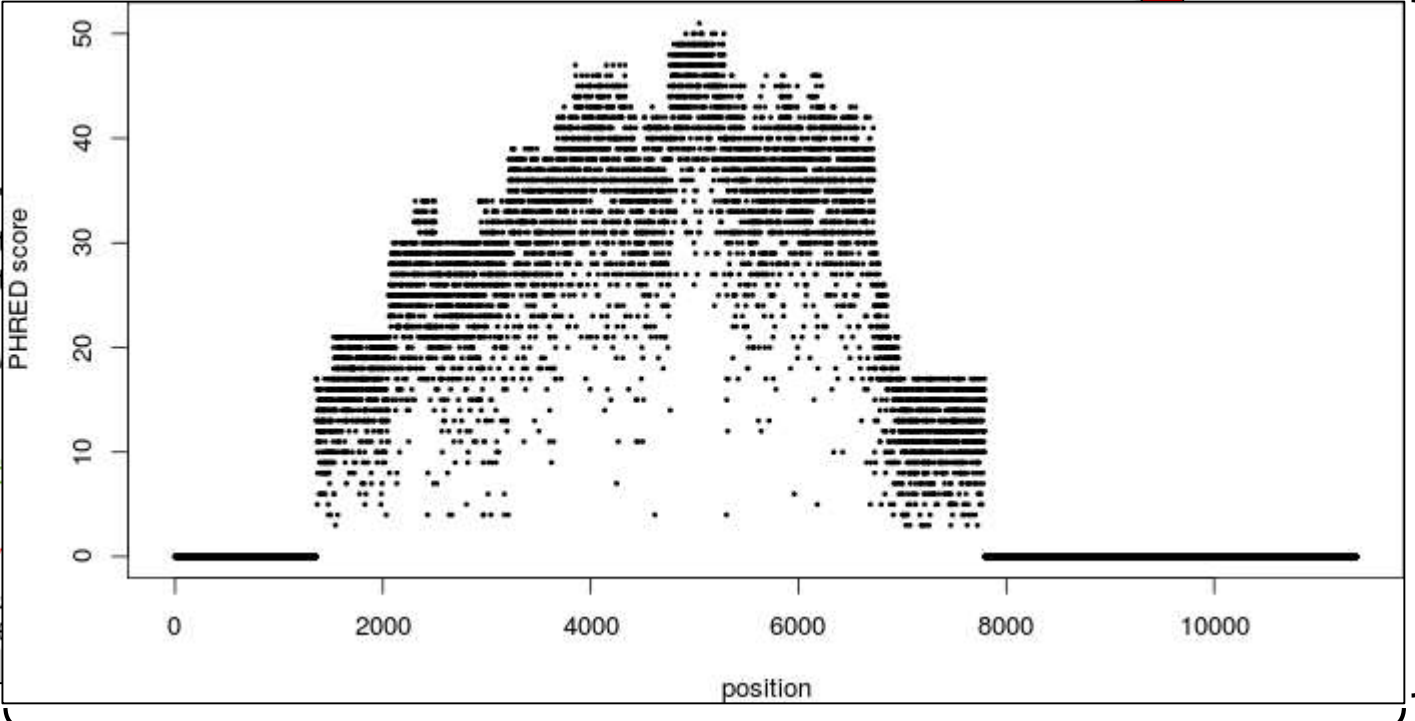
W11-4:スコア分布 (スライド70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考し調整もしています。具体的には、図の下と左側を4行分、それ以外pngファイル作成(描画)時にいろいろオプション指定している。pch=2よ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット

```
R -q
in_f <- "sequence4.fq"
out_f1 <- "sequence4.png"
out_f2 <- "sequence4.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq),
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13,
par(mar=c(4, 4, 0, 0))
plot(x=1:ncol(out), y=out,
type="p", xlab="position",
dev.off()
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out),
write.table(tmp, out_f2, se
```



#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#ファイル出力時の横幅と縦幅を指定(単位はピクセル)



350 pixel

700 pixel

W11-8: テキストファイル

• W11-4: スコア分布 (スライド 70)

「前処理 | クオリティチェック | [PHREDスコアに変換](#)」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的には、図の下と左側を4行分、それ以外を0行分だけ開けるように指定しています。pngファイル作成(描画)時にいろいろオプション指定している。pch=20はプロット時のマーカーを「小さい黒丸」にせよ、cex=0.5は大きさを通常の0.5倍にせよ、type="p"は、「点プロット(デフォルト)」にせよ、という意味です。

```
R -q
in_f <- "sequence4.fq"           #入力ファイル名を指定してin_fに格納
out_f1 <- "sequence4.png"       #出力ファイル名を指定してout_f1に格納
out_f2 <- "sequence4.txt"       #出力ファイル名を指定してout_f2に格納
param_fig <- c(700, 350)       #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード

library(ShortRead)             #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f)       #in_fで指定したファイルの読み込み
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix") #ASCIIコードのquality scoreをPHRED scoreに変換し
colnames(out) <- 1:ncol(out)    #列名を付与
rownames(out) <- as.character(id(fastq)) #行名を付与
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パ
par(mar=c(4, 4, 0, 0))        #下、左、上、右の順で余白(行)を指定
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
     type="p", xlab="position", ylab="PHRED score") #プロット
dev.off()                       #おまじない
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの
```

①最初の5行分と②最後の4行分を表示。③1列目はposition番号、④2列目がPHREDスコア

W11-8: テキストファイル

- W11-4: スコア分布 (スライド 70)

「前処理 | クオリティチェック | PHREDスコアに変換」の例題3を参考にしています。par(mar=c(4, 4, 0, 0))で、余白の調整もしています。具体的pngファイル作成(描画)時に、cex=0.5は大きさを通常

```
R -q
in_f <- "sequence4.fq"
out_f1 <- "sequence4.png"
out_f2 <- "sequence4.txt"
param_fig <- c(700, 100)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=param_fig,
     par(mar=c(4, 4, 0, 0)),
     plot(x=1:ncol(out), y=1:nrow(out),
          type="p", xlab="Position",
          dev.off())
#ファイルに保存(テキストファイル)
tmp <- cbind(colname=colnames(out),
             score=as.numeric(out))
write.table(tmp, out_f2,
            sep=" ",
            as.is=TRUE,
            row.names=FALSE,
            col.names=FALSE,
            append=FALSE)
```

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence4*
-rwxrwxrwx 1 iu iu 11384  6月 13 11:33 sequence4.fa
-rwxrwxrwx 1 iu iu 22765  6月 13 17:09 sequence4.fq
-rwxrwxrwx 1 iu iu 24763  6月 14 13:49 sequence4.png
-rwxrwxrwx 1 iu iu 86006  6月 14 13:49 sequence4.txt
iu@bielinux[result] head -n 5 sequence4.txt
1      0
2      0
3      0
4      0
5      0
iu@bielinux[result] tail -n 4 sequence4.txt
11369  0
11370  0
11371  0
11372  0
iu@bielinux[result] █
```

① points to line 1 of the head command output.

② points to line 11369 of the tail command output.

③ points to the first column (position number) of the tail output.

④ points to the second column (PHRED score) of the tail output.

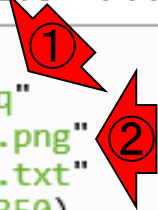
W11-9: sequence3.fq

①2番目に短い45,853 bpのファイル (sequence3.fq)に対しても同様な作業を実行。
②pngファイルを眺めると、確かに③コンティグ両末端部分のクオリティが低いことがわかる

W11-9: sequence3.fq (スライド 80)

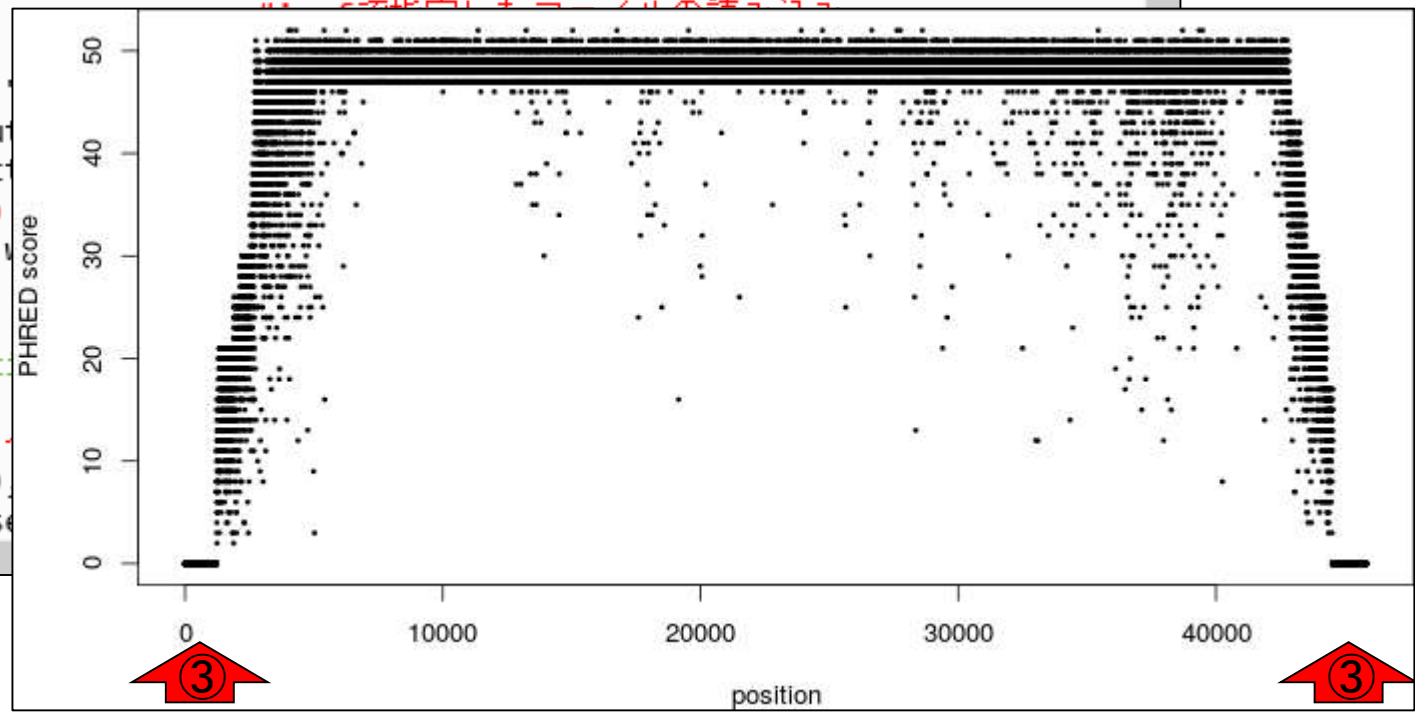
W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq),
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13,
par(mar=c(4, 4, 0, 0))
plot(x=1:ncol(out), y=out,
type="p", xlab="position",
dev.off()
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out),
write.table(tmp, out_f2, sep="|",
```



#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#パッケージの読み込み



クオリティスコアが0なのは、最初から①1,000塩基目ちよつと、②最後から1,500塩基ちよつと(44,000塩基目)くらいかな…とか妄想し、ある程度あたりをつけておいて、③sequence3.txtをlessで眺めたりしておく

W11-9: sequence3

W11-9: sequence3.fq (スライド 80)

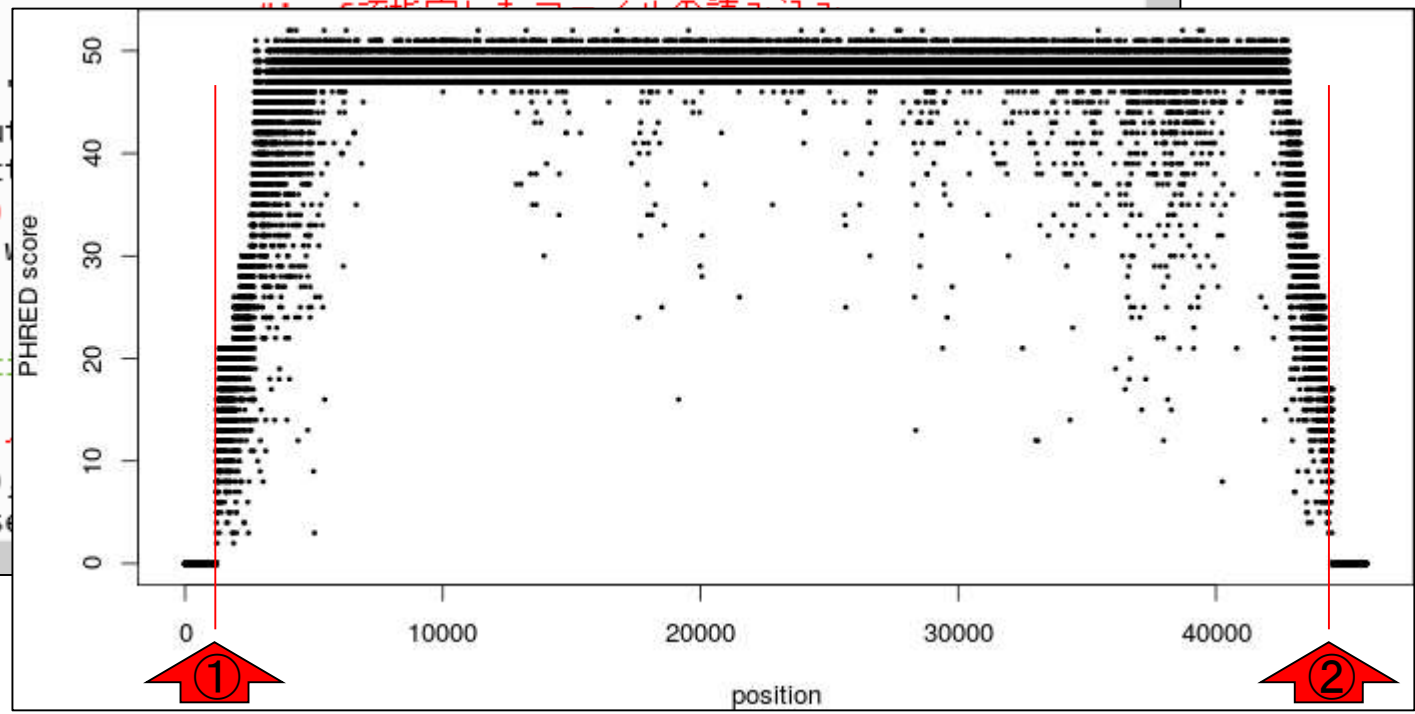
W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequences.png](#)と[sequences.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq),
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13,
par(mar=c(4, 4, 0, 0))
plot(x=1:ncol(out), y=out,
type="p", xlab="position",
dev.off()
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out),
write.table(tmp, out_f2, sep="")
```



#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#パッケージの読み込み



less sequence3.txt

- W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=param_fig,
     par(mar=c(4, 4, 0, 0)))
plot(x=1:ncol(out), y=1:nrow(out),
     type="p", xlab="Position",
     dev.off())
#ファイルに保存(テキスト)
tmp <- cbind(colnames(out), out)
write.table(tmp, out_f2,
            sep=" ",
            as.is=TRUE,
            append=FALSE,
            row.names=FALSE,
            col.names=TRUE,
            quote=FALSE)
```

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 91727 6月 13 17:09 sequence3.fq
-rwxrwxrwx 1 iu iu 20878 6月 14 16:46 sequence3.png
-rwxrwxrwx 1 iu iu 398859 6月 14 16:46 sequence3.txt
iu@bielinux[result] less sequence3.txt
```



less sequence3.txt

less実行直後。①1列目がposition番号(塩基番号)なので、ここが1,000ちょっとくらいになるまで、ざばーっと「スペース」キーで進み、② PHREDスコアが0より大きくなる場所を探る。残りは上下矢印キーで微調整

W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=10, width=param_fig[1], height=param_fig[2],
     par(mar=c(4, 4, 0, 0)), type="p", xlab="Position", ylab="Quality")
#ファイルに保存(テキストフォーマット)
tmp <- cbind(colnames(out), out)
write.table(tmp, out_f2, as.is=T, row.names=F, col.names=F)
```

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0

sequence3.txt

①1,224番目の塩基以降から0以上のPHREDスコアになっているようですね

less sequence3.txt

W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=10, width=1000, height=1000,
     par(mar=c(4, 4, 0, 0)),
     plot(x=1:ncol(out), y=1:nrow(out),
          type="p", xlab="Position", ylab="Quality"),
     dev.off())
#ファイルに保存(テキストフォーマット)
tmp <- cbind(rownames(out), out)
write.table(tmp, out_f2, sep=" ", as.is=TRUE)
```

```
iu@bielinux[~/Desktop/mac_share/result]
1214 0
1215 0
1216 0
1217 0
1218 0
1219 0
1220 0
1221 0
1222 0
1223 0
1224 3
1225 7
1226 11
1227 11
1228 7
1229 6
1230 7
1231 4
1232 12
1233 16
1234 15
:
```



less sequence3.txt

「G」と打って、最終行に移動したところ。
この状態からキーボードのuを押して効率的にページ上部を探索する

- W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=param_fig,
     par(mar=c(4, 4, 0, 0)),
     plot(x=1:ncol(out), y=1:nrow(out),
          type="p", xlab="Position",
          dev.off())
#ファイルに保存(テキスト)
tmp <- cbind(colnames(out), out)
write.table(tmp, out_f2, as.is=T)
```

```
iu@bielinux[~/Desktop/mac_share/result]
45833 0
45834 0
45835 0
45836 0
45837 0
45838 0
45839 0
45840 0
45841 0
45842 0
45843 0
45844 0
45845 0
45846 0
45847 0
45848 0
45849 0
45850 0
45851 0
45852 0
45853 0
(END)
```

①44,518番目の塩基までが0以上のPHREDスコアになっているようです

less sequence3.txt

W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq))
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=10, width=1000, height=1000,
     par(mar=c(4, 4, 0, 0)),
     plot(x=1:ncol(out), y=1:nrow(out),
          type="p", xlab="Position", ylab="Quality"),
     dev.off())
#ファイルに保存(テキストフォーマット)
tmp <- cbind(colnames(out), out)
write.table(tmp, out_f2, as.is=T, sep="\t",
```

Position	Quality Score
1	3
2	8
3	11
4	15
5	17
6	15
7	17
8	16
9	15
10	13
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0

less sequence3.txt

- W11-9: sequence3.fq (スライド 80)

W11-4と基本的に同じで入出力のみ異なる。出力ファイルは、[sequence3.png](#)と[sequence3.txt](#)。

```
R -q
in_f <- "sequence3.fq"
out_f1 <- "sequence3.png"
out_f2 <- "sequence3.txt"
param_fig <- c(700, 700)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "int")
colnames(out) <- 1:nrow(out)
rownames(out) <- as.character(1:nrow(out))
#ファイルに保存(pngフォーマット)
png(out_f1, pointsize=param_fig,
     par(mar=c(4, 4, 0, 0)),
     plot(x=1:ncol(out), y=1:nrow(out),
          type="p", xlab="Sequence",
          dev.off())
#ファイルに保存(テキスト)
tmp <- cbind(colnames(out), out)
write.table(tmp, out_f2,
            as.is=T,
            sep=" ",
            col.names=FALSE,
            row.names=FALSE,
            append=FALSE,
            quote=F,
            fileEncoding="UTF-8")
```

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 91727 6月 13 17:09 sequence3.fq
-rwxrwxrwx 1 iu iu 20878 6月 14 16:46 sequence3.png
-rwxrwxrwx 1 iu iu 398859 6月 14 16:46 sequence3.txt
iu@bielinux[result] less sequence3.txt
iu@bielinux[result] █
```

[5:01午後]

[5:02午後]

[5:02午後]

[5:34午後]

第7回原稿p106左中

assembly.fastq) も含まれる。ここでは、FASTQ ファイルを入力として、コンティグごとのクオリティスコア分布を眺めておく。例えば2番目に短いコンティグ (45,853 bp; sequence3.fq) のスコア分布の場合 (図 1a; W11-9)、最初の 1,223 bp までと最後の 1,335 bp が連続してスコア 0 になっており、中央の 1,224 bp から 44,518 bp までの計 43,295 bp (=44,518-1,224+1) がスコア 1 以上になっていると判断できる [W11-12]。これは、コンティグが環状であることを確認したあとに、どの部分をトリミングするかについての合理的な指針を与えるものでもある。



(b)

```

ACTCGTTCAGA
CTCGTTCAGAA
TCGTTCAGAAC*
CGTTCAGAACT
GTCAGAACTC
    
```

図 2. (a) 仮想環状コンティグ配列 ACTCGTTCAGAACTC のドットプロット。灰色で示した最後の 4 塩基が最初の 4 塩基と同じで、重複領域に相当する。(b) 環状なので重複除去手段は計 5 通り。*のついた前後 2 塩基づつトリムするやり方が推奨。

配列のドットプロット

ドットプロット (dot plot) は、比較したい 2 つの配列の類似度を視覚的に評価するために古くから用いられている描画手段である¹⁶⁾。基本的には、比較する 2 つの配列を x 軸 y 軸にそれぞれ並べて、同一塩基部分をハイライトさせるだけである。ここでは、単純な塩基配列を用いたドットプロットの解釈の基礎を述べ、実際の環状コンティグ (またはゲノム) の実例を sequence3 で示す。Bio-

グの選択肢として、4 塩基重複の場合は計 5 通り存在する

Contents (第7回後半分)

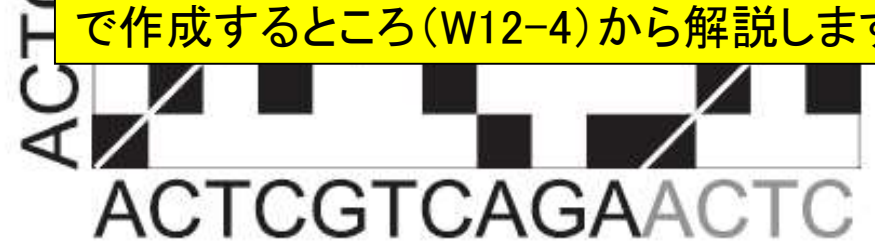
- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 環状化(ゲノム解読のfinishing作業の一部)
 - アセンブリ結果として、最初と最後の末端部分が同じ配列の場合は、通常そのコンティグは環状と判断。それを確認するための基本的な考え方、手段、および環状化のノウハウを伝授
 - W12: seqinrパッケージを用いて、仮想環状コンティグのドットプロットで感覚をつかむ
 - W13: 重複配列の除去の感覚をつかむ(これが環状化作業の実体)
 - W14: dotterプログラムで実際の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



第7回原稿p106左下

①コンティグが環状になっているかどうかを調べる基本的な戦略が②ドットプロット。③単純な塩基配列のFASTAファイルをLinux上で作成するところ(W12-4)から解説します

assembly.fastq) も含まれる。ここでは、FASTQ ファイルを入力として、コンティグごとのクオリティスコア分布を眺めておく。例えば2番目に短いコンティグ (45,853 bp; sequence3.fq) のスコア分布の場合 (図 1a; W11-9)、最初の 1,223 bp までと最後の 1,335 bp が連続してスコア 0 になっており、中央の 1,224 bp から 44,518 bp までの計 43,295 bp (=44,518-1,224+1) がスコア 1 以上になっていると判断できる [W11-12]。これは、コンティグが環状であることを確認したあとに、どの部分をトリミングするかについての合理的な指針を与えるものでもある。



(b)
ACTCGTCAGA
CTCGTCAGAA
TCGTCAGAAC*
CGTCAGAACT
GTCAGAACTC

図 2. (a) 仮想環状コンティグ配列 ACTCGTCAGAACTC のドットプロット。灰色で示した最後の 4 塩基が最初の 4 塩基と同じで、重複領域に相当する。(b) 環状なので重複除去手段は計 5 通り。*のついた前後 2 塩基づつトリムするやり方が推奨。

配列のドットプロット

ドットプロット (dot plot) は、比較したい 2 つの配列の類似度を視覚的に評価するために古くから用いられている描画手段である¹⁶⁾。基本的には、比較する 2 つの配列を x 軸 y 軸にそれぞれ並べて、同一塩基部分を **③** ライトさせるだけである。ここでは、単純な塩基配列を用いたドットプロットの解釈の基礎を述べ、実際の環状コンティグ (またはゲノム) の実例を sequence3 で示す。Bio-

グの選択肢として、4 塩基重複の場合は計 5 通り存在する

W12-4: hoge.fa

seqinrパッケージのdotPlot関数実行時に入力として用いるファイルhoge.faを作成する。①と②と③は作成したいhoge.faがないことを確認しているだけ

```
iu@bielinux[~/Desktop/mac_share/result]
① iu@bielinux[result] pwd [ 2:17午後 ]
/home/iu/Desktop/mac_share/result
② iu@bielinux[result] ls -l hoge.fa [ 2:17午後 ]
ls: cannot access hoge.fa: No such file or directory
③ iu@bielinux[result] more hoge.fa [ 2:17午後 ]
hoge.fa: No such file or directory
iu@bielinux[result] █ [ 2:17午後 ]
```


W12-4: hoge.fa

①echoコマンドは、第4回W9でも利用。ここでは任意の文字列「>test」を表示させているだけ。
②リダイレクト(>)でhoge.faファイルを新規作成して書き込み。③moreでhoge.faの中身を表示

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 2:17午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l hoge.fa [ 2:17午後 ]
ls: cannot access hoge.fa: No such file or directory
iu@bielinux[result] more hoge.fa [ 2:17午後 ]
hoge.fa: No such file or directory
① iu@bielinux[result] echo ">test" [ 2:17午後 ]
>test
② iu@bielinux[result] echo ">test" > hoge.fa [ 2:22午後 ]
iu@bielinux[result] more hoge.fa [ 2:22午後 ]
③ >test
iu@bielinux[result] █ [ 2:22午後 ]
```


W12-4: hoge.fa

①ACTCGTCAGAという文字列をhoge.faに追加書き込み。②既存のファイルに追加書き込みをするときは>>でした(第4回W12-2;第5回W17-2)。③これでFASTA形式ファイルの完成

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 2:17午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l hoge.fa [ 2:17午後 ]
ls: cannot access hoge.fa: No such file or directory
iu@bielinux[result] more hoge.fa [ 2:17午後 ]
hoge.fa: No such file or directory
iu@bielinux[result] echo ">test" [ 2:17午後 ]
>test
iu@bielinux[result] echo ">test" > hoge.fa [ 2:22午後 ]
iu@bielinux[result] more hoge.fa [ 2:22午後 ]
>test
iu@bielinux[result] echo "ACTCGTCAGA" >> hoge.fa [ 2:22午後 ]
iu@bielinux[result] more hoge.fa [ 2:30午後 ]
>test
ACTCGTCAGA
iu@bielinux[result] █ [ 2:30午後 ]
```



W12-5: dotPlot実行

- W12-5: dotPlot実行 (スライド 94)

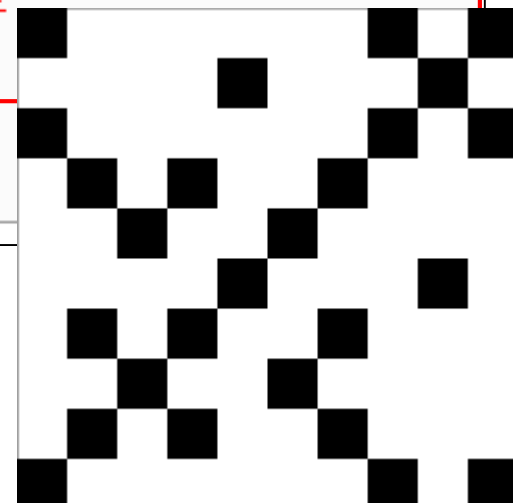
[seqinr](#)パッケージ中のdotPlot関数を用いてドットプロットを作成。

```
cd ~/Desktop/mac_share/result
```

```
R -q
```

```
in_f <- "hoge.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.png"        #出力ファイル名を指定してout_fに格納
param_fig <- c(300, 300)    #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード
library(seqinr)             #パッケージの読み込み
#入力ファイルの読み込み
hoge <- read.fasta(in_f, seqtype="DNA") #in_fで指定したファイルの読み込み
hoge                          #確認してるだけです
hoge[[1]]                    #確認してるだけです
#ファイルに保存(pngファイル)
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを指定
par(mar=c(0, 0, 0, 0))      #下、左、上、右の順で余白(行)を指定
dotPlot(hoge[[1]], hoge[[1]], xlab="", ylab="") #プロット
dev.off()                    #おまじない

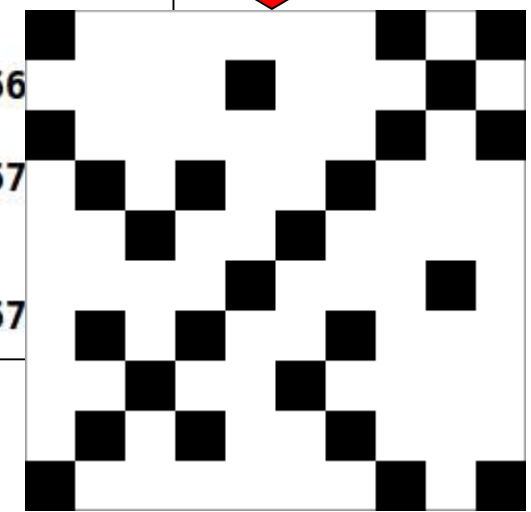
q(save="no")
```



コピー実行後に①lsで確認。②hoge1.pngが確かに作成されている。③中身はこれ。「eog hoge1.png&」で開くことができます(スライド77)

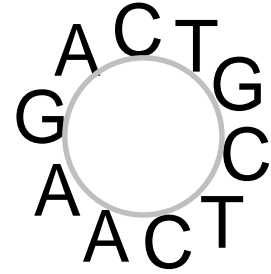
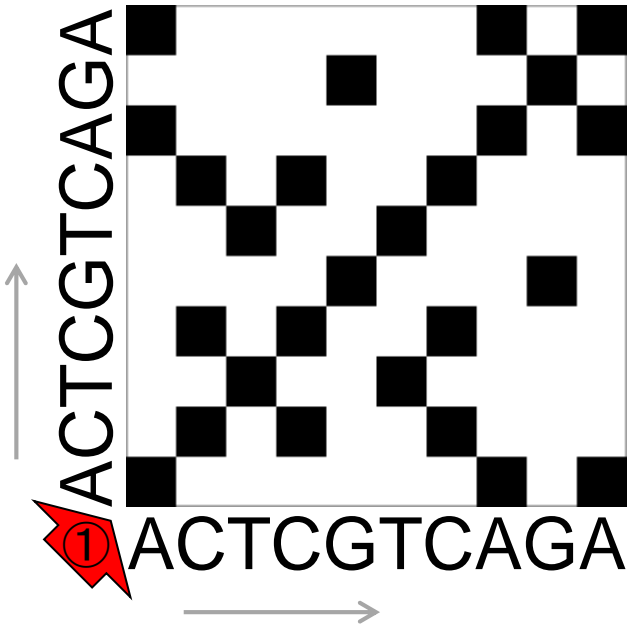
W12-5: dotPlot実行

```
iu@bielinux[~/Desktop/mac_share/result]
[1] "test"
attr(,"Annot")
[1] ">test"
attr(,"class")
[1] "SeqFastadna"
> #ファイルに保存(pngファイル)
> png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2])
#出力ファイルの各種パラメータを指定
> par(mar=c(0, 0, 0, 0)) #下、左、上、右の順で余白
(行)を指定
> dotPlot(hoge[[1]], hoge[[1]], xlab="", ylab="")#プロット
> dev.off() #おまじない
null device
      1
>
> q(save="no")
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l hoge*
-rwxrwxrwx 1 iu iu 837  6月 15 14:57 hoge1.png
-rwxrwxrwx 1 iu iu  17  6月 15 14:30 hoge.fa
iu@bielinux[result] █
```



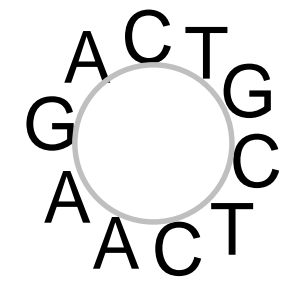
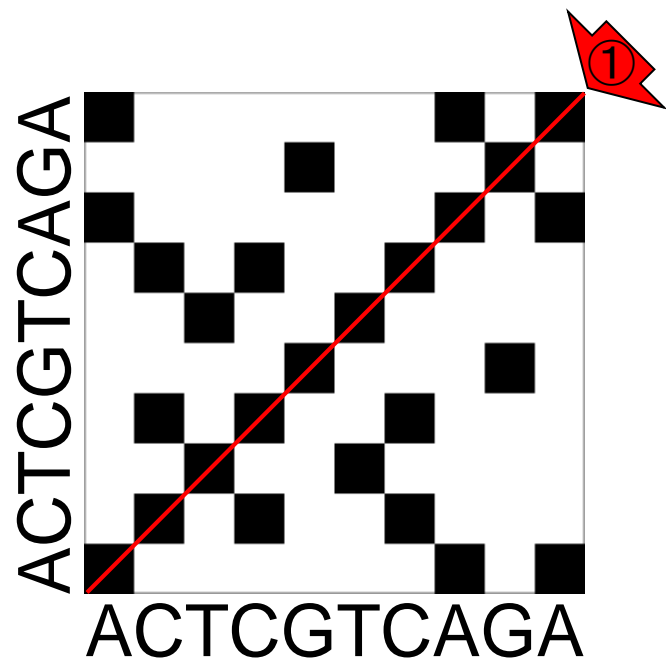
ドットプロットの解説。Rのseqinrパッケージ中のdotPlot関数実行結果ファイルは、①左下を原点として比較する2つの配列を並べている。一致が黒、不一致が白

W12-6: 解説



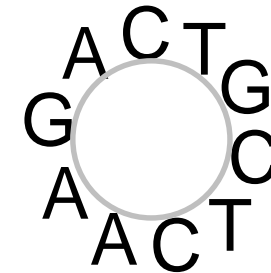
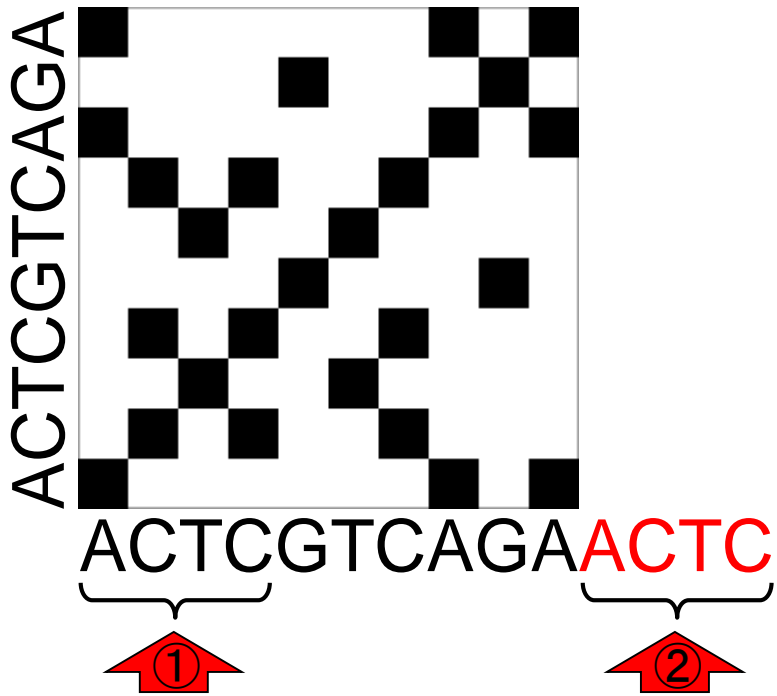
同一の配列を比較するときは、①必ず対角線上の塩基が一致(つまり黒)する

W12-6: 解説



W12-7: 環状コンティグ例

アセンブリ結果として、①最初と②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



W12-7: 環状コンティグ作成

両末端の4塩基がACTCで同じ、計14塩基からなるミニ環状コンティグファイル(hoge2.fa)を作成。最初の10塩基分は、W12-4で作成したhoge.faと同じ

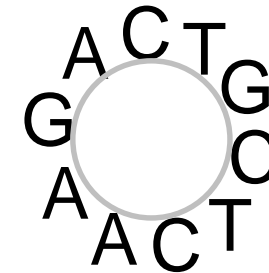
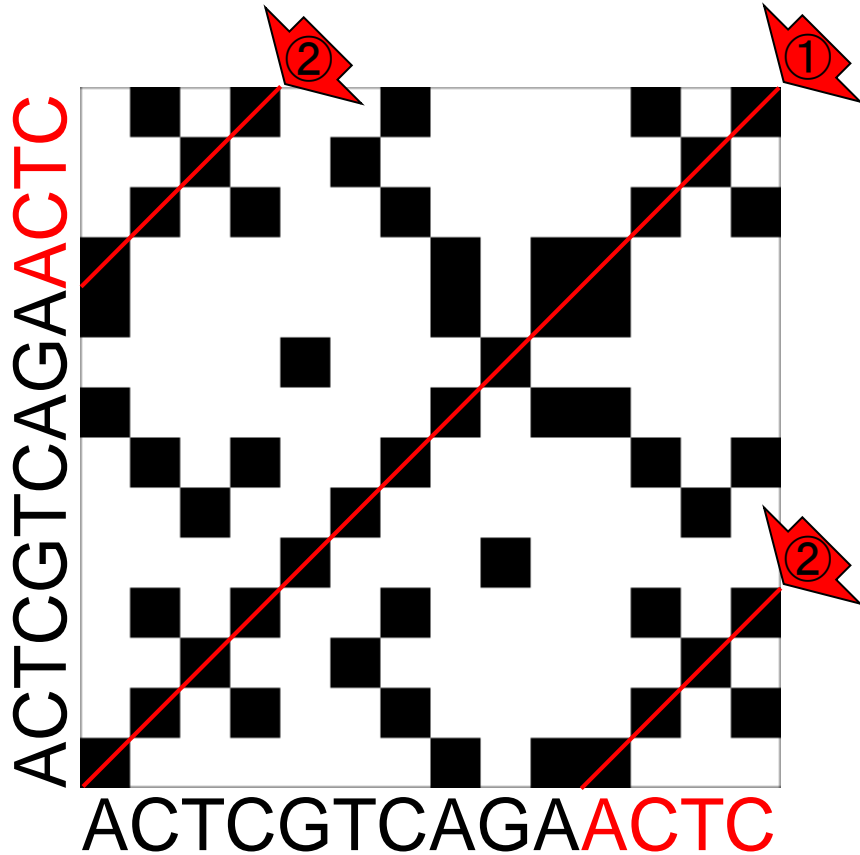
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] echo ">test" > hoge2.fa
iu@bielinux[result] echo "ACTCGTCAGAACTC" >> hoge2.fa
iu@bielinux[result] more hoge2.fa
>test
ACTCGTCAGAACTC
iu@bielinux[result] █
```

[3:09午後]
[3:09午後]
[3:09午後]
[3:09午後]

ACTCGTCAGAACTC

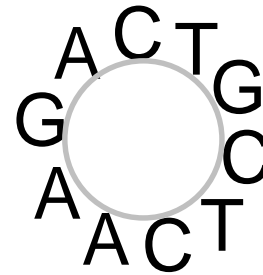
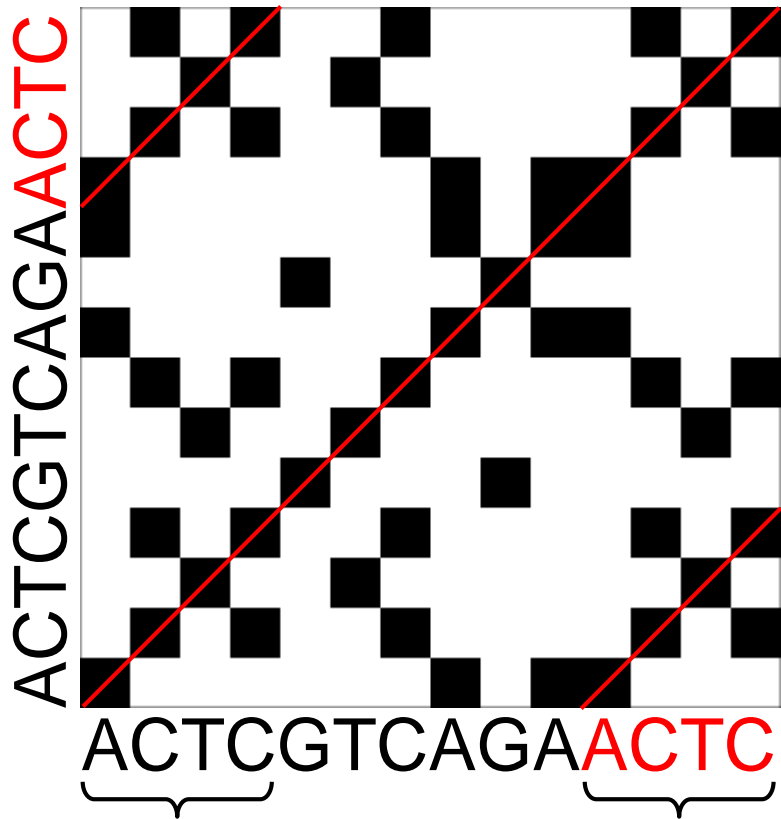
W12-8: 環状の場合

こんな感じに見えます。①対角線上にプロットされるのは同じですが、②対角線と平行に末端部分もプロットされるのが環状の特徴



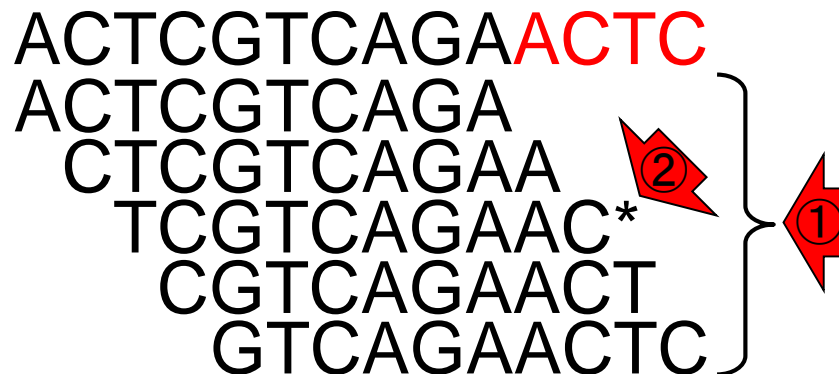
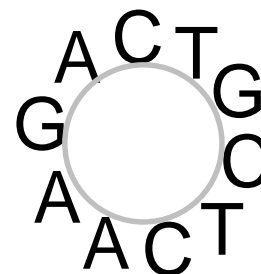
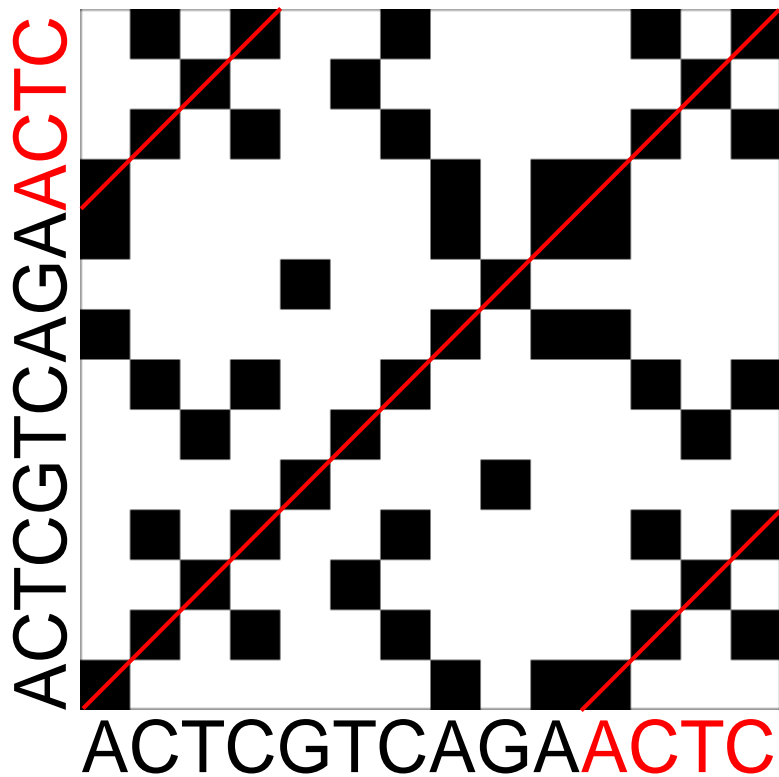
これは、コンティグの両末端が同じ配列であることを意味する。「重複する部分の除去」は、「complete genomeにする操作(finishing)」の一部に相当する

W12-8: 環状の場合



W13-1: 重複除去

重複除去(トリミング)の選択肢は、(この場合は結果的に同じになるが)①5通り存在する。通常(推奨)は、両末端はクオリティが低いので、②中央部分を残して両端をトリムする選択肢(*のついたもの)を採用する



Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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: seqnr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



第7回原稿p106右中

①簡単な例でLinuxコマンドを用いた実際の重複除去の例を示します

ゲ (またはゲノム) の実例を sequence3 で示す。Bio-Linux には dotter¹⁷⁾ というドットプロット用プログラムがプレインストールされているが [W12-1]、これは環状チェックの本番で用いる。まず seqinr¹⁸⁾ という R パッケージ中の dotPlot 関数を用いて、ドットプロットの基本形を示す。環状かどうかのチェックの場合は、同一の配列を並べて比較する。同一配列間のドットプロットの主な特徴は、必ず対角線上に位置する塩基が同じになるという点である [W12-6]。環状の場合は、コンティグの両末端の配列がほぼ同じになる。これは、ドットプロット上で対角線と平行の位置に、重複塩基数分だけの長さの一致部分がハイライトされることで判別できる [W12-8]。

例えば、ランダムな塩基配列 “ACTCGTCAGA” が真の環状ゲノムだと仮定すると、*de novo* アセンブリによって得られるコンティグは “ACTCGTCAGAACTC” のような感じとなる。この場合、灰色の4塩基分の重複がドットプロット上でハイライトされ (図 2a)、重複除去 (塩基のトリミング) が環状化作業に相当する。末端塩基のトリミン

グの選択肢として、4塩基重複の場合は計5通り存在するが、アスタリスク (*) のついた両末端から概ね同数の塩基をトリミングするのが推奨である (図 2b)。その理由は、図 1a で示すようなコンティグ末端部分のクオリティスコア分布を眺めれば納得できるであろう。この単純な例の場合は、結論としてはどの選択肢を採用しても同じ結果になるものの、数千塩基におよぶ実際のプラスミドコンティグの重複部分が完全一致となる可能性はほぼゼロという現実的な問題に当てはめるとよい。

数万～数百万塩基におよぶ配列のドットプロットの作成は、seqinr パッケージの dotPlot 関数では現実的に難しいため [W14-1]、Bio-Linux にプレインストールされている dotter プログラムを用いる [W12-1]。HGAP アセンブリ結果として得られた計4コンティグのうち、2番目に短いコンティグ (45,853 bp; sequence3.fa) が環状かどうかを確認したい場合は、「dotter sequence3.fa sequence3.fa」と打てばよい [W14-2]。ドットプロット全体を眺めると、対角線と平行の直線が配列末端部分に現れていることがわ

①

W13-2: cutコマンド

特定の範囲の切り出しはcutコマンドを利用。① hoge2.faはsingle-FASTA形式。②「tail -n 1」で、最後の1行分のみ取り出している。③パイプで流してcutコマンドを実行し、3-12文字目を表示。これが④環状コンテイングの重複除去後の塩基配列に相当する。これは1行1配列にしているからこそできるワザ！（60塩基程度ごとに改行が入っている通常のFASTA形式だとムリ！）

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] more hoge2.fa
>test
ACTCGTCAGAACTC
iu@bielinux[result] tail -n 1 hoge2.fa
ACTCGTCAGAACTC
iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 3-12
TCGTCAGAAC
iu@bielinux[result] █
```

ACTCGTCAGAACTC
ACTCGTCAGAA
CTCGTCAGAA
TCGTCAGAAC* ←④
CGTCAGAACT
GTCAGAACTC

W13-3: 続cutコマンド

①最初の10塩基分のみ取り出しても、②最後の10塩基分のみ取り出しても、このシンプルな環状ゲノムの場合は③アスタリスクのついたものと結果的には同じ

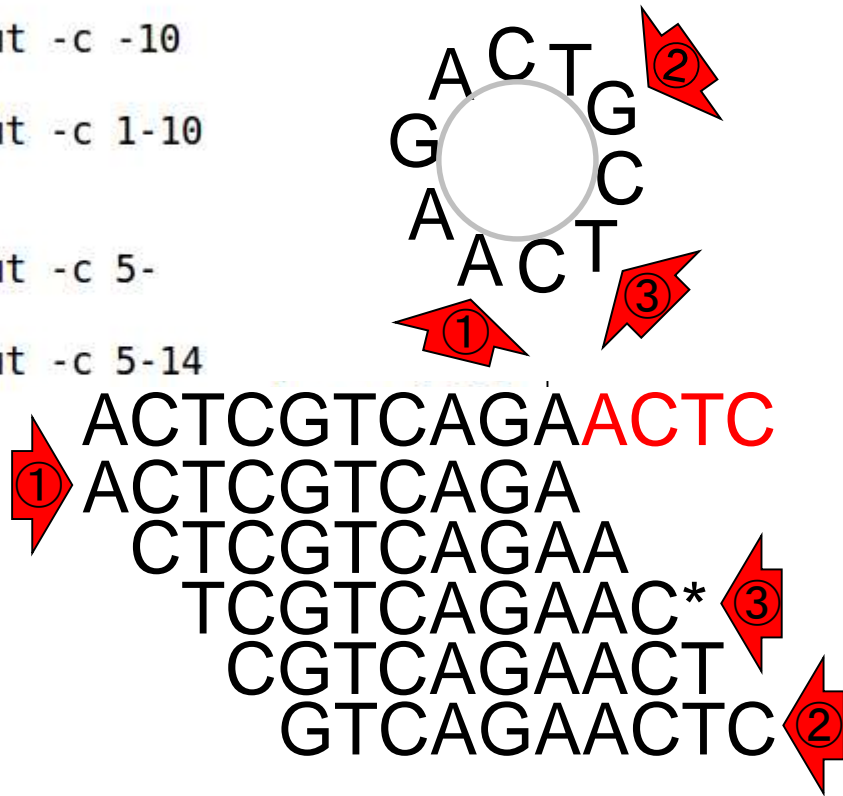
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:13午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] tail -n 1 hoge2.fa [ 4:13午後 ]
ACTCGTCAGAACTC
① iu@bielinux[result] tail -n 1 hoge2.fa | cut -c -10 [ 4:13午後 ]
ACTCGTCAGA
① iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 1-10 [ 4:13午後 ]
ACTCGTCAGA
iu@bielinux[result] [ 4:13午後 ]
② iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 5- [ 4:13午後 ]
GTCAGAACTC
② iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 5-14 [ 4:13午後 ]
GTCAGAACTC
iu@bielinux[result] █
```



理由はスタート地点をどこにするかという違いのみだから。本物の環状染色体の場合は、特定の遺伝子配列が先頭になるように回転させる慣例がある

W13-3: 続cutコマンド

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] tail -n 1 hoge2.fa
ACTCGTCAGAACTC
iu@bielinux[result] tail -n 1 hoge2.fa | cut -c -10
ACTCGTCAGA
iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 1-10
ACTCGTCAGA
iu@bielinux[result]
iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 5-
GTCAGAACTC
iu@bielinux[result] tail -n 1 hoge2.fa | cut -c 5-14
GTCAGAACTC
iu@bielinux[result] █
```



Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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



第7回原稿p106右下

①このあたりの話に移行します。seqinrパッケージのdotPlot関数で行うのが難しいところはすっ飛ばします。気になる人は第7回W14-1を自分でやってみてください

グ（またはゲノム）の実例を sequence3 で示す。Bio-Linux には dotter¹⁷⁾ というドットプロット用プログラムがプレインストールされているが [W12-1]、これは環状チェックの本番で用いる。まず seqinr¹⁸⁾ という R パッケージ中の dotPlot 関数を用いて、ドットプロットの基本形を示す。環状かどうかのチェックの場合は、同一の配列を並べて比較する。同一配列間のドットプロットの主な特徴は、必ず対角線上に位置する塩基が同じになるという点である [W12-6]。環状の場合は、コンティグの両末端の配列がほぼ同じになる。これは、ドットプロット上で対角線と平行の位置に、重複塩基数分だけの長さの一致部分がハイライトされることで判別できる [W12-8]。

例えば、ランダムな塩基配列 “ACTCGTCAGA” が真の環状ゲノムだと仮定すると、*de novo* アセンブリによって得られるコンティグは “ACTCGTCAGAACTC” のような感じとなる。この場合、灰色の 4 塩基分の重複がドットプロット上でハイライトされ (図 2a)、重複除去 (塩基のトリミング) が環状化作業に相当する。末端塩基のトリミン

グの選択が、アスタリスク (*) のついた両末端から概ね同数の塩基をトリミングするのが推奨である (図 2b)。その理由は、図 1a で示すようなコンティグ末端部分のクオリティスコア分布を眺めれば納得できるであろう。この単純な例の場合は、結論としてはどの選択肢を採用しても同じ結果になるものの、数千塩基におよぶ実際のプラスミドコンティグの重複部分が完全一致となる可能性はほぼゼロという現実的な問題に当てはめるとよい。



数万～数百万塩基におよぶ配列のドットプロットの作成は、seqinr パッケージの dotPlot 関数では現実的に難しいため [W14-1]、Bio-Linux にプレインストールされている dotter プログラムを用いる [W12-1]。HGAP アセンブリ結果として得られた計 4 コンティグのうち、2 番目に短いコンティグ (45,853 bp; sequence3.fa) が環状かどうかを確認したい場合は、「dotter sequence3.fa sequence3.fa」と打てばよい [W14-2]。ドットプロット全体を眺めると、対角線と平行の直線が配列末端部分に現れていることがわ

① sequence3.fa 同士のドットプロットを dotter で実行。画面はリターンキーを押して数秒後の状態

W14-2: dotter

```

iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 6:32午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3* [ 6:33午後 ]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 91727 6月 13 17:09 sequence3.fq
-rwxrwxrwx 1 iu iu 20878 6月 14 16:46 sequence3.png
-rwxrwxrwx 1 iu iu 398859 6月 14 16:46 sequence3.txt
iu@bielinux[result] dotter sequence3.fa sequence3.fa [ 6:33午後 ]

Detected sequence types: DNA vs. DNA
Karlin/Altschul statistics for these sequences and score matrix:
K = 0.162
Lambda = 0.177
=> Expected MSP score in a 100x100 matrix = 41.867
Expected residue score in MSP = 1.728
=> Expected MSP length = 24
45853 vs. 45853 residues => 2102.50 million dots. (Takes 2:02 minutes on an SGI MIPS R10000)
    
```



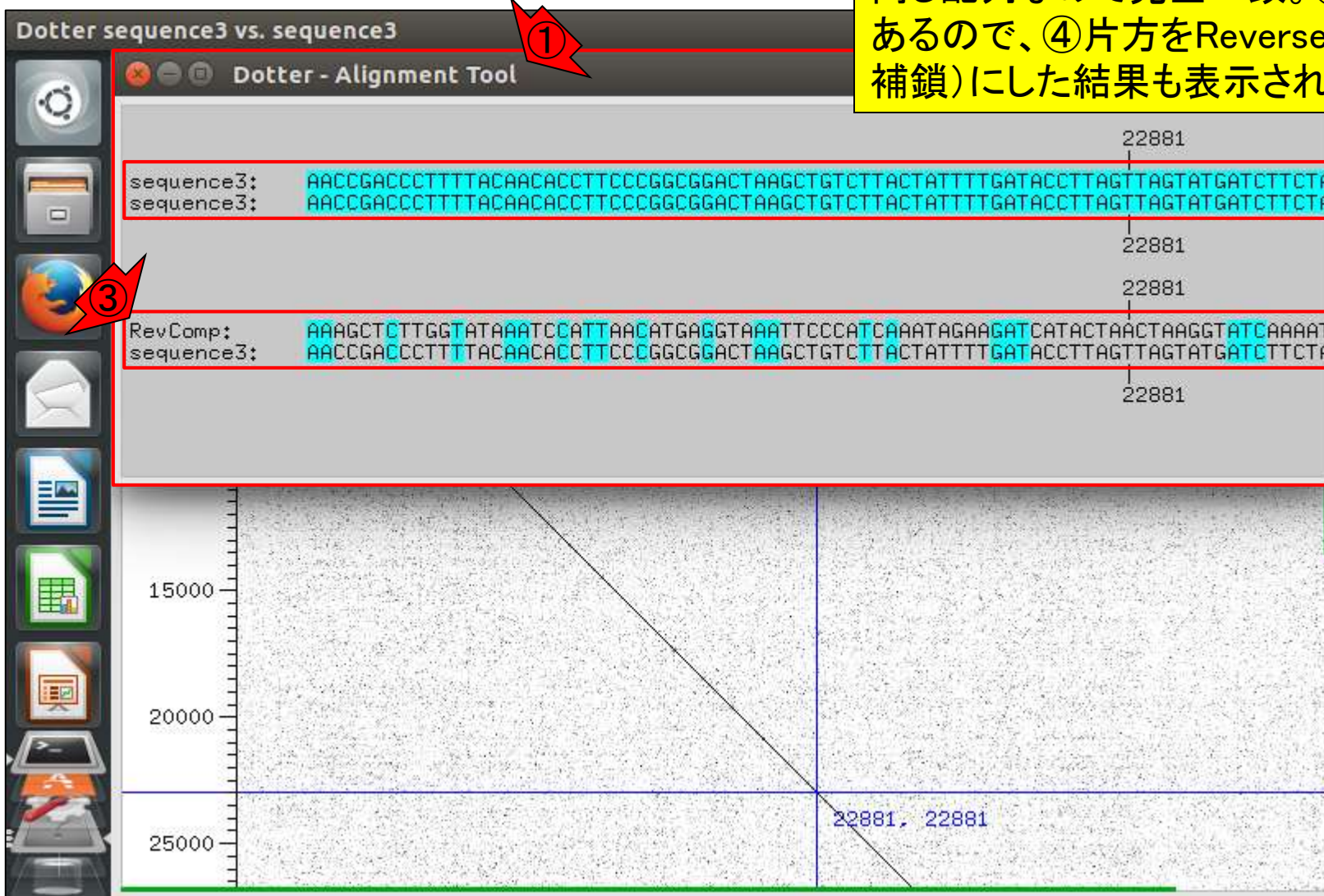
W14-2: dotter

計3つのウィンドウが立ち上がる。①Greyramp Toolはよくわかりませんが、ドットプロットのコントラスト調整用なのだろうと思います。②のあたりをクリックして、アラインメントのウィンドウをアクティブにする

The screenshot displays the Dotter software interface. At the top, the window title is "Dotter sequence3 vs. sequence3". A red arrow labeled "①" points to the "Greyramp Tool" window, which is used for adjusting the contrast of the dot plot. The main window shows a sequence alignment with two DNA sequences: "GGACTAAGCTGTCTTACTATTTTGATACCTTAGTTAGTATGATCTTCTA" and "AGGTAAATTCACATCAAAATAGAAGATCATACTAACTAAGGTATCAAAAT". A red arrow labeled "②" points to the alignment window. Below the alignment is a dot plot with a vertical blue line at position 22881 and a horizontal blue line at position 22881, intersecting at the coordinates "22881, 22881". The y-axis of the dot plot is labeled with 15000, 20000, and 25000.

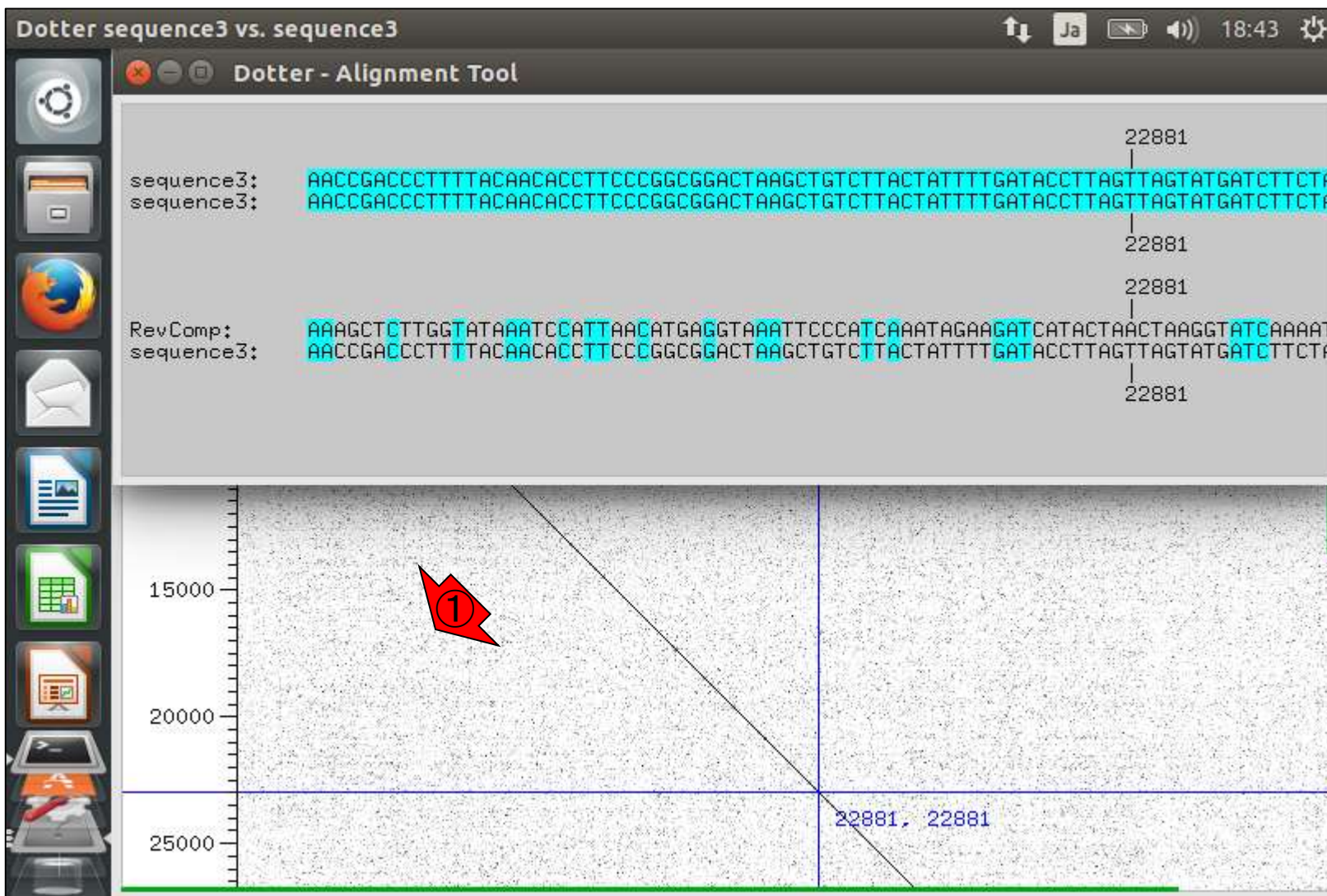
W14-2: dotter

①Alignment Toolは、比較している2つの配列の
アラインメント結果を表示。②今比較しているのは
同じ配列なので完全一致。③RevCompと書いて
あるので、④片方をReverse Complement(逆相
補鎖)にした結果も表示されていることがわかる



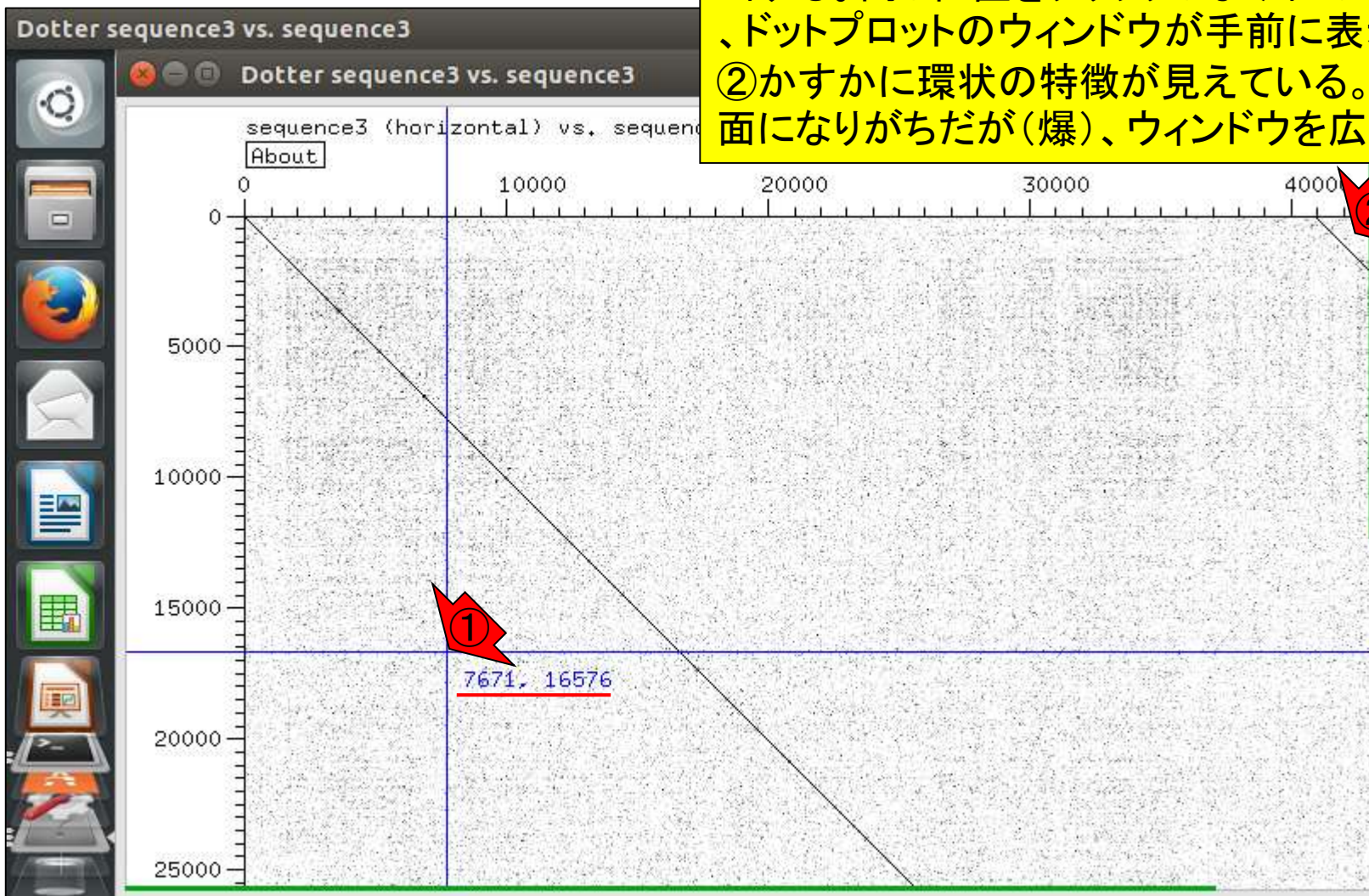
裏側に見えているのが主目的のドットプロット。①このあたりでクリックして、ドットプロットのウィンドウを手前に表示

W14-2: dotter



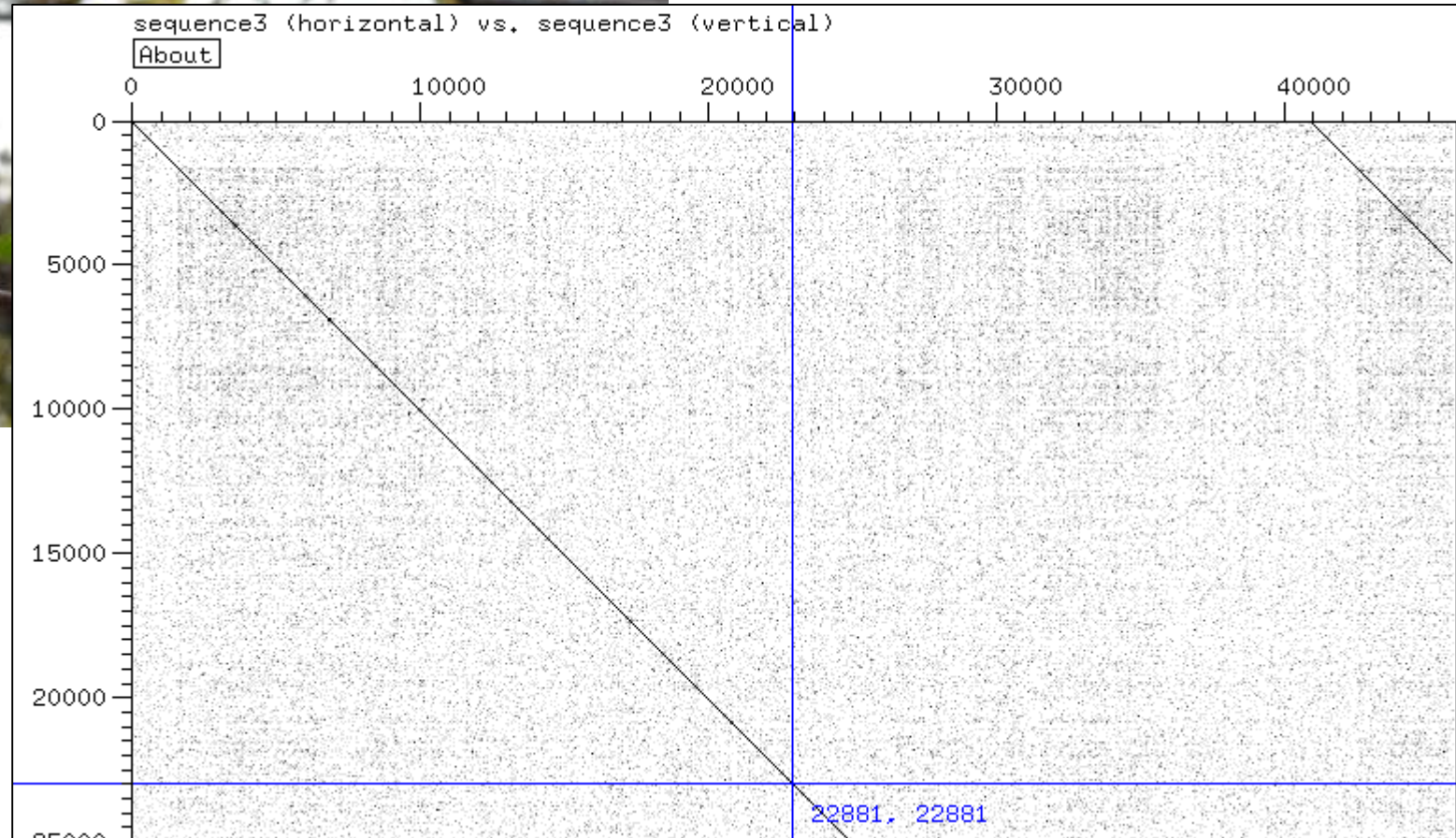
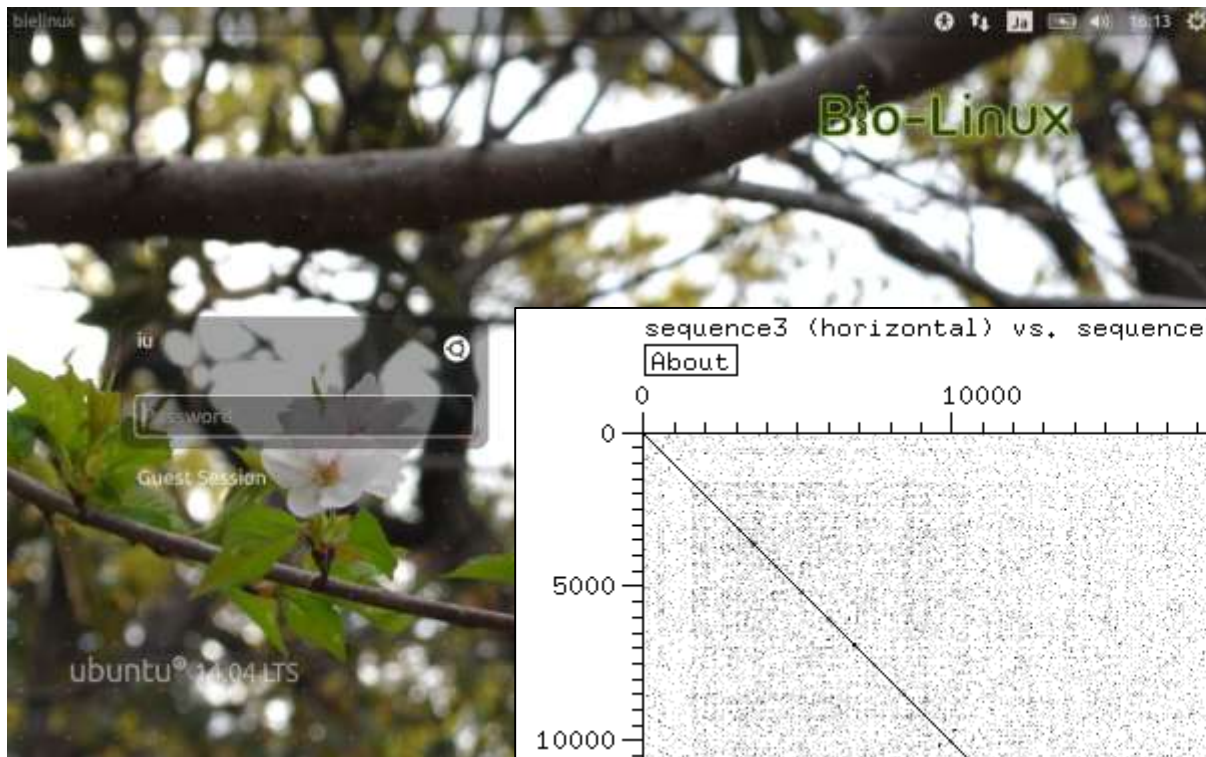
W14-2: dotter

赤下線部分に7671, 16576という数字が見えている。これは今比較している2つの配列の塩基番号(①の座標情報)に相当する。同じ位置をクリックしなければいけないわけではなく、ドットプロットのウィンドウが手前に表示されていればOK。②かすかに環状の特徴が見えている。バグってログイン画面になりがちだが(爆)、ウィンドウを広げて全体像を眺める

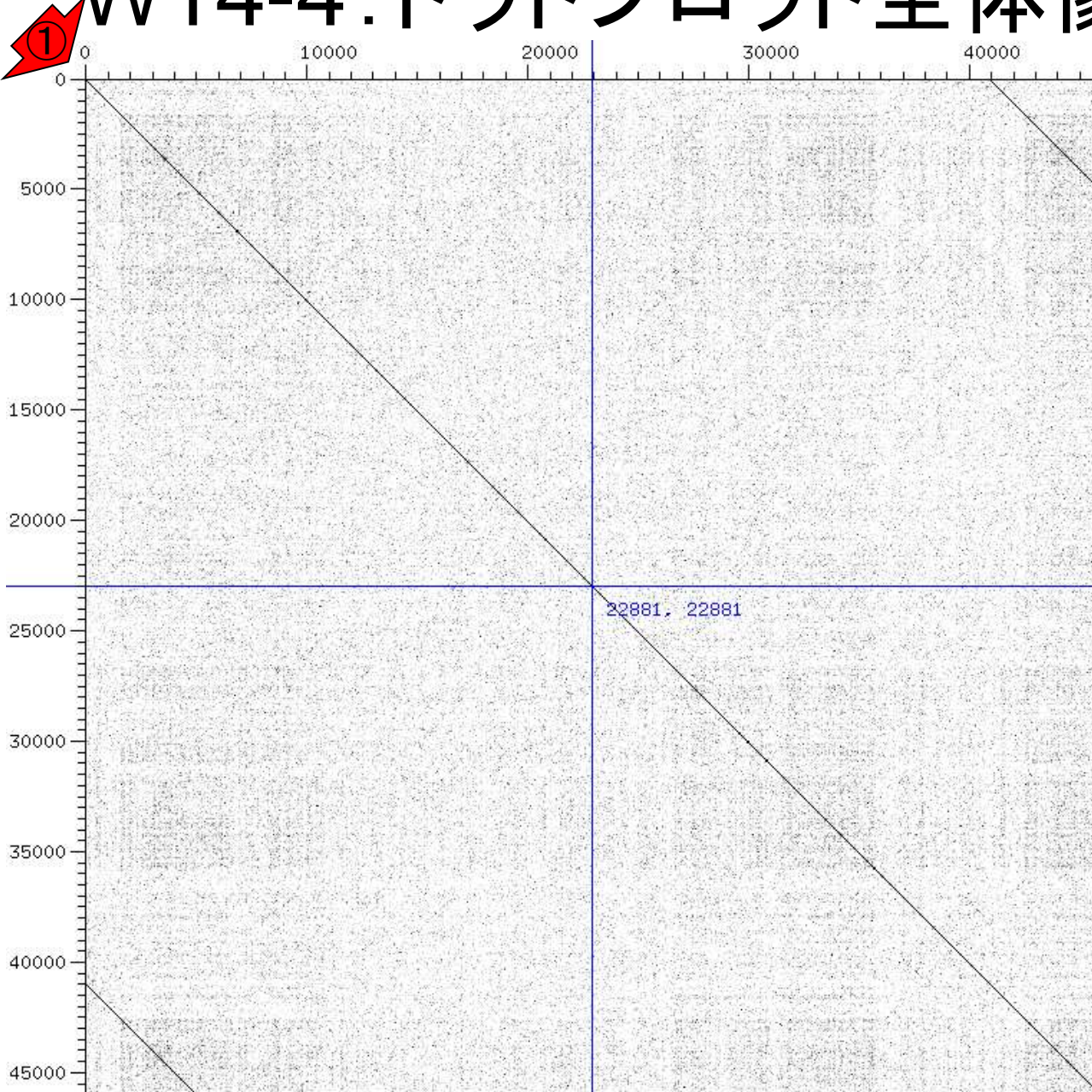


W14-3: バグった例

バグってログイン画面になっても、気を取り直して再挑戦しよう。右下のようなドットプロットが得られるはず



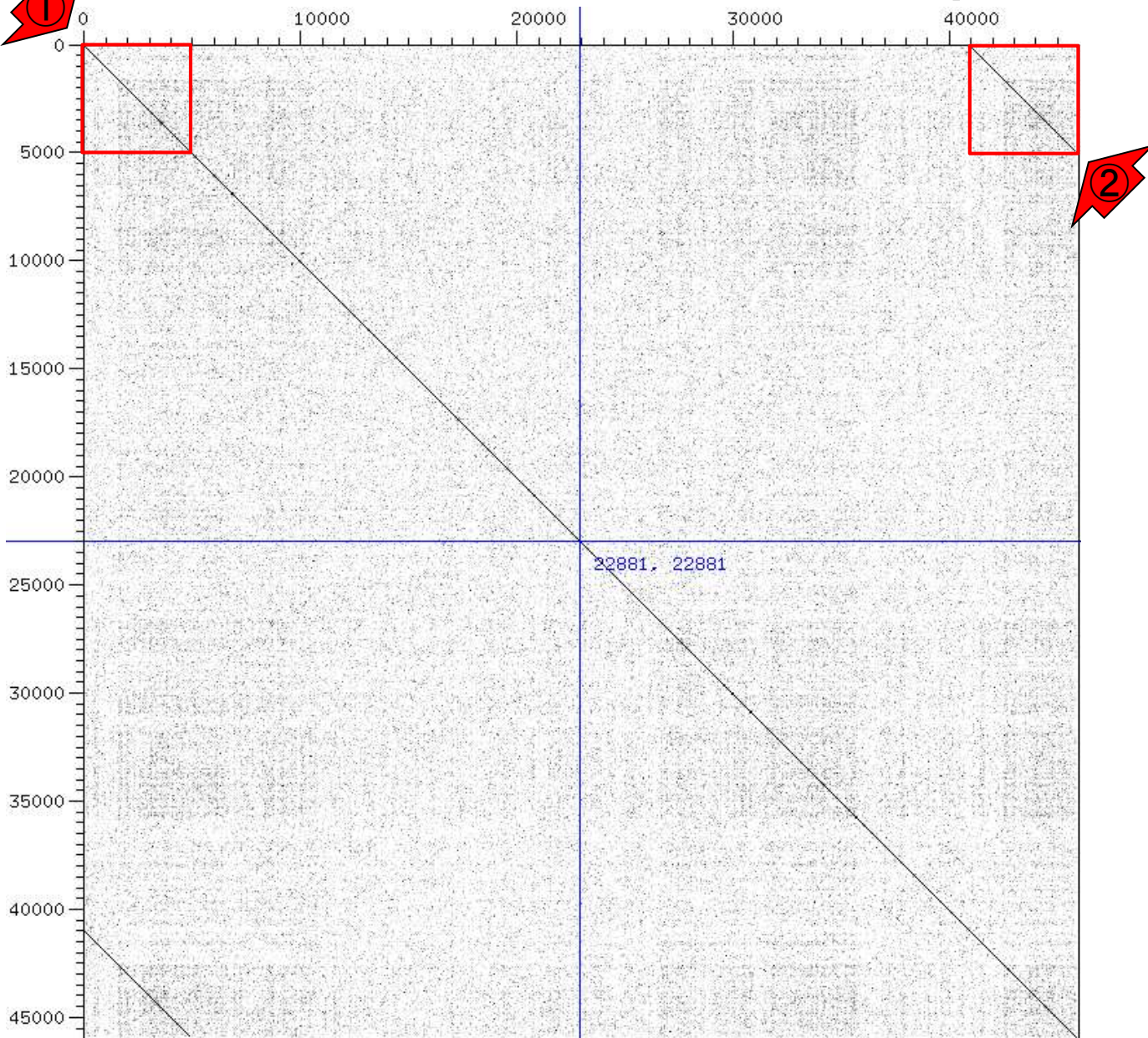
W14-4:ドットプロット全体像



dotterのドットプロットは①左上が原点のようだ。②右上(と左下)にもくっきりと「①の対角線」と並行のラインが見えているので、45,853 bpのsequence3は環状と判断

W14-4: ドットプロット全体像

大まかには、全部で45,853 bpのうち、①最初と②最後の約5,000 bpが重複していると判断



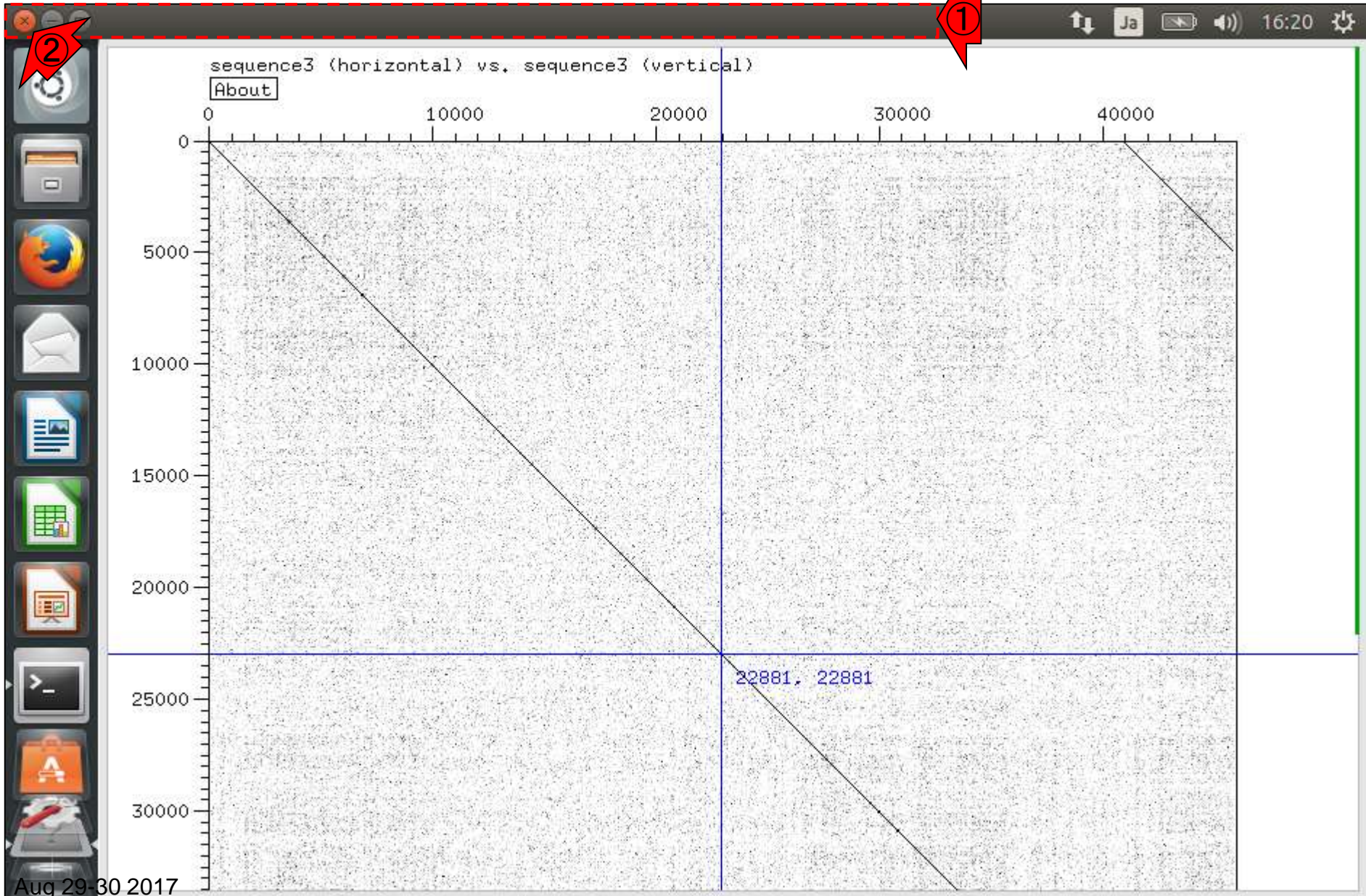
W14-5: dotter終了

Dotter実行結果を終了させるには、基本的に①該当するGUIの左上部分の×を押せばよい

The screenshot displays the 'Dotter - Alignment Tool' window. At the top, it shows 'Dotter sequence3 vs. sequence3'. Below this, two sequence alignments are visible: 'sequence3:' and 'RevComp:'. The alignment text is highlighted in cyan. A red arrow with the number '1' points to the top-left corner of the main window. Another red arrow with the number '1' points to the top-left corner of a smaller 'Greyramp Tool' window that is overlaid on the main window. The 'Greyramp Tool' window has a black display area with a red square and a white triangle, and control buttons for 'Close', 'Swap', and 'Undo'. The background of the main window shows a dot plot with a vertical axis labeled from 15000 to 30000 and a horizontal axis labeled with '22881'. The system tray at the bottom left shows the date 'Aug 29-30 2017'.

①赤い点線の枠内にカーソルを移動させるとメニューバーが見られるようになるので、②×。第3回W6-3

W14-5: dotter終了



W14-6: dotter終了後

```

iu@bielinux[~/Desktop/mac_share/result]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 91727 6月 13 17:09 sequence3.fq
-rwxrwxrwx 1 iu iu 20878 6月 14 16:46 sequence3.png
-rwxrwxrwx 1 iu iu 398859 6月 14 16:46 sequence3.txt
iu@bielinux[result] dotter sequence3.fa sequence3.fa [ 6:38午後 ]

Detected sequence types: DNA vs. DNA
Karlin/Altschul statistics for these sequences and score matrix:
K = 0.162
Lambda = 0.177
=> Expected MSP score in a 100x100 matrix = 41.867
Expected residue score in MSP = 1.728
=> Expected MSP length = 24
45853 vs. 45853 residues => 2102.50 million dots. (Takes 2:02 minutes on an SGI MIPS R10000)

(dotter:17869): Gtk-WARNING **: GtkSpinButton: setting an adjustment with non-zero page size is deprecated

(dotter:17869): Gtk-WARNING **: GtkSpinButton: setting an adjustment with non-zero page size is deprecated
iu@bielinux[result] █ [ 9:14午後 ]
    
```



第7回原稿p106-107

①dotter出力結果のドットプロットを眺めて全体の傾向を把握する。次にBLASTを実行して正確なアラインメント情報を得る。クオリティスコア分布情報(W11-9)とアラインメント情報を用いて、重複除去の本番を行う

—106—

Jpn. J. Lactic Acid Bact.

Vol. 27, No. 2

かる (図 1b)。大まかには、最初と最後の約 5,000 bp が重複しており、この重複除去がアセンブル後の後処理に相当する [W14-4]。もちろんより正確な一致領域を調べる必要はあり、著者らはこの目的のために BLAST¹⁹⁾ を用いる。



と、45,853 bp からなる sequence3.fa の [1, 4884 bp] と [40967, 45853 bp] の範囲が 99% 一致となっていることがわかる [W15-9]。重複除去の選択肢は、以下に示すように概ね 4,884 通り存在する：

配列相同性検索 (BLAST)

・ [4885, 45853 bp] を残す

Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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: seqnr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



第7回原稿p107の左上

①BLASTを(トリッキーな使い方が) sequence3.faの重複領域の正確な情報を得る目的で利用する

配列相同性検索 (BLAST)

BLAST (Basic local alignment search tool) は、局所的なアラインメント (local alignment) を行うプログラムである。問い合わせ (query) 配列と呼ばれる手元の塩基配列またはアミノ酸配列の特徴や性質を把握する目的で、GenBank などの公共塩基配列 DB に対して検索を行ったことがある人は多いだろう。プログラム内部では、query 配列と似たものがあるかどうかを DB 配列に対して検索し、指定した閾値を満たす query 側と DB 側の部分配列のアラインメント結果が出力される。ここでは、query 側および DB 側の配列を sequence3.fa として、(Bio-Linux にプレインストールされている) BLAST を実行する²⁰⁾。同一配列間の比較なので、トップヒット (top hit) は sequence3 の全長配列間で 100% 一致のアラインメントとなる。今詳細に調べたいアラインメント結果は、セカンドヒットの「最初と最後の約 5,000 bp の重複配列」である。



- ・ [4885, 45853 bp] を残す
(最初の 4884 bp、および最後の 0 bp 分をトリム)
- ・ [4884, 45852 bp] を残す
(最初の 4883 bp、および最後の 1 bp 分をトリム)
- ・ [4883, 45851 bp] を残す
(最初の 4882 bp、および最後の 2 bp 分をトリム)
...
- ・ [3, 40968 bp] を残す
(最初の 2 bp、および最後の 4885 bp 分をトリム)
- ・ [2, 40967 bp] を残す
(最初の 1 bp、および最後の 4886 bp 分をトリム)
- ・ [1, 40966 bp] を残す
(最初の 0 bp、および最後の 4887 bp 分をトリム)

概ねとした理由は、[1, 4884 bp] と [40967, 45853 bp] の範囲のアラインメント結果には Gap が含まれており、この Gap の取り扱いに関する不確定要素があるため

W15-1 : makeblastdb

```
iu@bielinux[~/Desktop/mac_share/result]
① iu@bielinux[result] pwd [ 2:17午後 ]
/home/iu/Desktop/mac_share/result
② iu@bielinux[result] makeblastdb -version [ 2:17午後 ]
makeblastdb: 2.2.28+
Package: blast 2.2.28, build Jun 3 2013 11:17:14
iu@bielinux[result] [ 2:17午後 ]
```

①「-h」で大まかな利用法(usage)を確認。
②より詳細な説明は「-help」で出るようだ

W15-1 : makeblastdb

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 2:17午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] makeblastdb -version [ 2:17午後 ]
makeblastdb: 2.2.28+
Package: blast 2.2.28, build Jun 3 2013 11:17:14
iu@bielinux[result] makeblastdb -h [ 2:17午後 ]
USAGE
makeblastdb [-h] [-help] [-in input_file] [-input_type type]
             [-dbtype molecule_type] [-title database_title] [-parse_seqs]
             [-hash_index] [-mask_data mask_data_files] [-gi_mask]
             [-gi_mask_name gi_based_mask_names] [-out database_name]
             [-max_file_sz number_of_bytes] [-taxid TaxID] [-taxid_map TaxID
MapFile]
             [-logfile File_Name] [-version]
DESCRIPTION
Application to create BLAST databases, version 2.2.28+
Use '-help' to print detailed descriptions of command line argument
s
iu@bielinux[result] █ [ 2:19午後 ]
```



①makeblastdb本番。入力はsequence3.fa。塩基配列であることを示すnuclを-dbtypeオプションで指定

W15-1 : makeblastdb

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 2:35午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls sequence3.fa* [ 2:36午後 ]
sequence3.fa
iu@bielinux[result] makeblastdb -in sequence3.fa -dbtype nucl -hash
index
Building a new DB, current time: 06/19/2017 14:36:09
New DB name: sequence3.fa
New DB title: sequence3.fa
Sequence type: Nucleotide
Keep Linkouts: T
Keep MBits: T
Maximum file size: 10000000000B
Adding sequences from FASTA; added 1 sequences in 0.00131798 second
s.
iu@bielinux[result] [ 2:36午後 ]
```

W15-1 : makeblastdb

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 2:35午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls sequence3.fa* [ 2:36午後 ]
sequence3.fa
iu@bielinux[result] makeblastdb -in sequence3.fa -dbtype nucl -hash
_index
Building a new DB, current time: 06/19/2017 14:36:09
New DB name: sequence3.fa
New DB title: sequence3.fa
Sequence type: Nucleotide
Keep Linkouts: T
Keep MBits: T
Maximum file size: 10000000000B
Adding sequences from FASTA; added 1 sequences in 0.00131798 second
s.
iu@bielinux[result] ls sequence3.fa* [ 2:36午後 ]
sequence3.fa sequence3.fa.nhr sequence3.fa.nsd
sequence3.fa.nhd sequence3.fa.nin sequence3.fa.nsi
sequence3.fa.nhi sequence3.fa.nog sequence3.fa.nsq
iu@bielinux[result] [ 2:37午後 ]
```



①blastnを実行。②DB側、③query側はともにsequence3.fa。
④出力ファイル名はsequence3_blast.txt。計算は一瞬

W15-2: blastn

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:08午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls sequence3* [ 3:09午後]
sequence3.fa      sequence3.fa.nin  sequence3.fa.nsq
sequence3.fa.nhd  sequence3.fa.nog  sequence3.fq
sequence3.fa.nhi  sequence3_fa_nsd  sequence3.nng
sequence3.fa.nhr  sequence3_fa_nsi  sequence3.
① iu@bielinux[result] blastn -db sequence3.fa -query sequence3.fa -ou
t sequence3_blast.txt
iu@bielinux[result] [ 3:09午後]
④
```


①lsで確認。確かに②出力ファイル(sequence3_blast.txt)があります

W15-2: blastn

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:08午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls sequence3* [ 3:09午後]
sequence3.fa      sequence3.fa.nin  sequence3.fa.nsq
sequence3.fa.nhd  sequence3.fa.nog  sequence3.fq
sequence3.fa.nhi  sequence3.fa.nsd  sequence3.png
sequence3.fa.nhr  sequence3.fa.nsi  sequence3.txt
iu@bielinux[result] blastn -db sequence3.fa -query sequence3.fa -out
sequence3_blast.txt
iu@bielinux[result] ls sequence3* [ 3:09午後]
sequence3_blast.txt  sequence3.fa.nin  sequence3.fq
sequence3.fa         sequence3.fa.nog  sequence3.png
sequence3.fa.nhd     sequence3.fa.nsd  sequence3.txt
sequence3.fa.nhi     sequence3.fa.nsi
sequence3.fa.nhr     sequence3.fa.nsq
iu@bielinux[result] █ [ 3:14午後]
```



Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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



W15-3: 結果を眺める

blastn実行結果ファイルを眺めるべく、① sequence3_blast.txtの最初の10行分(デフォルトが10行)を表示。②行数は3,852行

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:32午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3_blast.txt [ 3:32午後]
-rwxrwxrwx 1 iu iu 226098 6月 19 15:09 sequence3_blast.txt
iu@bielinux[result] head sequence3_blast.txt [ 3:32午後]
BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb
Miller (2000), "A greedy algorithm for aligning DNA sequences", J
Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
iu@bielinux[result] wc sequence3_blast.txt [ 3:32午後]
3852 8793 226098 sequence3_blast.txt
iu@bielinux[result] █ [ 3:32午後]
```



W15-3: 結果を眺める

BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller (2000), "A greedy algorithm for aligning DNA sequences", J Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
1 sequences; 45,853 total letters



Query= sequence3
Length=45853



Sequences producing significant alignments:

	Score (Bits)	E Value
sequence3	8.468e+04	0.0

> sequence3
Length=45853

Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

```

Query 1      TTTTAGCGGCGGTGTTTGAAGTGCCTGCACTTCTCGAAACACAGTCAATCCTAATTGCCAA 60
          |||
Sbjct 1      TTTTAGCGGCGGTGTTTGAAGTGCCTGCACTTCTCGAAACACAGTCAATCCTAATTGCCAA 60

```

W15-3: 結果を眺める

- ① トップヒットのアラインメント結果の概要。トップヒット(top hit)はsequence3の全長配列間で100%一致のアラインメントなので、
- ② 「Identities = 45853/45853 (100%)」

BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller (2000), "A greedy algorithm for aligning DNA sequences", J Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
1 sequences; 45,853 total letters

Query= sequence3
Length=45853

②

Score = 8.468e+04 bits (45853), Expect = 0.0
 Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
 Strand=Plus/Plus

Sequences producing significant alignments:

	Score (Bits)	E Value
sequence3	8.468e+04	0.0

> sequence3
Length=45853

①

Score = 8.468e+04 bits (45853), Expect = 0.0
 Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
 Strand=Plus/Plus

```

Query 1      TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
          |||
Sbjct 1      TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
  
```


W15-3: 結果を眺める

トップヒットのアラインメント結果は、① dotterによるドットプロット(W14-4)の左上から右下にかけての対角線に相当します

BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller (2000), "A greedy algorithm for aligning DNA sequences", J Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
1 sequences; 45,853 total letters

Query= sequence3
Length=45853

Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

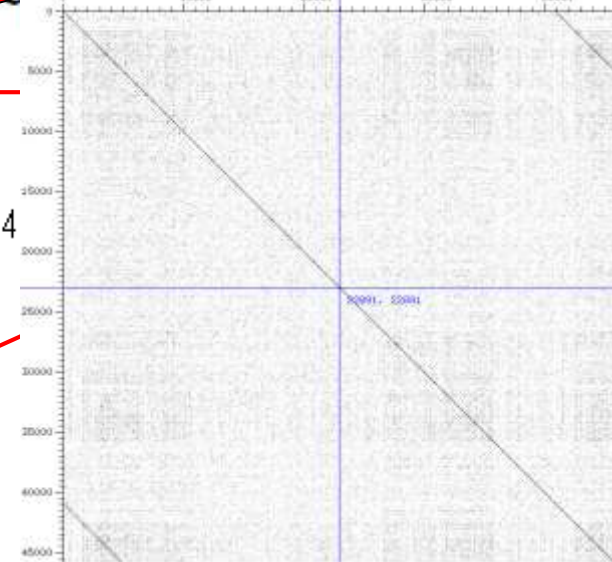
Sequences producing significant alignments:
sequence3

Score (Bits)
8.468e+04

> sequence3
Length=45853

Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

```
Query 1 TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
      |||
Sbjct 1 TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
```



W15-4: ヒット数

①BLAST結果ファイル中のヒット総数を把握したい場合は、"Score ="という文字列を含む行数をgrepで調べればよい

BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller (2000), "A greedy algorithm for aligning DNA sequences", J Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
1 sequences; 45,853 total letters



Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

Query= sequence3
Length=45853

Sequences producing significant alignments:

	Score (Bits)	E Value
sequence3	8.468e+04	0.0

> sequence3
Length=45853

Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

```
Query 1      TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
          |||
Sbjct 1      TTTTAGCGGCGGTGTTTGAAC TGCCGCAC TTCTCGAAACACAGTCAATCCTAATTGCCAA 60
```

- ① "Score =" という文字列を含む行を表示。
- ② その行数は10個。つまりヒット数は10

W15-4: ヒット数

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:06午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3_blast.txt [ 4:06午後]
-rwxrwxrwx 1 iu iu 226098 6月 19 15:09 sequence3_blast.txt
iu@bielinux[result] grep "Score =" sequence3_blast.txt [ 4:06午後]
Score = 8.468e+04 bits (45853), Expect = 0.0
Score = 8844 bits (4789), Expect = 0.0
Score = 8844 bits (4789), Expect = 0.0
Score = 106 bits (57), Expect = 2e-23
Score = 93.5 bits (50), Expect = 2e-19
Score = 93.5 bits (50), Expect = 2e-19
Score = 82.4 bits (44), Expect = 3e-16
Score = 75.0 bits (40), Expect = 6e-14
Score = 75.0 bits (40), Expect = 6e-14
Score = 63.9 bits (34), Expect = 1e-10
iu@bielinux[result] grep -c "Score =" sequence3_blast.txt
10
iu@bielinux[result] [ 4:06午後]
```



W15-5: grep -n

①grep実行時に-nをつけることで検索文字列(この場合"Score =")を含む行番号を表示。例えばセカンドヒットは3,092行目、②サードヒットは3,425行目などというのがすぐにわかる。③BLAST結果ファイル(sequence3_blast.txt)は3,852行だった

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3_blast.txt [ 4:28午後]
-rwxrwxrwx 1 iu iu 226098 6月 19 15:09 sequence3_blast.txt
① iu@bielinux[result] grep -n "Score =" sequence3_blast.txt
27: Score = 8.468e+04 bits (45853), Expect = 0.0
3092: Score = 8844 bits (4789), Expect = 0.0
② 3425: Score = 8844 bits (4789), Expect = 0.0
3758: Score = 106 bits (57), Expect = 2e-23
3771: Score = 93.5 bits (50), Expect = 2e-19
3784: Score = 93.5 bits (50), Expect = 2e-19
3797: Score = 82.4 bits (44), Expect = 3e-16
3806: Score = 75.0 bits (40), Expect = 6e-14
3815: Score = 75.0 bits (40), Expect = 6e-14
3824: Score = 63.9 bits (34), Expect = 1e-10
③ iu@bielinux[result] wc sequence3_blast.txt [ 4:28午後]
3852 8793 226098 sequence3_blast.txt
iu@bielinux[result] █ [ 4:28午後]
```

①grep -Aオプションで一致した行を含め後ろの3行分を表示。
②1位は、一致領域が45,853 bpで100%一致、Gapsも0/45853。W14-4のドットプロットの③対角線の見栄えと一致

W15-6: grep -A

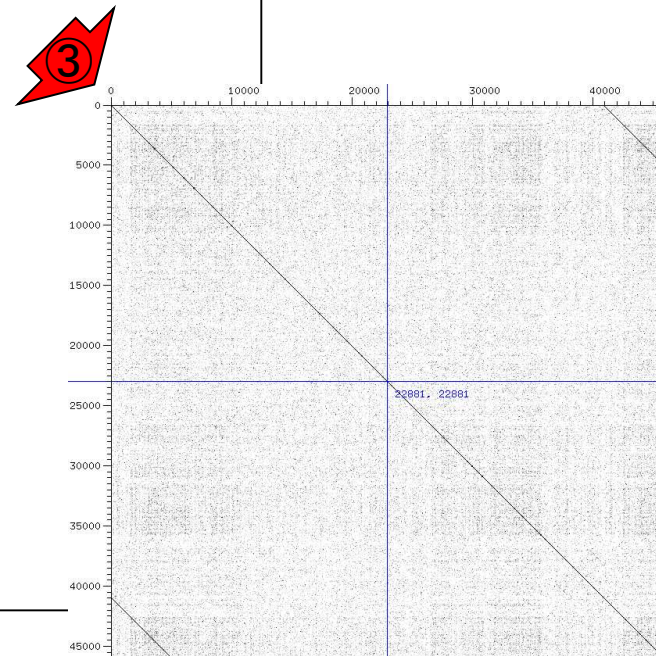
```
File Edit View Search Terminal Help
iu@bielinux[result] grep -A 3 "Score =" sequence3_blast.txt
Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

--
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus

--
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus

--
Score = 106 bits (57), Expect = 2e-23
Identities = 68/73 (93%), Gaps = 2/73 (3%)
Strand=Plus/Minus

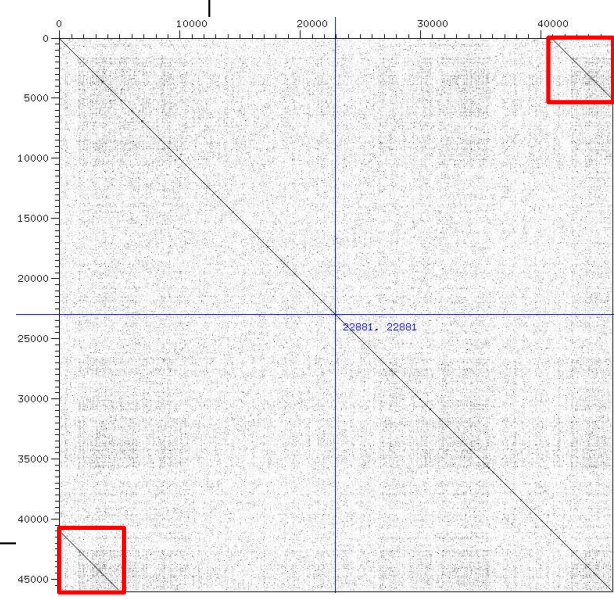
--
```



W15-6: grep -A

同率2位の①と②の2つのヒットは、一致領域が4,901 bp となっており、W14-4のドットプロット上の2つの赤四角の見栄えと一致する。どちらがどちらに相当するかは、W15-5で得られたセカンドヒットの3,092行目以降、サードヒットの3,425行目以降のアラインメントを眺めればわかる

```
iu@bielinux[result] grep -A 3 "Sco
Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus
--
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus
--
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus
--
Score = 106 bits (57), Expect = 2e-23
Identities = 68/73 (93%), Gaps = 2/73 (3%)
Strand=Plus/Minus
--
```



第7回原稿p107の左中

①今はこのあたりの話。これから
②のアラインメント結果を詳細に
眺め、重複除去ポイントを探る

のアラインメント結果が出力される。ここでは、query 側およびDB側の配列を sequence3.fa として、(Bio-Linux にプレインストールされている) BLAST を実行する²⁰⁾。同一配列間の比較なので、トップヒット (top hit) は sequence3 の全長配列間で 100% 一致のアラインメントとなる。今詳細に調べたいアラインメント結果は、セカンドヒットの「最初と最後の約 5,000 bp の重複配列」である。これらの予想は、図 1b のドットプロットを事前^②眺めておけば立てられる。ドットプロットは、コンティグの全体像の理解に役立つだけでなく、BLAST 実行結果の理解の助けにもなる。補足情報的な位置づけではあるが、なるべく併用するといいたいだろう。

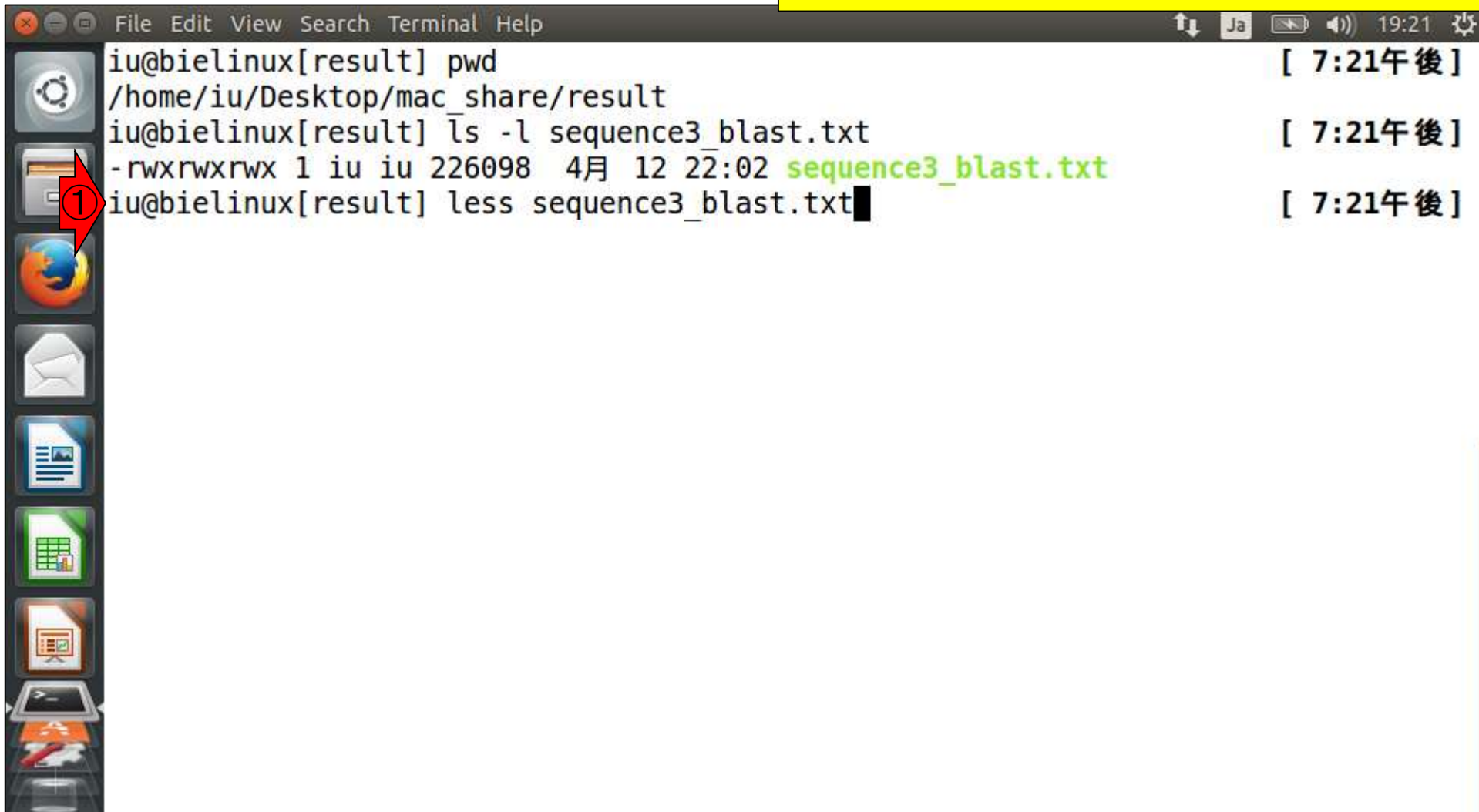
BLAST の実行は、① DB 側配列の BLAST 用 DB への変換、および② query 配列の相同性検索の 2 ステップで完了する [W15]。具体的には、①では makeblastdb コマンドで DB 側配列である sequence3.fa を入力として、BLAST 用 DB (インデックスファイル) を作成する [W15-1]。②では、query 側と DB 側の配列の種類 (塩基配列またはアミノ酸配列) や目的に応じて、以下に示す 5 つのプログラムを使い分ける：

(最初の 1 bp、および最後の 4886 bp 分をトリム)
・ [1, 40966 bp] を残す
(最初の 0 bp、および最後の 4887 bp 分をトリム)

① 概ねとした理由は、[1, 4884 bp] と [40967, 45853 bp] の範囲のアラインメント結果には Gap が含まれており、この Gap の取り扱いに関する不確定要素があるためである。範囲と塩基数の関係や計算法を含めて混乱しがちなところではあるが、始端側の [1, 4884 bp] の塩基数は $(4884 - 1 + 1) = 4884$ bp と計算し、終端側の [40967, 45853 bp] の塩基数は $(45853 - 40967 + 1) = 4887$ bp と計算する。アラインメント結果の始端側と終端側の塩基数が異なるのは、結論としては Gap 数の違いに起因するためであり、気にしなくてよい。実際に我々が行う重複除去は、概ね両側から同数程度の塩基のトリムである。その理由は、アセンブリ結果として得られるコンティグの末端部分のクオリティは、中央部分に比べて低いためである (図 1a; W11-9)。重複塩基数が 4900 bp 程度であることを踏まえ、BLAST アラインメント結果のセカンドまたはサードヒットの 2400-2500 番目付近を眺め、Gap やミス

W15-7: less

lessコマンドでsequence3_blast.txtを開き、Score = で検索。画面の横幅を広めにとっておいたほうがよい。第3回のW14-6-2に文字列検索のやり方あり



The image shows a terminal window with a dark background and a light-colored text area. The window title is "File Edit View Search Terminal Help". The terminal output is as follows:

```
iu@bielinux[result] pwd [ 7:21午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3_blast.txt [ 7:21午後 ]
-rwxrwxrwx 1 iu iu 226098 4月 12 22:02 sequence3_blast.txt
iu@bielinux[result] less sequence3_blast.txt [ 7:21午後 ]
```

A red arrow with the number "1" points to the terminal prompt "iu@bielinux[result] less sequence3_blast.txt". The terminal window has a sidebar on the left with various application icons, including a terminal icon, a file manager icon, a web browser icon, an email icon, a document icon, a spreadsheet icon, a presentation icon, and a laptop icon. The system tray on the right shows the language "Ja", network, volume, and time "19:21".

W15-7: less

```
File Edit View Search Terminal Help
BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb
Miller (2000), "A greedy algorithm for aligning DNA sequences", J
Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
        1 sequences; 45,853 total letters

Query= sequence3

Length=45853

E
Sequences producing significant alignments:
sequence3 blast.txt

Score
(Bits)      V
```


W15-7: less

```

iu@bielinux[~/Desktop/mac_share/result]
BLASTN 2.2.28+

Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb
Miller (2000), "A greedy algorithm for aligning DNA sequences", J
Comput Biol 2000; 7(1-2):203-14.

Database: sequence3.fa
         1 sequences; 45,853 total letters

Query= sequence3
Length=45853

                                     Score
E                                     (Bits)      V
Sequences producing significant alignments:
/Score =
    
```



①トップヒットのものが最初に見える。②全長の45,853 bp全てで完全一致なので、③queryの1-60番目の塩基とDB側(Sbjct; Subjectの意味)の1-60番目の塩基だけで眺めても完全一致となっていることがわかる

W15-7: less

```
File Edit View Search Terminal Help
Score = 8.468e+04 bits (45853), Expect = 0.0
Identities = 45853/45853 (100%), Gaps = 0/45853 (0%)
Strand=Plus/Plus

Query 1 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCAA 60
      |||
Sbjct 1 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCAA 60

Query 61 TTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTAGCCA 120
      |||
Sbjct 61 TTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTAGCCA 120

Query 121 TTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTAT 180
      |||
Sbjct 121 TTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTAT 180

Query 181 GCTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCCGATCACAATAAGT 240
      |||
Sbjct 181 GCTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCCGATCACAATAAGT 240

:
```


「n」と打って、2番目に一致するScore =が先頭行にくるページを表示した結果。①query配列の40,967番目の塩基がDB側配列の1番目の塩基と一致していることを意味する。nとは逆方向に検索していきたい場合はN

W15-8: less

```

iu@bielinux[~/Desktop/mac_share/result]
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus

Query  40967  TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCCA  41026
      |||
Sbjct  1      TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTG-CCA  59

Query  41027  ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTTAGCC  41086
      |||
Sbjct  60     ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTTAGCC  119

Query  41087  ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA  41146
      |||
Sbjct  120    ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA  179

Query  41147  TGCCTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA  41206
      ||
Sbjct  180    TG-CTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA  238

:

```



①query配列の40,967番目の塩基は、
②のあたりのポジションに相当します

W15-8: less

Terminal output (less command):

```
iu@bielinux[~/Desktop/mac_share/result]
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus
Query 40967 TTTTAGCGGCGGTGTTTGAAGTCCGCACT
      |||
Sbjct 1 TTTTAGCGGCGGTGTTTGAAGTCCGCACT

Query 41027 ATTGCAATCAATAGTGACAATTTACCCCAA
      |||
Sbjct 60 ATTGCAATCAATAGTGACAATTTACCCCAA

Query 41087 ATTACGGACACCTCCATCTTTTGATAGCGC
      |||
Sbjct 120 ATTACGGACACCTCCATCTTTTGATAGCGC

Query 41147 TGCCTAATCACAATTACTGCGGCTGAAGCG
      |||
Sbjct 180 TG-CTAATCACAATTACTGCGGCTGAAGCG
```

Dot plot details:

- X-axis: 0 to 40000
- Y-axis: 0 to 45000
- Coordinate: 22881, 22881
- Red arrow ② points to the coordinate 22881 on the x-axis.

②や③のように、DB側(Sbjct)のところどころでGapが見られる。が、④全体で4,901 bpのアラインメントのうち31個だけGapがあった程度なので、実質的に無視でよい

W15-8: less

```
iu@bielinux[~/Desktop/mac_share/result]
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus

Query  40967  TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCCA  41026
      |||
Sbjct  1       TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTG-CCA  59

Query  41027  ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTAGCC  41086
      |||
Sbjct  60     ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTAGCC  119

Query  41087  ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA  41146
      |||
Sbjct  120    ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA  179

Query  41147  TGCCTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA  41206
      |||
Sbjct  180    TG-CTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA  238
```



W15-8: less

「n」と打って、3番目に一致するScore =が先頭行にくるページを表示した結果。①query配列の1番目の塩基がDB側配列の40,967番目の塩基と一致していることを意味する。2番目と3番目はQueryとSbjctが入れ替わっているだけ

```

iu@bielinux[~/Desktop/mac_share/result]
Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus
Query 1 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTG-CCA 59
|||||
Sbjct 40967 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCCA 41026
Query 60 ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTTAGCC 119
|||||
Sbjct 41027 ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTTAGCC 41086
Query 120 ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA 179
|||||
Sbjct 41087 ATTACGGACACCTCCATCTTTTGATAGCGCTAACAAGTGCTACTTCAACAAATCCTTTTA 41146
Query 180 TG-CTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA 238
|||||
Sbjct 41147 TGCCTAATCACAATTACTGCGGCTGAAGCGCCTGGGCAGCAACGGTTCGATCACAATAA 41206
:

```



W15-9: less

上矢印キーを10回押し、10行分だけページ上部に移動した結果画面。セカンドヒットのアラインメント結果の最後のほうを確認するのが目的

```
iu@bielinux[~/Desktop/mac_share/result]
Query 45753 AGAAAGAATTAACGAATTACGCAAAGAAGCCATTGATTACTCTACTAGAAAATCTTATGT 45812
|||||
Sbjct 4784 AGAAAGAATTAACGAATTACGCAAAGAAGCCATTGATTACTCTACTAGAAAATCTTATGT 4843

Query 45813 CACGACCAAATTATTTTTTATCGCCAACATGATTAAGCACA 45853
|||||
Sbjct 4844 CACGACCAAATTATTTTTTATCGCCAACATGATTAAGCACA 4884

Score = 8844 bits (4789), Expect = 0.0
Identities = 4869/4901 (99%), Gaps = 31/4901 (1%)
Strand=Plus/Plus

Query 1 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTG-CCA 59
|||||
Sbjct 40967 TTTTAGCGGCGGTGTTTGAAGTCCGCACTTCTCGAAACACAGTCAATCCTAATTGCCCA 41026
Query 60 ATTGCAATCAATAGTGACAATTTACCCCAAAAACCAGGGGTCTGTCGTTTAATTTTAGCC 119
|||||
```



W16-1: トリム候補領域

上矢印キーをさらに押し続け、(重複塩基数が4900 bp程度なのでその半分の)2400 - 2500番目付近を眺める。具体的には①の赤枠分くらいを眺め、どこにも mismatches や Gap がないことを確認

iu@bielinux[~/Desktop/mac_share/result]

```
Query 2390 GAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTAGAACTGTTGATGAT 2419
|||||
Sbjct 43363 GAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTAGAACTGTTGATGAT 43422

Query 2450 GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAACAGAT 2509
|||||
Sbjct 43423 GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAACAGAT 43482

Query 2510 CCGCGAATAGATCGTCAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA 2569
|||||
Sbjct 43483 CCGCGAATAGATCGTCAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA 43542

Query 2570 GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT 2629
|||||
Sbjct 43543 GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT 43602

Query 2630 TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC 2689
|||||
Sbjct 43603 TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC 43662
:
```



W16-1:トリム候補領域

①のところでトリムすることにする。左端にする理由は、上が2450番目、下が43423番目の塩基だとすぐにわかるから

```
iu@bielinux[~/Desktop/mac_share/result]
Query  2390  GAATTATCAAGCTACGACTGGGGATTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT  2449
      |||
Sbjct  43363  GAATTATCAAGCTACGACTGGGGATTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT  43422
      |||
Query  2450  GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAAACAGAT  2509
      |||
Sbjct  43423  GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAAACAGAT  43482
      |||
Query  2510  CCGCGAATAGATCGTCAAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA  2569
      |||
Sbjct  43483  CCGCGAATAGATCGTCAAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA  43542
      |||
Query  2570  GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT  2629
      |||
Sbjct  43543  GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT  43602
      |||
Query  2630  TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC  2689
      |||
Sbjct  43603  TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC  43662
      |||
:
```



①

W16-2:トリム後の配列

①2450番目の塩基をトリム後の1塩基目にする場合は、[2450, 43422 bp]を残せばよい。こうすることで、トリム後の塩基配列の最初のほうは①の赤枠のようになり、最後のほうは②のようになるはずである。③qで終了

```
iu@bielinux[~/Desktop/mac_share/result]
Query 2390 GAATTATCAAGCTACGACTGGGGATTTCGATA
Sbjct 43363 GAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTAGAACTGTTGATGAT
Query 2450 GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAAACAGAT
Sbjct 43423 GAGCAGGGGTATTACTACTACATCATTCAAATGGGAACGGCACTTGGGAAAAAACAGAT
Query 2510 CCGCGAATAGATCGTCAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA
Sbjct 43483 CCGCGAATAGATCGTCAAATTTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGA
Query 2570 GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT
Sbjct 43543 GAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAACGCAGTATGTAAGCTTAAT
Query 2630 TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC
Sbjct 43603 TCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAAC
```



第7回原稿p107の右中

今は①のあたり。1つ前のスライドで決めた領域のトリム(重複除去)を行う。混乱してきたら、W13-1を復習。このあと②トリムを実行します(W16-3)

sequence3の全長配列間で100%一致のアラインメントとなる。今詳細に調べたいアラインメント結果は、セカンドヒットの「最初と最後の約5,000 bpの重複配列」である。これらの予想は、図1bのドットプロットを事前に眺めておけば立てられる。ドットプロットは、コンティグの全体像の理解に役立つだけでなく、BLAST実行結果の理解の助けにもなる。補足情報的な位置づけではあるが、なるべく併用するといいたいだろう。

BLASTの実行は、①DB側配列のBLAST用DBへの変換、および②query配列の相同性検索の2ステップで完了する[W15]。具体的には、①ではmakeblastdbコマンドでDB側配列であるsequence3.faを入力として、BLAST用DB(インデックスファイル)を作成する[W15-1]。②では、query側とDB側の配列の種類(塩基配列またはアミノ酸配列)や目的に応じて、以下に示す5つのプログラムを使い分ける：

blastn : query側、DB側がともに塩基配列。

blastp : query側、DB側がともにアミノ酸配列。

blastx : query側は塩基配列、DB側はアミノ酸配列。

query配列をアミノ酸配列に翻訳して検索。

概ねとしの範囲のアラインメント結果にはGapが含まれており、このGapの取り扱いに関する不確定要素があるためである。範囲と塩基数の関係や計算法を含めて混乱しがちなところではあるが、始端側の[1, 4884 bp]の塩基数は $(4884 - 1 + 1) = 4884$ bpと計算し、終端側の[40967, 45853 bp]の塩基数は $(45853 - 40967 + 1) = 4887$ bpと計算する。アラインメント結果の始端側と終端側の塩基数が異なるのは、結論としてはGap数の違いに起因するためであり、気にしなくてよい。実際に我々が行う重複除去は、概ね両側から同数程度の塩基のトリムである。その理由は、アセンブリ結果として得られるコンティグの末端部分のクオリティは、中央部分に比べて低いためである(図1a; W11-9)。重複塩基数が4900 bp程度であることを踏まえ、BLASTアラインメント結果のセカンドまたはサードヒットの2400-2500番目付近を眺め、Gapや mismatchesのない領域でトリミング領域を決定する[W16-1]。ここでは、tailとcutコマンドを組み合わせる[2450, 43422 bp]の範囲を抽出し、 $(43422 - 2450 + 1) = 40,973$ bpの長さの環状コンティグ(sequence3_trimmed.fa)として出力した[W16-3]。

W16-3:トリム実行

①まずはトリム後のFASTAファイル(ファイル名: sequence3_trimmed.fa)の description行を作成。W12-7とほぼ同じ

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:07午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*.fa [ 7:05午後 ]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
iu@bielinux[result] echo ">sequence3_trimmed" > sequence3_trimmed.fa [ 7:05午後 ]
iu@bielinux[result] more sequence3_trimmed.fa [ 7:05午後 ]
>sequence3_trimmed
iu@bielinux[result] █ [ 7:05午後 ]
```



W16-3:トリム実行

- ①トリム実行本番。W13-2とほぼ同じ。
- ② sequence3.faの最終行のみ取り出してパイプで流し、
- ③ 2450-43422文字目を抽出した結果を、
- ④ sequence3_trimmed.faに追加書き込み

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:07午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*.fa [ 7:05午後 ]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
iu@bielinux[result] echo ">sequence3_trimmed" > sequence3_trimmed.fa [ 7:05午後 ]
iu@bielinux[result] more sequence3_trimmed.fa [ 7:05午後 ]
>sequence3_trimmed
iu@bielinux[result] tail -n 1 sequence3.fa | cut -c 2450-43422 >> sequence3_trimmed.fa [ 7:05午後 ]
iu@bielinux[result] █
```



①lsで確認。1 bp = 1 byte。ファイルサイズ的に妥当な印象を受ける。②moreでも確認

W16-4: moreで確認

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:07午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*.fa [ 7:05午後 ]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
iu@bielinux[result] echo ">sequence3_trimmed" > sequence3_trimmed.fa [ 7:05午後 ]
iu@bielinux[result] more sequence3_trimmed.fa [ 7:05午後 ]
>sequence3_trimmed
iu@bielinux[result] tail -n 1 sequence3.fa | cut -c 2450-43422 >> sequence3_trimmed.fa
iu@bielinux[result] ls -l sequence3*.fa [ 7:05午後 ]
-rwxrwxrwx 1 iu iu 45865 6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 40993 6月 19 19:06 sequence3_trimmed.fa
iu@bielinux[result] more sequence3_trimmed.fa [ 7:09午後 ]
```



W16-4: moreで確認

①赤枠で示すトリム後の塩基配列の最初のほうは、W16-2と全く同じになっていることからうまくトリムできたと判断

```
iu@bielinux[~/Desktop/mac_share/result]
>sequence3 trimmed
GAGCAGGGGTATTACTACTACATCATTCCAATGGGAACGGCACTTGGGAAAAACAGATCCGCGAATAGATCGTCAAAT
TTAACAGAATATCAAAAAGAAACCCCAATTGATCTAAGAGAAGTAGTGCGCATTATCAAATATTGGAGAAAGGCTCATAAC
GCAGTATGTAAGCTTAATTCTTATGCACTAGAGACCACGGTTCTTGACTTTATAGATACTAATCCAATATATTCCAACCTCT
CGAGAATTTATAGAAAATTTTCTTCTTTATCTTTCAAAGCAGTTTTAGGTTCTGTTCAAGATAGAAAGGGCATTCAAGGA
GATCTTAATTCTTTGGATTATATAGATCGTTTAGAAATTCAGGAAAAAGCGATATACTACCATGACATGATAAAAGAAGCT
AATAATTACGAATCTGAATCCATGAGTGAGCAAGCAATAAAATCTTGGACAAATTTTTTTGGAGACTGATCGATGAAGATA
AGATTCGATATAGCTAAGCAAACACGCCTGAAGCAATTCAATTATTGTCTGCACAAAGACATTTATATTCTCAAGCAAAA
TTTATAGAATTTTTTAGCTTTTTTTCAACATTAGCTCCAATTATTCTTGTACTTTTTTATTAATAAATAGAATTTGTATTCAG
TTCATTACAACCATCATCACTGTGG
Query 2390 GAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT 2449
CAAGAAAAGTTTGACACTCTTATAT
Sbjct 43363 GAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT 3422
AATAAGTTCTCACAAAATAAAAATC
Query 2450 GAGCAGGGGTATTACTACTACATCATTCCAATGGGAACGGCACTTGGGAAAAACAGAT 2509
ATAATTATTGTCAAAGGAGTAATG
Sbjct 43423 GAGCAGGGGTATTACTACTACATCATTCCAATGGGAACGGCACTTGGGAAAAACAGAT 43482
TGCATGTTTCTTCTGCTTATTATCG
ACTACAATTCCAATTGTTATCAAAT
AACCAGATAGATGATATCATTCACTCTATAGATAAAAATAGATATTCAATTTCTGTTAAAGATTTAAGGTCTATTCAAGAT
TTTATATTTGTTAATATTAGATCTGCAACTATTTTAGTTCCAAATTGGCTTTACTTTGTTACCAAGAAAAGCCATGAAGAA
GCAATGAAATCTTCTACTAAAAGTATTGCTGATAATTTAAAAGACATTCGAAACATTAGGATTATGGTGTAATAACATGT
GCTGGATCTTGTTAAAATAATCTTATAAATTGTCGTCTAACCCTTATTGACACTATGTTGAAGGTTATTTATTAATGCT
TCTAGTTCGACCATAACGGACGATGATAGTTAATTGAAAGTTACGGGTTAGCCACTACTAAAAAGATAGCCTTTCTTGGT
GTAAAGCGAACTCCCTGAGTTGTAGTAAATCTAAACTCAGGGAGTTTTGCTATTCTTTCAGCTAAAATAAAACAACAAGTA
--More-- (3%)
```


W16-4: moreで確認

スペースキーをガスガス押して最後まで表示し終わったところ。②最後の塩基配列の赤枠部分もW16-2と全く同じになっていることからうまくトリムできたと判断

iu@bielinux[~/Desktop/mac_share/result]



```

TGACAACACCAGGAATCCCCGGGCAATTGCGTTAATTAAGCAAACACTTCTCAACTGTCAGCACTAATTGTTATTGGT
GAGATGGCACACTGAACGTTGCAATGACAGCCCTAATCCAAGCTGAAGCACATATTCGTATCGGTGTTATTCCCATGGGAA
CGTTAATAATTTTGAACCCGCTACCAGTTACCAACTGACCCCAAGCGGCCATCGAACTAATTTGTAICTAGCCGGCGA
CCAAGCAGTCGGCATGCTAGTTTGAATCAACGCCGGGCAGTCGTGAGTTCAGTACATTCGGTAATTTGGCTGACATTT
CCAATGAAGTTCGACAATCTGAGAAGCAGCGATTTCGGTAAATTAAGCTATCTTTACCGTGCTATTCGCCATATTGGTCACA
ATAAGTCACTGCCAATTCGATATCAATTCAAACACGAAGGAAGCCACACCTTAAAAACATGGTTCTGTTTGATTACAACGA
CAAATCAGTCGGTGGGCACGTCTATAGCGCGTCTGCTCCAGGAAAAATGCATATTAGTCTACTTAACAACATTGGCTGGC
GGCAAGTAATTCATATATTTGGTTCGCACTAACGGGGAACCTTGCAAAACTCCAAGGCCATTACACAGCTAACGGCAACCA
GTGCACGAATTACGAGTGCAACTGGTCAAGCCGTGACGACACGTATTGACGGCGACCCAGCGGTTAAACTGCCAATTGAAT
TGACCTATTTGACAGACCGCTTCGATATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT 2449
TATTAATTGCAAGAATTCATGTTATGATATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT 3422
GACAAAAATCACAAATTAATTTTTTGGATAATCATTGTACGCGGAAAGGC
Query 2450 GAGCAGGGGTATTACTACTACATCATTCCAATGGGAACGGCACTTGGGAAAAAACAGAT 2509
GAAAGAAATTCGATGTACAACGATG
Sbjct 43423 GAGCAGGGGTATTACTACTACATCATTCCAATGGGAACGGCACTTGGGAAAAAACAGAT 43482
AAAGCACGGGTTGAAAACGAGGATAAAAAAGGGGAATGTGTTCATGTCTGTAGATTCTTGGTTTAAAAATATGTTTCAG
AAAACATAAATATAGATAGTAAAAAAGCGCAAGGGCAAGAACTAGTAGAAATTGGTTAACTTCAAATATTAAGATCTTA
GTCAAAAAACGAGGAAAACCTTGAATTATACTCGGACTCAGAATTTGCACTAAAAATGGGATCATTGCTCGAAAGACAC
AGATTAGACCTCTTGACGATGTTGACCAAATGATTATCTTTTCGGCAAAGGGGAGCACCGCTAATTTAGATACGTCTCAAT
GGAATCAGGTGTTTGTAAATGTTCCAGATAGCGCTCCAGAATTAAGAAAAATGGATGGAGAAAATGGGCTTAGTTCTATAA
AAGTCTTGAATTATCTTAAACAGCTATTGAATGGAATATCGCAATATCAATCGGCAGATATTAAGGAATTCAGCAAGCAC
TTAGACTGGAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAACTGTTGATGAT
iu@bielinux[result]

```

[7:11午後]

Contents (第7回後半分)

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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と同じ配列の場合は、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



W17-1: FASTQのトリム

①FASTQファイル(sequence3.fq)の場合は、2行目(塩基配列情報の行)と4行目(クオリティ情報の行)についてのみheadとtailを組合せた操作(W16-3)を行えばよい。②得られるファイルはsequence3_trimmed.fq

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*.fq
-rwxrwxrwx 1 iu iu 91727  6月 13 17:09 sequence3.fq
iu@bielinux[result] wc sequence3.fq
  4      4 91727 sequence3.fq
iu@bielinux[result] head -n 1 sequence3.fq | tail -n 1 > sequence3_trimmed.fq
iu@bielinux[result] head -n 2 sequence3.fq | tail -n 1 | cut -c 2450-43422 >> sequence3_trimmed.fq
iu@bielinux[result] head -n 3 sequence3.fq | tail -n 1 >> sequence3_trimmed.fq
iu@bielinux[result] head -n 4 sequence3.fq | tail -n 1 | cut -c 2450-43422 >> sequence3_trimmed.fq
iu@bielinux[result] ls -l sequence3*.fq
-rwxrwxrwx 1 iu iu 91727  6月 13 17:09 sequence3.fq
-rwxrwxrwx 1 iu iu 81967  6月 20  2017 sequence3_trimmed.fq
iu@bielinux[result] █
```

[12:04午後]

[12:04午後]

[12:04午後]

[12:04午後]



FASTQファイル(sequence3_trimmed.fq)を入力として、W11-9(スライド80)と同じようなクオリティスコア分布を作成

W17-2:クオリティ分布

- W17-2:クオリティスコア分布 (スライド163-165)

トリム後のFASTQ形式ファイル([sequence3_trimmed.fq](#))を入力として、図1aおよびW11-9と同じようなクオリティスコア分布を作成。出力ファイルは、[sequence3_trimmed.png](#)と[sequence3_trimmed.txt](#)。

```
cd ~/Desktop/mac_share/result

R -q
in_f <- "sequence3_trimmed.fq"           #入力ファイル名を指定してin_fに格納
out_f1 <- "sequence3_trimmed.png"       #出力ファイル名を指定してout_f1に格納
out_f2 <- "sequence3_trimmed.txt"       #出力ファイル名を指定してout_f2に格納
param_fig <- c(700, 350)                #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
#必要なパッケージをロード
library(ShortRead)                      #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
#本番(PHREDスコアに変換)
out <- as(quality(fastq), "matrix")      #ASCIIコードのquality scoreをPHRED scoreに変換し、データ種
colnames(out) <- 1:ncol(out)             #列名を付与
rownames(out) <- as.character(id(fastq)) #行名を付与
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを
par(mar=c(4, 4, 0, 0))                  #下、左、上、右の順で余白(行)を指定
plot(x=1:ncol(out), y=out, pch=20, cex=0.5, #プロット
     type="p", xlab="position", ylab="PHRED score") #プロット
dev.off()                                 #おまじない
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out), as.vector(out)) #保存したい情報をtmpに格納
write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの中身を指定
q(save="no")
pwd
ls -l sequence3_trimmed*
```

W17-2:クオリティ分布

①コピー実行後に得られるファイル。
「eog sequence3_trimmed.png &」
で開けます

```

iu@bielinux[~/Desktop/mac_share/result]
> #ファイルに保存(pngファイル)
> png(out_f1, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
の各種パラメータを指定
> par(mar=c(4, 4, 0, 0)) #下、左、上、右の順で余白(行)を指定
> plot(x=1:ncol(out), y=out, pch=20, cex=0.5,#プロット
+ type="p", xlab="position", ylab="PHRED score")#プロット
> dev.off() #おまじない
null device
1
> #ファイルに保存(テキストファイル)
> tmp <- cbind(colnames(out), as.vector(out))#保存したい情報をtmpに格納
> write.table(tmp, out_f2, sep="\t", append=F, quote=F, row.names=F, col.names=F)
#tmpの中身を指定したファイル名で保存
> q(save="no")
iu@bielinux[result] pwd [12:22午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3_trimmed* [12:22午後]
-rwxrwxrwx 1 iu iu 40993 6月 19 19:06 sequence3_trimmed.fa
-rwxrwxrwx 1 iu iu 81967 6月 20 12:04 sequence3_trimmed.fq
-rwxrwxrwx 1 iu iu 19113 6月 20 2017 sequence3_trimmed.png
-rwxrwxrwx 1 iu iu 357641 6月 20 2017 sequence3_trimmed.txt
iu@bielinux[result] [12:22午後]
  
```



W17-2:クオリティ分布

①pngファイルを眺めているところ。W11-9で見られていた両側の低クオリティ領域がうまくトリムされていることがわかる。図3aと同じ

- W17-2:クオリティスコア分布 (スライド163-165)

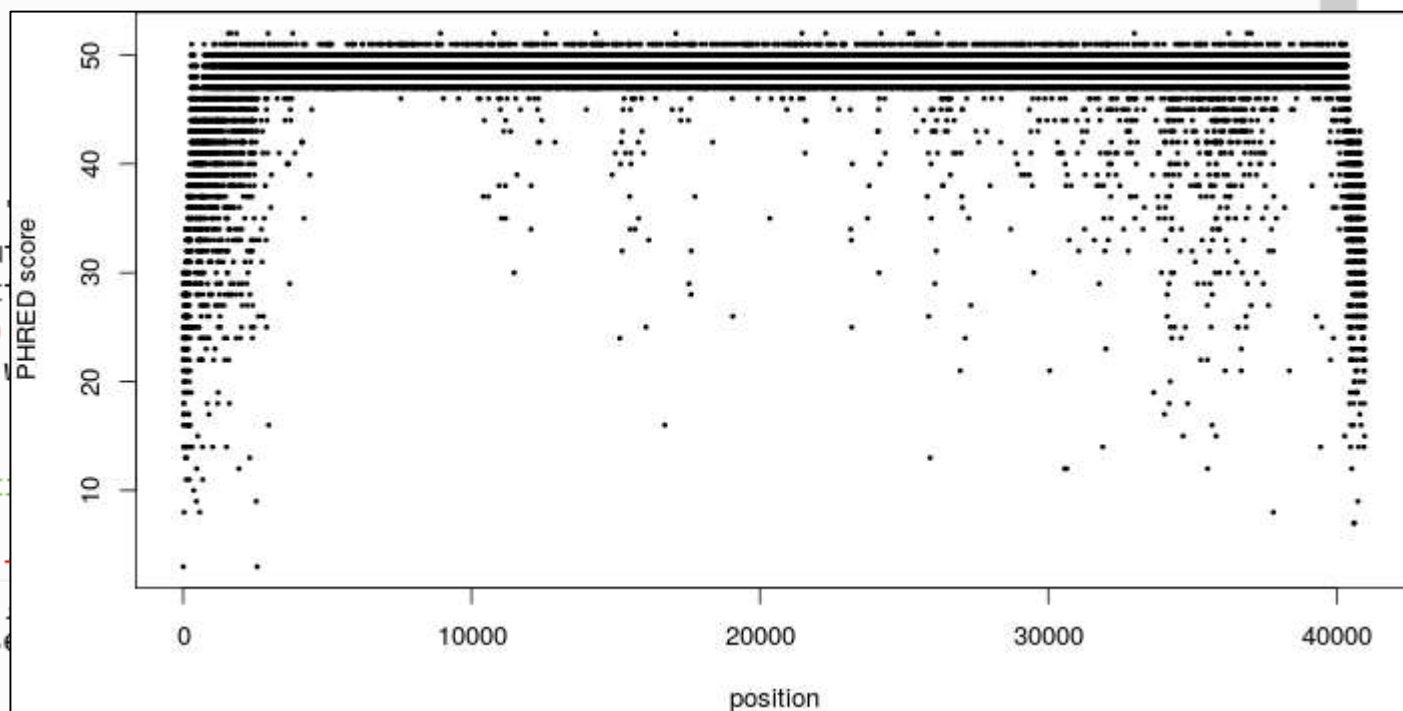
トリム後のFASTQ形式ファイル(sequence3_trimmed.fq)を入力として、図1aおよびW11-9と同じようなクオリティスコア分布を作成。出力ファイルは、sequence3_trimmed.pngとsequence3_trimmed.txt。

```
cd ~/Desktop/mac_share/result
```

```
R -q
in_f <- "sequence3_trimmed.fq"
out_f1 <- "sequence3_trimmed.png"
out_f2 <- "sequence3_trimmed.txt"
param_fig <- c(700, 350)
#必要なパッケージをロード
library(ShortRead)
#入力ファイルの読み込み
fastq <- readFastq(in_f)
#本番(PHREDスコアに変換)
out <- as(quality(fastq),
colnames(out) <- 1:ncol(out)
rownames(out) <- as.character(1:50)
#ファイルに保存(pngファイル)
png(out_f1, pointsize=13,
par(mar=c(4, 4, 0, 0))
plot(x=1:ncol(out), y=out,
type="p", xlab="position",
dev.off()
#ファイルに保存(テキストファイル)
tmp <- cbind(colnames(out),
write.table(tmp, out_f2, sep="\t",
q(save="no")
pwd
ls -l sequence3_trimmed*
```



#入力ファイル名を指定してin_fに格納
 #出力ファイル名を指定してout_f1に格納
 #出力ファイル名を指定してout_f2に格納
 #ファイル出力時の横幅と縦幅を指定(単位はピクセル)



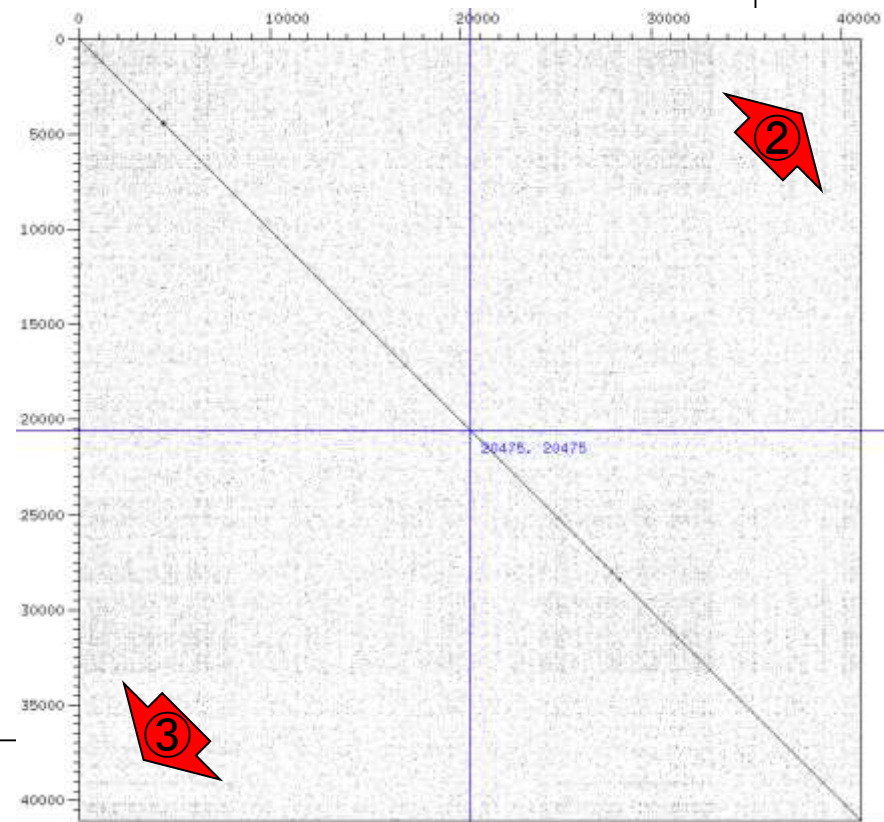
W17-3: ドットプロット

① sequence3.fa 同士のドットプロットを dotter で実行。トリム前 (W14-4) と違って、配列末端部分の一致領域 (② や ③ 付近の対角線のプロット) がなくなっていることがわかる。うまく重複領域をトリムできていることを意味する

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence3*.fa
-rwxrwxrwx 1 iu iu 45865  6月 13 11:33 sequence3.fa
-rwxrwxrwx 1 iu iu 40993  6月 19 19:06 sequence3_trimmed.fa
iu@bielinux[result] dotter sequence3_trimmed.fa sequence3_trimmed.fa
```

[12:30午後]

[12:30午後]



ここまでのまとめ

- イン트로ダクション(主に予習事項の確認)
- W10: multi-FASTAファイルの分割
- W11: FASTQファイルの分割とクオリティスコア分布
- 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

PacBioデータの *de novo* アセンブリ結果ファイルをもとに、3番目に長い配列(約4.6万塩基のsequence3.fa)の重複配列除去を行った。時間はかかるが、2番目に長い配列(約8.7万塩基のsequence2.fa)も同じ要領で可能なので、トライしてみてください

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p188の左中

今は①のあたり。②でも書いているが、本番ではなく予備的なアノテーション

ゲノムアノテーション

ゲノムのアノテーション (genome annotation) とは、ゲノム上のどの位置にどのような遺伝子がコードされているかなどを調べ、注釈づけを行う作業である。バクテリアの自動アノテーションに関しては、MiGAP⁹⁾ や RAST¹⁰⁻¹²⁾ などの様々なウェブサービスが提供されており、手軽に実行可能である。今回は、乳酸菌に特化したアノテーションパイプライン DFAST (DDBJ Fast Annotation and Submission Tool)¹³⁾ を用いる。本ウェブサービスは、連載第5回¹⁴⁾ でも紹介したアノテーションパイプライン Prokka¹⁵⁾ をベースとして、乳酸菌 (主に *Lactobacillus* 属および *Pediococcus* 属) 用に整備された参照データベースを組み合わせたものである。また、アノテーションだけでなく、DDBJ¹⁶⁾ への塩基配列登録支援を行うこともできる (もちろん登録は任意) のが特徴である。典型的なゲノムサイズ (数 MB 程度) の乳酸菌であれば、5分ほどで結果が返される。ここでは、アセンブリ結果の検証の一環として、アセンブリ結果ファイル (LH_hgap.fa) を入力として DFAST を実行する [W4]。位置づけとしては、予備的なアノテーションである。

第7回までで予想していた通り、この2つの配列が (環状の) プラスミドであることを支持している。

sequence1 (2,289,497 bp) については、最初の 30,000 塩基あたりまでに prophage protein などファージ関連遺伝子が見られる [W4-6]。また、sequence4 (11,372 bp) については、4,912 番目から 5,699 番目の領域に transposase がコードされている。この領域については後で議論するが、アノテーション結果の概観レベルでは見逃してもよい。後述する BLAST¹⁷⁾ の実行結果と合わせて総合的に判断する視点が重要である。

BLAST の実行と可視化

第7回では、sequence3 同士を例として配列内の両末端部分の重複をドットプロットで大まかに調べ (第7回 W14-2)、BLAST で重複領域の詳細なアラインメントを行った。ドットプロットについては、sequence3 よりも2倍程度長い sequence2 同士についても dotter¹⁸⁾ を実行可能であり、両末端部分の重複が見いだせる [W5-1]。sequence4 同士のドットプロットは、[1, 500 bp] と [750, 1350 bp] 付近の領域が似ているものの、環状を示唆す

W4-1: DFAST

DFASTは乳酸菌に特化したゲノムアノテーションを行ってくれるウェブツール。入力はmulti-FASTAファイル。出力はGFF形式ファイルなどのアノテーション結果。DFASTのトップ画面上で、①をクリック。となっはいるが、**細菌(しかも乳酸菌のみ)に特化しているため、エアーハンズオン(やったふり)でアノテーションツール利用法の概要を把握する程度で十分です**

DFAST and DAGA are integrated genome annotation tools and resources. DFAST is an annotation platform for bacterial genomes. Its core annotation process is based on PROKKA and curated reference database tailored to specific organisms. It also generates DDBJ-compliant submission files for Mass Submission System (MSS) at DDBJ. DAGA is a genome archive that stores bacterial genomes obtained from DDBJ/ENA/GenBank and Sequence Read Archive (SRA). All the genomes deposited in DAGA are consistently annotated using DFAST. This website provides genome resources for *Lactic Acid Bacteria*.

DFAST DDBJ Fast Annotation and Submission Tool

Upload your Genome, Annotate, and Submit to DDBJ.

You can see the Example of the annotation from [here](#).

W4-1: DFAST

The screenshot shows the DFAST web interface in a browser window. The address bar displays `https://dfast.nig.ac.jp/analysis/annotation/`. The page title is "DFAST: DDBJ Fast ...". The main content area has a header "DFAST" and a description: "DFAST is an annotation platform for bacterial genomes. Its core annotation process is based on PROKKA and curated reference database tailored to specific organisms. It also generates DDBJ-compliant submission files for Mass Submission System (MSS) at DDBJ."

Below the description, there are two input fields: "Query File (Fasta format)" and "Job Title". The "Query File" field has a "参照..." button next to it, which is highlighted with a red arrow and a circled "1". The "Job Title" field contains the text "(optional)".

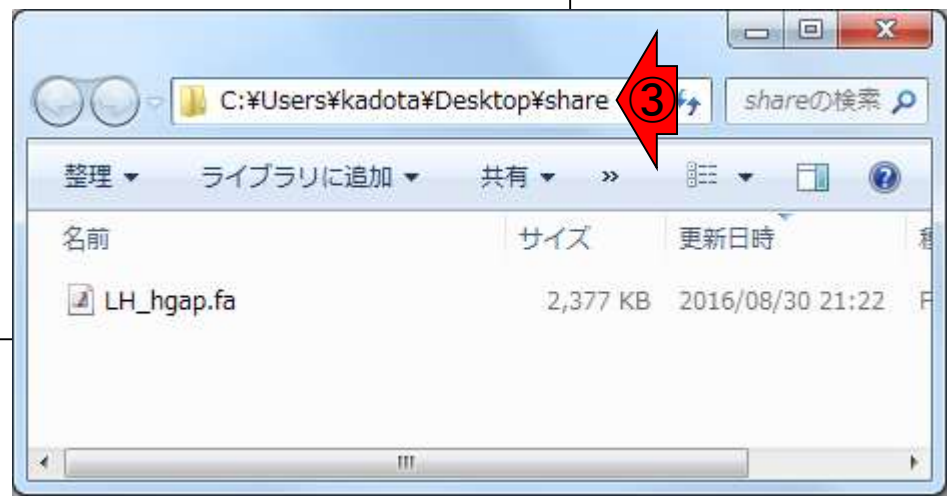
Below these fields is the "Mail Address" section, which contains a text input field with the placeholder text "E-mail notification will be sent to this address when the job is completed. (opti".

At the bottom, there is a section titled "Specify metadata and parameters." with the text: "These data other than minimum contig length can be altered later. Reference Databases for genera other than Lactobacillus and Pediococcus are not fully supported."

W4-2: LH_hgap.fa

①アノテーションしたいmulti-FASTAファイル(LH_hgap.fa)を共有フォルダにコピー。②共有フォルダ内の状況はヒトそれぞれだが、最低限LH_hgap.faが見えていけばよい。(私のWindows環境では)ホストOS上の絶対パスとして③のところにLH_hgap.faが見えるので、これをアップロードします

```
iu@bielinux[~/Desktop/result]
iu@bielinux[result] pwd
/home/iu/Desktop/result
iu@bielinux[result] ls
corrected.fastq  LH_hgap.fa          sequence3.fa  test3.fa
JSLAB8_1.sh     polished_assembly.fasta  sequence4.fa  test4.fa
JSLAB8_2.sh     polished_assembly.fastq  smrtpipe.log
JSLAB8_3.sh     sequence1.fa          test1.fa
JSLAB8_4.sh     sequence2.fa          test2.fa
① iu@bielinux[result] cp LH_hgap.fa ../mac_share [ 9:21午後]
iu@bielinux[result] ls -l ../mac share [ 9:22午後]
total 2377
-rwxrwxrwx 1 iu iu 2433662 8月 30 21:22 LH_hgap.fa ②
iu@bielinux[result] █ [ 9:22午後]
```



①参照ボタン、②に移動して、③目的のファイル(LH_hgap.fa)を指定して、④開く

W4-3: アップロードと実行

The image shows a web browser window displaying the DFAST website and a Windows file explorer window. The browser window shows the DFAST interface with a 'Query File (Fasta format)' field and a '参照...' button (1). The file explorer window shows the file 'LH_hgap.fa' (3) in the 'C:\Users\kadota\Desktop\share' directory (2). The '開く(O)' button (4) is highlighted in the file explorer.

DFAST

DFAST is an annotation platform for bacterial genomes. Its core annotation process is based on PROKKA and curated reference data for various organisms. It also generates DDBJ-compliant submission files and manages the Submission System (MSS) at DDBJ.

Query File (Fasta format)

Job Title

(optional)

Mail Address

E-mail notification will be sent to this address when the job is completed.

Specify metadata and parameters.

These data other than minimum contig length can be altered. Databases for genera other than Lactobacillus and Pedicoccus are not supported.

アップロードするファイルの選択

C:\Users\kadota\Desktop\share

名前	更新日時	種類	サイズ
LH_hgap.fa	2016/08/30 21:22	FA ファイル	2,377 KB

ファイル名(N): LH_hgap.fa

開く(O) キャンセル

W4-3: アップロードと実行

①ジョブタイトル(hogegeee)をテキストにつけて、②計算が終わったらメールでお知らせしてくれるので入力して、③ページ下部に移動

DFAST is an annotation platform for bacterial genomes. Its core annotation process is based on PROKKA and curated reference database tailored to specific organisms. It also generates DDBJ-compliant submission files for Mass Submission System (MSS) at DDBJ.

Query File (Fasta format)
C:\User: 参照...

Job Title
hogegeee

Mail Address
kadota@iu.a.u-tokyo.ac.jp

Specify metadata and parameters.
These data other than minimum contig length can be altered later. Reference Databases for genera other than Lactobacillus and Pediococcus are not fully supported.

Genus Lactob
Species sp
Strain unkown

W4-3: アップロードと実行

①いろいろオプション指定できるがとりあえずここは無視して、②Run。③は設定値(デフォルトは200 bp)以下の短い配列を除くため。それ以外の赤枠の属・種名等のオプションはアノテーション結果に影響を与えることはなく、後で変更することも可能であるため、規定値のままでOK

Specify metadata and parameters.
These data other than minimum contig length can be altered later. Reference Databases for genera other than Lactobacillus and Pediococcus are not fully supported.

Genus	Species	Strain
Lactob	sp. ex) plantarum, delbrueckii subsp. bulgaricus	unkown

Locus Tag Prefix: LOCUS
Minimum Contig Length: 200

▼ Show Perform Genome Assessment (optional)

Run

計算が始まったようだ。とりあえず計算終了メールが来るまで思考停止

W4-3: アップロードと実行

DFAST

Remember the [current URL](#) to access this page. The result will be deleted 30 days after your last visit.
Delete this job now. => This procedure cannot be undone.

Title : hogegeee

JobID : f341d803-072b-48db-a363-1de4c3a686a5

Status : RUNNING

[2016-08-31 15:18:55.909400] Job submitted.
[2016-08-31 15:18:55.932959] Job started.

計算終了メールが届く。このときは、約5分で計算が終了。①にアクセス

W4-4: 計算終了

2016/08/31 (水) 15:22

DFAST Report <dfast@nig.ac.jp>

[DFAST] Your requested job completed.

宛先 kadota@iu.a.u-tokyo.ac.jp

Your requested job has completed.

Job Title : hogegeee

Job ID : f341d803-072b-48db-a363-1de4c3a686a5

Submitted at :2016-08-31 15:18:55.909400.

Please visit the link below to check the result.

<https://dfast.nig.ac.jp/analysis/annotation/f341d803-072b-48db-a363-1de4c3a686a5>

①

DFAST <dfast@nig.ac.jp>

W4-5: 結果を眺める

DFAST

Remember the [current URL](#) to access this page. The result will be deleted 30 days after your last visit.
Delete this job now. => This procedure cannot be undone.

Title : hogegee

JobID : f341d803-072b-48db-a363-1de4c3a686a5

Status : COMPLETE

```
[2016-08-31 15:18:55.909400] Job submitted.  
[2016-08-31 15:18:55.932959] Job started.  
[2016-08-31 15:21:52.789970] Job completed.
```

Result Features DDBJ Submission Log

Genome Statistics

Total Length (bp)	2,433,614
No. of Sequences	4

Download Files

Genbank Flat
File : [annotation.gbk](#)
GFF3-formated



W4-5: 結果を眺める

これがDFASTによるアノテーション結果の全体像。①このあたりの数値は、DDBJ Pipeline上でHGAPを実行したときの結果(第7回W9-2)と全く同じで妥当

The screenshot shows a web browser window with the URL <https://dfast.nig.ac.jp/analysis/annotation/f341d803>. The page has tabs for 'Result', 'Features', 'DDBJ Submission', and 'Log'. The 'Result' tab is active, displaying 'Genome Statistics' and 'Download Files'.

Genome Statistics	
Total Length (bp)	2,433,614
No. of Sequences	4
GC Content (%)	38.2%
N50	2,289,497
Gap Ratio (%)	0.0%
No. of CDSs	2,389
No. of rRNA	12
No. of tRNA	56
No. of CRISPRS	1
Coding Ratio (%)	86.7%

The 'Download Files' section lists the following files:

- Genbank Flat File : [annotation.gbk](#)
- GFF3-formated File : [annotation.gff](#)
- Genome Fasta File : [genome.fna](#)
- Protein Fasta File : [protein.faa](#)
- CDS Fasta File : [cds.fna](#)
- RNA Fasta File : [rna.fna](#)
- Feature Table : [features.tsv](#)
- Genome Statistics :

A red box highlights the 'Total Length (bp)' and 'No. of Sequences' rows, with a red arrow pointing to the value '4' in the 'No. of Sequences' row, labeled with a circled '1'.

W4-5: 結果を眺める

①アノテーションファイルの一般的な形式であるGFFファイルをダウンロードしてエクセルなどで眺めてもよいが、ここでは②Featuresタブをクリックして、ウェブ上でアノテーション結果を眺める

The screenshot shows the DFAST web interface for a genome analysis job. The browser address bar shows the URL: <https://dfast.nig.ac.jp/analysis/annotation/f341d803>. The page has four tabs: "Result", "Features", "DDBJ Submission", and "Log". The "Features" tab is selected and highlighted with a red arrow labeled "2".

On the left side, under the heading "Genome Statistics", there is a table with the following data:

Total Length (bp)	2,433,614
No. of Sequences	4
GC Content (%)	38.2%
N50	2,289,497
Gap Ratio (%)	0.0%
No. of CDSs	2,389
No. of rRNA	12
No. of tRNA	56
No. of CRISPRS	1
Coding Ratio (%)	86.7%

On the right side, under the heading "Download Files", there is a list of files for download:

- Genbank Flat File : [annotation.gbk](#)
- GFF3-formated File : [annotation.gff](#) (highlighted with a red arrow labeled "1")
- Genome Fasta File : [genome.fna](#)
- Protein Fasta File : [protein.faa](#)
- CDS Fasta File : [cds.fna](#)
- RNA Fasta File : [ma.fna](#)
- Feature Table : [features.tsv](#)
- Genome Statistics :

W4-5: 結果を眺める

こんな感じになります。①デフォルトはページあたり25エントリーになっているので、②の部分が25行分ずつ表示されます

The screenshot shows the DFAST database interface. At the top, there are tabs for 'Result', 'Features', 'DDBJ Submission', and 'Log'. Below the tabs, the 'Annotated Features' section is active. A 'Show' dropdown menu is set to '25 entries'. A search box is present to the right. Below the search box is a table with columns: LocusTag, Seq. ID, Location, Feature Type, Product, Gene, Nucleotide, Translation, and Edit. The first seven rows of the table are highlighted with a red box. A red arrow points to the 'Show 25 entries' dropdown, and another red arrow points to the first row of the table.

	LocusTag	Seq. ID	Location	Feature Type	Product	Gene	Nucleotide	Translation	Edit
1	LOCUS_00001	sequence1	151..384	CDS	hypothetical protein		View	View	Edit
2	LOCUS_00002	sequence1	350..886	CDS	hypothetical protein		View	View	Edit
3	LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein		View	View	Edit
4	LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein		View	View	Edit
5	LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein		View	View	Edit
6	LOCUS_00006	sequence1	2355..2732	CDS	prophage protein		View	View	Edit
7	LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein		View	View	Edit

W4-5: 結果を眺める

Result Features DDBJ Submission Log

Annotated Features

Show **25** entries

Search:

No.	Locu Tag	Seq. ID	Location	Feature Type	Product	Gene	Nucleotide	Translation	Edit
1	LOCUS_00001	sequence1	151..384	CDS	hypothetical protein		View	View	Edit
2	LOCUS_00002	sequence1	350..886	CDS	hypothetical protein		View	View	Edit
3	LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein		View	View	Edit
4	LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein		View	View	Edit
5	LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein		View	View	Edit
6	LOCUS_00006	sequence1	2355..2732	CDS	prophage protein		View	View	Edit
7	LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein		View	View	Edit

W4-5: 結果を眺める

こんな感じになります。横幅が広がるが細かいことは気にしない。①のあたりを見ることで、入力ファイル(LH_hgap.fa)の配列および座標順にアノテーションされた結果が表示されていることがわかる

Browser address: <https://dfast.nig.ac.jp/analysis/annotation/f341d803-072b-48db-a363-1de4c3a686a5/f>

dfast.nig.ac.jp の待機中

Result Features DDBJ Submission Log

Annotated Features

Show entries Search:

No.	Locus Tag	Seq. ID	Position	Feature Type	Product	Gene	Nucleotide	Transcript
1	LOCUS_00001	sequence1	151..384	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
2	LOCUS_00002	sequence1	350..886	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
3	LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
4	LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
5	LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
6	LOCUS_00006	sequence1	2355..2732	CDS	prophage protein		<input type="button" value="View"/>	<input type="button" value="View"/>
7	LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
8	LOCUS_00008	sequence1	2930..3031	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
9	LOCUS_00009	sequence1	3067..3534	CDS	holin		<input type="button" value="View"/>	<input type="button" value="View"/>
10	LOCUS_00010	sequence1	3547..4578	CDS	1,4-beta-N-acetylmuramidase		<input type="button" value="View"/>	<input type="button" value="View"/>

W4-6: sequence1概観

Result Features DDBJ Submission Log

Annotated Features

Show entries Search:

No.	Locus Tag	Seq. ID	Position	Feature Type	Product	Gene	Nucleotide	Translation
1	LOCUS_00001	sequence1	151..384	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
2	LOCUS_00002	sequence1	350..886	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
3	LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
4	LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
5	LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
6	LOCUS_00006	sequence1	2355..2732	CDS	prophage protein		<input type="button" value="View"/>	<input type="button" value="View"/>
7	LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
8	LOCUS_00008	sequence1	2930..3031	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
9	LOCUS_00009	sequence1	3067..3534	CDS	holin		<input type="button" value="View"/>	<input type="button" value="View"/>
10	LOCUS_00010	sequence1	3547..4578	CDS	1,4-beta-N-acetylmuramidase		<input type="button" value="View"/>	<input type="button" value="View"/>

W4-6: sequence1概観

①sequence1の左端(最初の3031 bpまで)は「hypothetical proteinが多いなあ…」とか、②「prophage proteinがある」とか…

Result Features DDBJ Submission Log

https://dfast.nig.ac.jp/analysis/annotation/f341d803-072b-48db-a363-1de4c3a686a5/f dfast.nig.ac.jp の待機中

Annotated Features

Show entries Search:

No.	Locus Tag	Seq. ID	Location	Feature Type	Product	Gene	Nucleotide	Trans
1	LOCUS_00001	sequence1	151..384	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
2	LOCUS_00002	sequence1	350..886	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
3	LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
4	LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
5	LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
6	LOCUS_00006	sequence1	2355..2732	CDS	prophage protein		<input type="button" value="View"/>	<input type="button" value="View"/>
7	LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
8	LOCUS_00008	sequence1	2930..3031	CDS	hypothetical protein		<input type="button" value="View"/>	<input type="button" value="View"/>
9	LOCUS_00009	sequence1	3067..3534	CDS	holin		<input type="button" value="View"/>	<input type="button" value="View"/>
10	LOCUS_00010	sequence1	3547..4578	CDS	1,4-beta-N-acetylmuramidase		<input type="button" value="View"/>	<input type="button" value="View"/>

W4-6: sequence1 概観

赤枠はsequence1の①19649から②34418 bpの範囲(全部で2,289,497 bpの長さがあるので、このあたりもまだ左端といえる)。赤下線で示すように、このあたりにも**ファージ(phage)**関連のものがちらほら存在

LOCUS	sequence	start	end	type	name	other	view
41	LOCUS_00041	sequence1	19314..19339	CDS	hypothetical protein		View
42	LOCUS_00042	sequence1	19649..20095	CDS	<u>phage protein</u>		View
43	LOCUS_00043	sequence1	20310..20585	CDS	hypothetical protein		View
44	LOCUS_00044	sequence1	20578..21057	CDS	hypothetical protein		View
45	LOCUS_00045	sequence1	21180..21698	CDS	<u>phage terminase small subunit</u>	xtmA	View
46	LOCUS_00046	sequence1	21698..23599	CDS	<u>phage terminase large subunit</u>	xtmB	View
47	LOCUS_00047	sequence1	23609..23785	CDS	hypothetical protein		View
48	LOCUS_00048	sequence1	23786..24961	CDS	<u>phage portal protein</u>		View
49	LOCUS_00049	sequence1	24942..26834	CDS	<u>phage capsid protein</u>		View
50	LOCUS_00050	sequence1	27026..27313	CDS	hypothetical protein		View
51	LOCUS_00051	sequence1	27294..27671	CDS	hypothetical protein		View
52	LOCUS_00052	sequence1	27674..28096	CDS	hypothetical protein		View
53	LOCUS_00053	sequence1	28096..28476	CDS	hypothetical protein		View
54	LOCUS_00054	sequence1	28480..29088	CDS	<u>phage tail protein</u>		View
55	LOCUS_00055	sequence1	29239..29589	CDS	tail protein		View
56	LOCUS_00056	sequence1	29793..34418	CDS	phage tail tape measure protein		View

W4-7: sequence2概観

2275	LOCUS_02275	sequence1	2288792..2289016	CDS	phage-related antirepressor		View	View
2276	LOCUS_02276	sequence1	2289054..2289158	CDS	hypothetical protein		View	View
2277	LOCUS_02277	sequence1	2289155..2289301	CDS	hypothetical protein		View	View
① 2278	LOCUS_02278	sequence2	complement (353..1006)	CDS	plasmid replication initiation protein		View	View
2279	LOCUS_02279	sequence2	1221..1391	CDS	transposase		View	View
2280	LOCUS_02280	sequence2	1671..1901	CDS	transposase		View	View
2281	LOCUS_02281	sequence2	3404..4126	CDS	ribose-5-phosphate isomerase A	rpiA_1	View	View
2282	LOCUS_02282	sequence2	complement (4735..5214)	CDS	resolvase		View	View
2283	LOCUS_02283	sequence2	5571..6200	CDS	transposase		View	View
2284	LOCUS_02284	sequence2	complement (6420..7796)	CDS	NADH oxidase		View	View
2285	LOCUS_02285	sequence2	complement (7912..9117)	CDS	pyridine nucleotide-disulfide oxidoreductase		View	View
2286	LOCUS_02286	sequence2	complement (9477..10061)	CDS	resolvase		View	View

第7回でsequence2は環状のプラスミド配列だろうと予想していたが、①それを補強するアノテーション結果

W4-7: sequence2概観

LOCUS	LOCUS ID	Sequence	Coordinates	Type	Function	Accession	View
2350	LOCUS_02350	sequence2	complement (59756..60343)	CDS	transposase		View
2351	LOCUS_02351	sequence2	60541..61773	CDS	hypothetical protein		View
2352	LOCUS_02352	sequence2	complement (62032..62334)	CDS	addiction module toxin RelE/StbE family protein		View
2353	LOCUS_02353	sequence2	complement (62324..62602)	CDS	antitoxin RelB		View
2354	LOCUS_02354	sequence2	62706..63293	CDS	integrase		View
2355	LOCUS_02355	sequence2	63668..63862	CDS	hypothetical protein		View
2356	LOCUS_02356	sequence2	64552..65073	CDS	plasmid replication initiation protein		View
2357	LOCUS_02357	sequence2	complement (65012..66325)	CDS	plasmid replication protein		View
2358	LOCUS_02358	sequence2	66931..67815	CDS	ATPase involved in chromosome partitioning		View
2359	LOCUS_02359	sequence2	67812..68225	CDS	hypothetical protein		View
2360	LOCUS_02360	sequence2	68731..69660	CDS	transposase		View
2361	LOCUS_02361	sequence2	69764..70123	CDS	6-phospho-beta-glucosidase	bgIB_2	View
2362	LOCUS_02362	sequence2	70107..70517	CDS	GntR family transcriptional regulator	gntR_5	View



W4-8: sequence3概観

2380	LOCUS_02380	sequence2	85034..85756	CDS	ribose-5-phosphate isomerase A	rpiA_2	View	View
2381	LOCUS_02381	sequence2	complement (86366..86848)	CDS	resolvase		View	View
2382	LOCUS_02382	sequence3	complement (192..440)	CDS	integral membrane protein		View	View
2383	LOCUS_02383	sequence3	653..802	CDS	diacylglycerol kinase		View	View
2384	LOCUS_02384	sequence3	871..1278	CDS	diacylglycerol kinase		View	View
2385	LOCUS_02385	sequence3	2306..2923	CDS	hypothetical protein		View	View
2386	LOCUS_02386	sequence3	2927..3805	CDS	hypothetical protein		View	View
2387	LOCUS_02387	sequence3	4302..5033	CDS	chromosome partitioning protein ParA	parA_2	View	View
2388	LOCUS_02388	sequence3	5166..5405	CDS	hypothetical protein		View	View
2389	LOCUS_02389	sequence3	5430..5768	CDS	hypothetical protein		View	View
2390		sequence3	6737..6884	repeat_region	CRISPR		View	
2391	LOCUS_02390	sequence3	7570..7767	CDS	hypothetical protein		View	View
2392	LOCUS_02391	sequence3	7818..8537	CDS	filamentation induced by cAMP protein Fic	fic	View	View
2393	LOCUS_02392	sequence3	complement	CDS	hypothetical protein		View	View

W4-8: sequence3概観

①赤下線で示す接合伝達(conjugal transfer)関連遺伝子が多く見られることから、sequence3がプラスミドであることを裏付けている

LOCUS	LOCUS ID	Feature Name	Coordinates	Type	Description	View
2396	LOCUS_02395	sequence3	11613..11924	CDS	hypothetical protein	View
2397	LOCUS_02396	sequence3	11963..12577	CDS	hypothetical protein	View
2398	LOCUS_02397	sequence3	12579..12914	CDS	<u>conjugal transfer protein</u>	View
2399	LOCUS_02398	sequence3	12935..13297	CDS	<u>conjugal transfer protein</u>	View
2400	LOCUS_02399	sequence3	13266..13925	CDS	<u>conjugal transfer protein</u>	View
2401	LOCUS_02400	sequence3	13937..15955	CDS	<u>conjugal transfer protein</u>	View
2402	LOCUS_02401	sequence3	15948..17366	CDS	<u>conjugal transfer protein</u>	View
2403	LOCUS_02402	sequence3	17367..18521	CDS	hypothetical protein	View
2404	LOCUS_02403	sequence3	18535..19152	CDS	hypothetical protein	View
2405	LOCUS_02404	sequence3	19106..19507	CDS	hypothetical protein	View
2406	LOCUS_02405	sequence3	19508..19978	CDS	<u>conjugal transfer protein</u>	View
2407	LOCUS_02406	sequence3	19980..21491	CDS	<u>conjugal transfer protein</u>	View
2408	LOCUS_02407	sequence3	21506..21895	CDS	hypothetical protein	View
2409	LOCUS_02408	sequence3	21914..22753	CDS	<u>conjugal transfer protein</u>	View
2410	LOCUS_02409	sequence3	22769..23179	CDS	hypothetical protein	View



①

W4-9: sequence4概観

LOCUS	LOCUS ID	Sequence	Coordinates	Feature	Description	Accession	View
2437	LOCUS_02436	sequence3	43900..44526	CDS	hypothetical protein		View
2438	LOCUS_02437	sequence4	51..335	CDS	oxidoreductase		View
2439	LOCUS_02438	sequence4	391..513	CDS	oxidoreductase		View
2440	LOCUS_02439	sequence4	523..1263	CDS	oxidoreductase		View
2441	LOCUS_02440	sequence4	complement (1544..1999)	CDS	LysR substrate binding domain protein		View
2442	LOCUS_02441	sequence4	complement (1966..2214)	CDS	LysR family transcriptional regulator		View
2443	LOCUS_02442	sequence4	complement (2524..2892)	CDS	phosphoglycerate mutase		View
2444	LOCUS_02443	sequence4	3067..3843	CDS	sorbose-specific PTS system IIC component		View
2445	LOCUS_02444	sequence4	3858..4151	CDS	mannose-specific PTS system IID component	ptnD_2	View
2446	LOCUS_02445	sequence4	4133..4753	CDS	mannose-specific PTS system IID component	ptnD_3	View
2447	LOCUS_02446	sequence4	4781..4873	CDS	mannose-specific PTS system IIA component		View
2448	LOCUS_02447	sequence4	4912..5220	CDS	transposase		View
2449	LOCUS_02448	sequence4	5361..5699	CDS	transposase		View
2450	LOCUS_02449	sequence4	5711..6058	CDS	mannose-specific PTS system IIA component		View

W4-9: sequence4概観

LOCUS	sequence4	coordinates	CDS	protein name	accession	view
2446	LOCUS_02445	sequence4	4155..4755	mannose-specific PTS system IID component	panD_3	View
2447	LOCUS_02446	sequence4	4781..4873	mannose-specific PTS system IIA component		View
2448	LOCUS_02447	sequence4	4912..5220	transposase		View
2449	LOCUS_02448	sequence4	5361..5699	transposase		View
2450	LOCUS_02449	sequence4	5711..6058	mannose-specific PTS system IIA component		View
2451	LOCUS_02450	sequence4	6114..6596	mannose/fructose/sorbose-specific PTS system IID component		View
2452	LOCUS_02451	sequence4	6738..7010	hypothetical protein		View
2453	LOCUS_02452	sequence4	7028..7204	hypothetical protein		View
2454	LOCUS_02453	sequence4	7515..7844	excinuclease ABC subunit A	uvrA_3	View
2455	LOCUS_02454	sequence4	8068..9012	excinuclease ABC subunit A	uvrA_4	View
2456	LOCUS_02455	sequence4	9012..9446	excinuclease ABC subunit A	uvrA_5	View
2457	LOCUS_02456	sequence4	9552..10025	excinuclease ABC subunit A	uvrA_6	View
2458	LOCUS_02457	sequence4	10378..10611	penicillin-binding protein 2A		View
2459	LOCUS_02458	sequence4	10608..11261	penicillin-binding protein 2A		View

Showing 1 to 2,459 of 2,459 entries

Previous 1 Next

第8回原稿p188の右上

ゲノムアノテーション

ゲノムのアノテーション (genome annotation) とは、ゲノム上のどの位置にどのような遺伝子がコードされているかなどを調べ、注釈づけを行う作業である。バクテリアの自動アノテーションに関しては、MiGAP⁹⁾ や RAST¹⁰⁻¹²⁾ などの様々なウェブサービスが提供されており、手軽に実行可能である。今回は、乳酸菌に特化したアノテーションパイプライン DFAST (DDBJ Fast Annotation and Submission Tool)¹³⁾ を用いる。本ウェブサービスは、連載第5回¹⁴⁾ でも紹介したアノテーションパイプライン Prokka¹⁵⁾ をベースとして、乳酸菌 (主に *Lactobacillus* 属および *Pediococcus* 属) 用に整備された参照データベースを組み合わせたものである。また、アノテーションだけでなく、DDBJ¹⁶⁾ への塩基配列登録支援を行うこともできる (もちろん登録は任意) のが特徴である。典型的なゲノムサイズ (数 MB 程度) の乳酸菌であれば、5分ほどで結果が返される。ここでは、アセンブリ結果の検証の一環として、アセンブリ結果ファイル (LH_hgap.fa) を入力として DFAST を実行する [W4]。位置づけとしては、予備的なアノテーションである。

第7回までで予想していた通り、この2つの配列が (環状の) プラスミドであることを支持している。

sequence1 (2,289,497 bp) については、最初の 30,000 塩基あたりまでに prophage protein などファージ関連遺伝子が見られる [W4-6]。また、sequence4 (11,372 bp) については、4,912 番目から 5,699 番目の領域に transposase がコードされている。この領域については後で議論するが、アノテーション結果の概観レベルでは見逃してもよい。後述する BLAST¹⁷⁾ の実行結果と合わせて総合的に判断する視点が重要である。

①

BLAST の実行と可視化

第7回では、sequence3 同士を例として配列内の両末端部分の重複をドットプロットで大まかに調べ (第7回 W14-2)、BLAST で重複領域の詳細なアラインメントを行った。ドットプロットについては、sequence3 よりも2倍程度長い sequence2 同士についても dotter¹⁸⁾ を実行可能であり、両末端部分の重複が見いだせる [W5-1]。sequence4 同士のドットプロットは、[1, 500 bp] と [750, 1350 bp] 付近の領域が似ているものの、環状を示唆す

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



W5-2: dotter (sequence4)

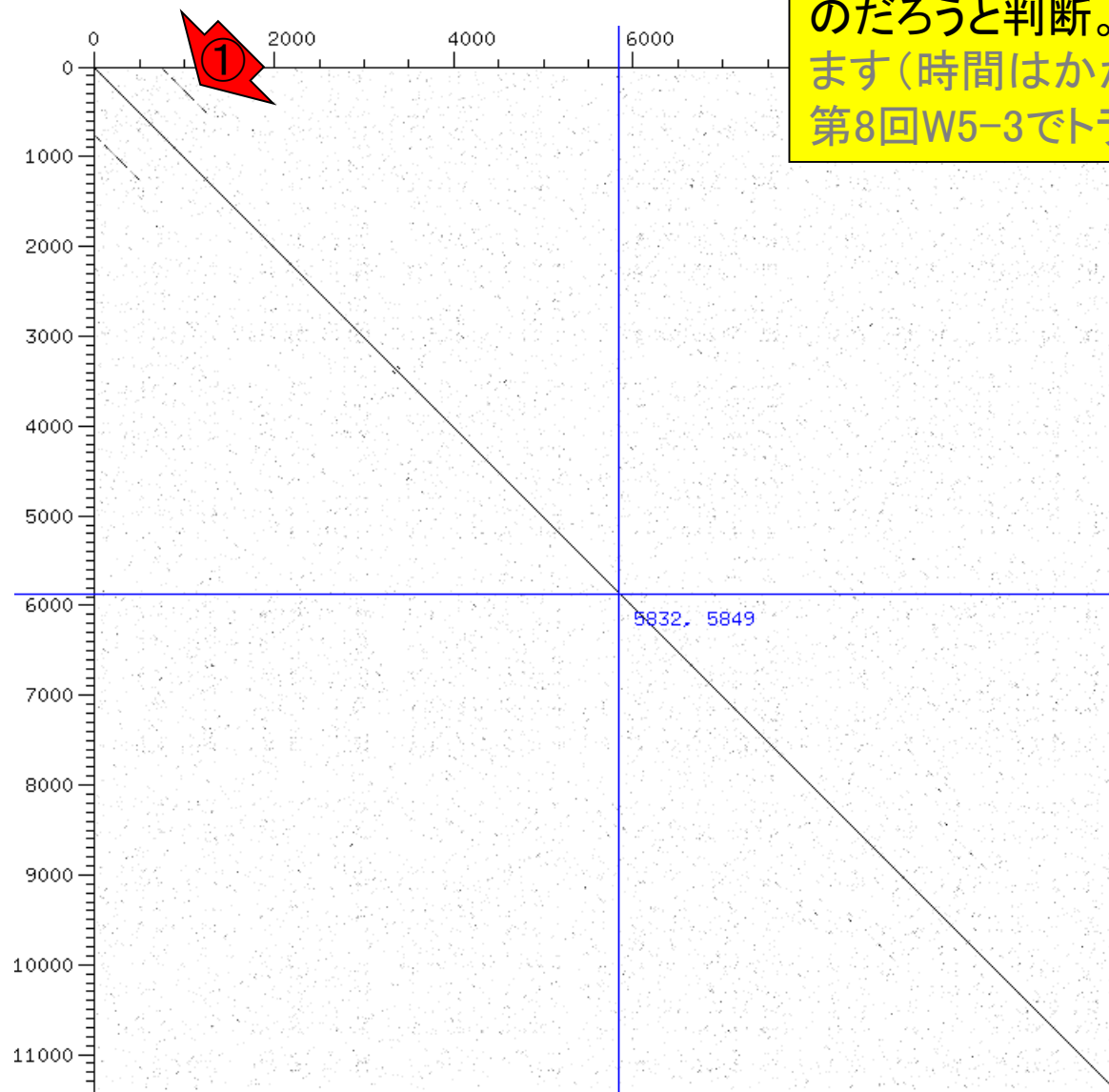
```

iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 1:44午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls [ 1:44午後 ]
corrected.fastq      sequence3_blast.txt  sequence3_trimmed.fa
hoge1.png            sequence3.fa          sequence3_trimmed.fq
hoge2.fa             sequence3.fa.nhd      sequence3_trimmed.png
hoge2.png            sequence3.fa.nhi      sequence3_trimmed.txt
hoge.fa              sequence3.fa.nhr      sequence3.txt
LH_hgap.fa           sequence3.fa.nin      sequence4.fa
polished_assembly.fasta sequence3.fa.nog      sequence4.fq
polished_assembly.fastq sequence3.fa.nsd      sequence4.png
sequence1.fa         sequence3.fa.nsi      sequence4.txt
sequence1.fq         sequence3.fa.nsq      smrtpipe.log
sequence2.fa         sequence3.fq
sequence2.fq         sequence3.png
iu@bielinux[result] dotter sequence4.fa sequence4.fa [ 1:44午後 ]
    
```



W5-2: dotter (sequ

①この結果は、(1 - 500)番目と(750 - 1,350)番目付近の領域が似ていることを意味する。しかし両末端ではないので、sequence4は環状(プラスミド)ではないのだらうと判断。sequence2同士は第8回W5-1にあります(時間はかかるが実行可能)。sequence1同士は第8回W5-3でトライしています(事実上実行不可能)



第8回原稿p188の右下

今は①のあたり。sequence1をquery側の配列、sequence1-4をDB側の配列としてBlast実行する話へと移行していきます

ンパイライン Prokka¹⁵⁾ をベースとして、乳酸菌（主に *Lactobacillus* 属および *Pediococcus* 属）用に整備された参照データベースを組み合わせたものである。また、アノテーションだけでなく、DDBJ¹⁶⁾ への塩基配列登録支援を行うこともできる（もちろん登録は任意）のが特徴である。典型的なゲノムサイズ（数 MB 程度）の乳酸菌であれば、5分ほどで結果が返される。ここでは、アセンブリ結果の検証の一環として、アセンブリ結果ファイル（LH_hgap.fa）を入力として DFAST を実行する [W4]。位置づけとしては、予備的なアノテーションである。

オプションとして、Job Title には好きな名称をつけられ、DDBJ Pipeline 実行時と同じくジョブ完了通知をメールで受け取ることもできる。“Minimum Contig Length” オプションは、設定値以下の短い断片配列を除くためのものである。今回の入力配列は、全てデフォルトの 200 塩基以上なので影響はない。属・種名などのオプションは、デフォルトのままで構わない。アノテーション結果に影響を与えることはなく、後で変更することも可能だからである [W4-3]。主なアノテーション結果は、入力ファイル中

BLAST の実行と可視化

第7回では、sequence3 同士を例として配列内の両末端部分の重複をドットプロットで大まかに調べ（第7回 W14-2）、BLAST で重複領域の詳細なアラインメントを行った。ドットプロットについては、sequence3 よりも2倍程度長い sequence2 同士についても dotter¹⁸⁾ を実行可能であり、両末端部分の重複が見いだせる [W5-1]。sequence4 同士のドットプロットは、[1, 500 bp] と [750, 1350 bp] 付近の領域が似ているものの、環状を示唆する結果は得られなかった [W5-2]。約 2.3Mb と最も長い sequence1 同士は、ゲスト OS (Bio-Linux) への割り当てメモリが 2GB の著者らの PC 環境では、dotter を実行できなかった（数分程度では描画できなかった） [W5-3]。

通常は、sequence1 vs. sequence1 のような同一配列間の比較以外にも、sequence1 vs. sequence4 のような異なる配列間の比較も行い、配列類似領域を探索する。合計 4 配列しかないこのアセンブリ結果の場合はそれほどの作業量でもないが、ペアワイズアラインメントだと一般に組合

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p188の右下

①sequence1をquery側の配列、sequence1-4を含むLH_hgap.faをDB側の配列として、Blast実行する話へと移行。

②Blastだといっぺんに調べられて便利

る結果はsequence1同士は、ゲストOS (Bio-Linux) への割り当てメモリが2GBの著者らのPC環境では、dotterを実行できなかった (数分程度では描画できなかった) [W5-3]。

①

通常は、sequence1 vs. sequence1のような同一配列間の比較以外にも、sequence1 vs. sequence4のような異なる配列間の比較も行い、配列類似領域を探索する。合計4配列しかないこのアセンブリ結果の場合はそれほどの作業量でもないが、ペアワイズアラインメントだと一般に組合せ数が膨大になる。BLASTはこのような局面でも威力を発揮する。例えば、query側の配列としてsequence1を、データベース (DB) 側の配列としてLH_hgap.faを指定すれば、(自分自身を含む) 4通りの比較に相当する [W6]。

②

オプションとして、Job Titleには好きな名称をつけられ、DDBJ Pipeline 実行時と同じくジョブ完了通知をメールで受け取ることもできる。“Minimum Contig Length”オプションは、設定値以下の短い断片配列を除くためのものである。今回の入力配列は、全てデフォルトの200塩基以上なので影響はない。属・種名などのオプションは、デフォルトのままで構わない。アノテーション結果に影響を与えることはなく、後で変更することも可能だからである [W4-3]。主なアノテーション結果は、入力ファイル中の配列の順番通りに、「どの配列 (コンティグ) 中のどの座標上にどんな遺伝子が存在するか」という情報である。ウェブ上のFeaturesタブ経由で見られる情報以外に、拡張子が.gbkのGenbank形式やGFF3形式ファイルがダウ

W6-1 : makeblastdb

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:48午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls LH_hgap.fa* [ 3:48午後 ]
LH_hgap.fa
iu@bielinux[result] makeblastdb -in LH_hgap.fa -dbtype nucl -hash_index
X
Building a new DB, current time: 06/21/2017 15:48:19
New DB name: LH_hgap.fa
New DB title: LH_hgap.fa
Sequence type: Nucleotide
Keep Linkouts: T
Keep MBits: T
Maximum file size: 10000000000B
Adding sequences from FASTA; added 4 sequences in 0.0474792 seconds.
iu@bielinux[result] ls LH_hgap.fa* [ 3:48午後 ]
LH_hgap.fa LH_hgap.fa.nhr LH_hgap.fa.nsd
LH_hgap.fa.nhd LH_hgap.fa.nin LH_hgap.fa.nsi
LH_hgap.fa.nhi LH_hgap.fa.nog LH_hgap.fa.nsq
iu@bielinux[result] █ [ 3:48午後 ]
```


①DB側をLH_hgap.fa、query側をsequence1.fa、出力ファイルをsequence1_blast.txtとしてblastnを実行。
②出力ファイルは12MB。sequence1.fa同士のアラインメント結果を丸々含んでいるためであろう

W6-2: blastn

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:55午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
① iu@bielinux[result] blastn -db LH_hgap.fa -query sequence1.fa \
> -out sequence1_blast.txt
iu@bielinux[result] ls -l sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 12473982 6月 21 2017 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] ls -lh sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 12M 6月 21 2017 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 2.2M 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4.4M 6月 13 17:09 sequence1.fq
② iu@bielinux[result] █ [ 3:56午後 ]
```


ヒット数

① "Score =" という文字列を含む行数は1,351個。つまりヒット数は1,351。これだけ多いと全体像の把握が困難なので、Blast結果閲覧専用のViewerを利用する

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:55午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] blastn -db LH_hgap.fa -query sequence1.fa \
> -out sequence1_blast.txt
iu@bielinux[result] ls -l sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 12473982 6月 21 2017 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] ls -lh sequence1* [ 3:56午後 ]
-rwxrwxrwx 1 iu iu 12M 6月 21 2017 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 2.2M 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4.4M 6月 13 17:09 sequence1.fq
① iu@bielinux[result] grep -c "Score =" sequence1_blast.txt
1351
iu@bielinux[result] [ 4:17午後 ]
```

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p189の左上

①Blast結果閲覧用Viewerの1つであるBlastViewerで説明していきます

但し、sequence1は非常に長いため、デフォルト出力形式のBLAST実行結果ファイル(sequence1_blast.txt)を眺めて全体像を把握するのは困難である。

もちろん、BOV¹⁹⁾やBLASTGrabber²⁰⁾ [W7-1]など、BLAST実行結果の全体像を把握するための可視化ソフトウェア(ビューワ; viewer)は存在する。ここでは、BlastViewerを利用する [W7-2]。BlastViewerはWindows用とMacintosh用のみが提供されているため、ホストOS上でインストールして利用する。XML形式の

BLAST実行結果ファイル(sequence1_blast.xml)しか受け付けないが、DB側の配列ごとにヒット数(配列類似領域数; HSP数)が示されているなど、全体的な操作感がよい [W7-4]。例えば、sequence1に対するヒット数が1,347個、sequence4が2個、sequence3とsequence2がそれぞれ1個であったことがわかる。また、ヒット数やスコア分布の全体像を眺めることで、sequence1にいくつかの重複領域が存在することや、11,372 bpからなるsequence4の大部分の領域がsequence1と類似していることなどがわかる [W8-1]。

た [W8-3]。また、スコアの低いほう (Score=8,907) の2つめのHSP (以下、HSP2) は、sequence1の領域 [555167, 560027 bp] と、sequence4の領域 [4852, 1 bp] から形成されていた [W8-4]。いずれのHSPも、sequence1がPlus (+) 鎖、sequence4がMinus (-) 鎖でアラインメントされていた。これは、sequence1の一続きの領域 [549494, 560027 bp] と、領域 [4853, 5705 bp] を除くsequence4の全長がほぼ一致していることを意味する (図1a; W8-5)。

② アラインメントされなかったsequence4の領域 [4853, 5705 bp] に対するアノテーション結果を眺めると、transposaseがコードされていたことがわかる (図1b; W8-5)。これは、該当領域が挿入配列(insertion sequence; IS)であることを示唆する。これはおそらく、乳酸菌の培養途中で一部の細胞にISの挿入が起こったためであろう。結果として、シーケンスされた細胞集団の中にISを含むものと含まないものが混在することになり、sequence4が独立したコンティグとして出力されたものと思われる。sequence4は全体的にクオリティスコアが低く

W7-2: BlastViewer

①Korilogというところが提供している BlastViewerのウェブページ。②Windows版と③Macintosh版。ときどき②と③の上下が入れ替わるようなので注意。次のスライド以降はWindows版で説明

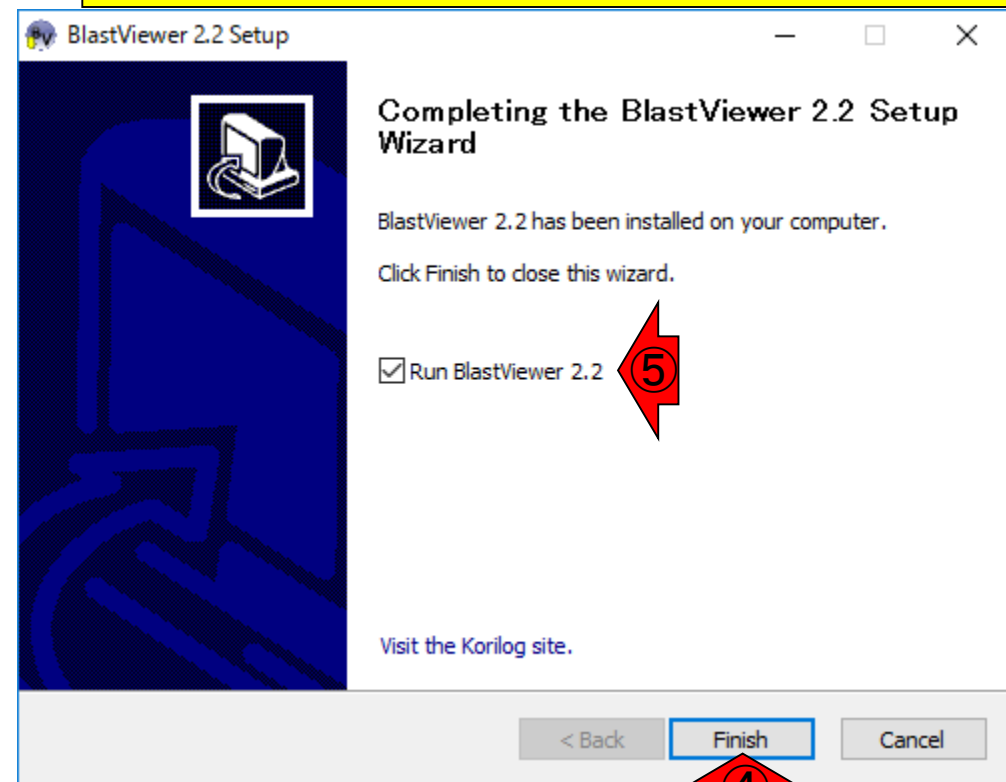
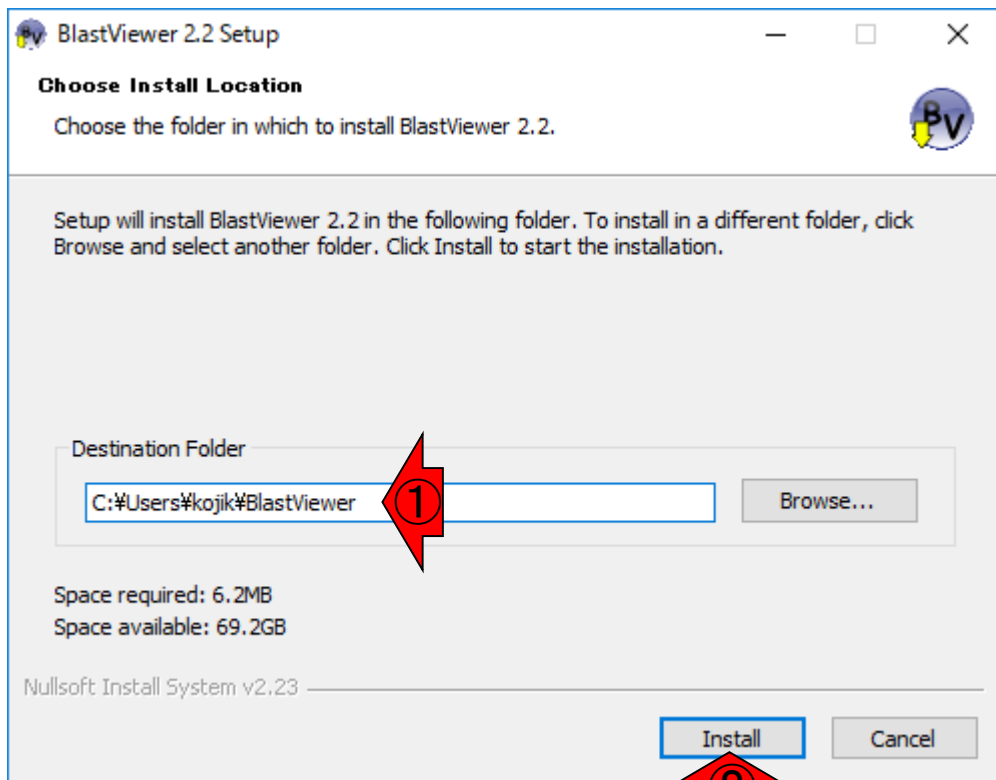
The screenshot shows the Korilog website interface. At the top, there is a navigation bar with links for CNET, REVIEWS, NEWS, DOWNLOAD, VIDEO, and HOW TO. Below this is a search bar and icons for Windows, Apple, and Android. A large green button labeled 'Start Download' is prominent, with a sub-button 'Start Download' and a list of steps: 1. Click to Start Download, 2. Run and Install, 3. Scan for Issues.

The main content area features the 'Korilog' logo (marked with a red arrow '1') and a brief description of the company. Below this is a 'Narrow Results' section with filters for 'By Price' (Free (2)), 'By Category' (Educational Software (2)), and 'By Operating System' (Mac OS X 10.4 (1), Macintosh (1), Windows (1), Windows 2000 (1), Windows Vista (1), Windows XP (1)).

The search results list two 'BlastViewer' products. The first is for Windows (marked with a red arrow '2'), described as 'Analyze the reports produced by the NCBI's BLAST sequence database search system.' It has 411 total downloads and 3 last week. The second is for Mac (marked with a red arrow '3'), described as 'View and analyze data contained in a Blast result file.' It has 278 total downloads and 1 last week. Both entries have a 'Download' button.

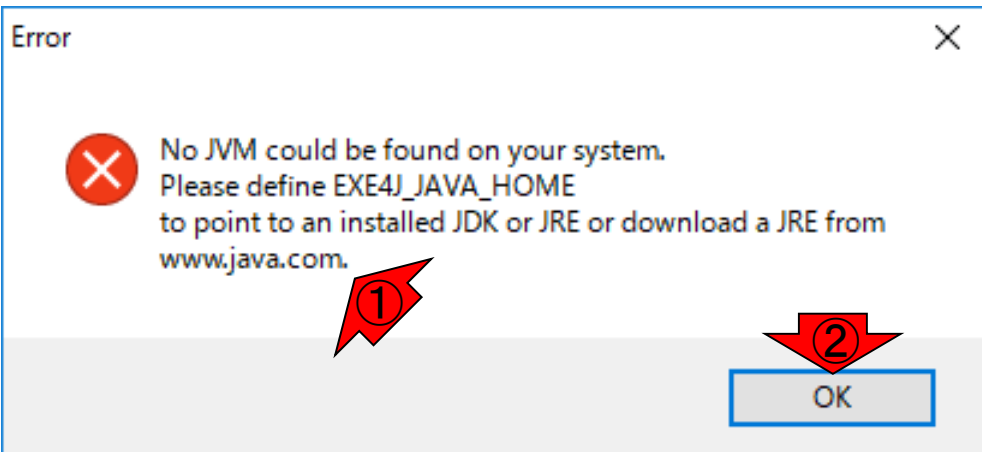
W7-2: BlastViewer

Windows 10上でのインストール画面。①ここに作成されるようだ。②Install。③デスクトップにBlastViewerのアイコンが作成されるはず。④Finish。デフォルトでは⑤にチェックが入っているのでBlastViewer (ver. 2.2)が起動する



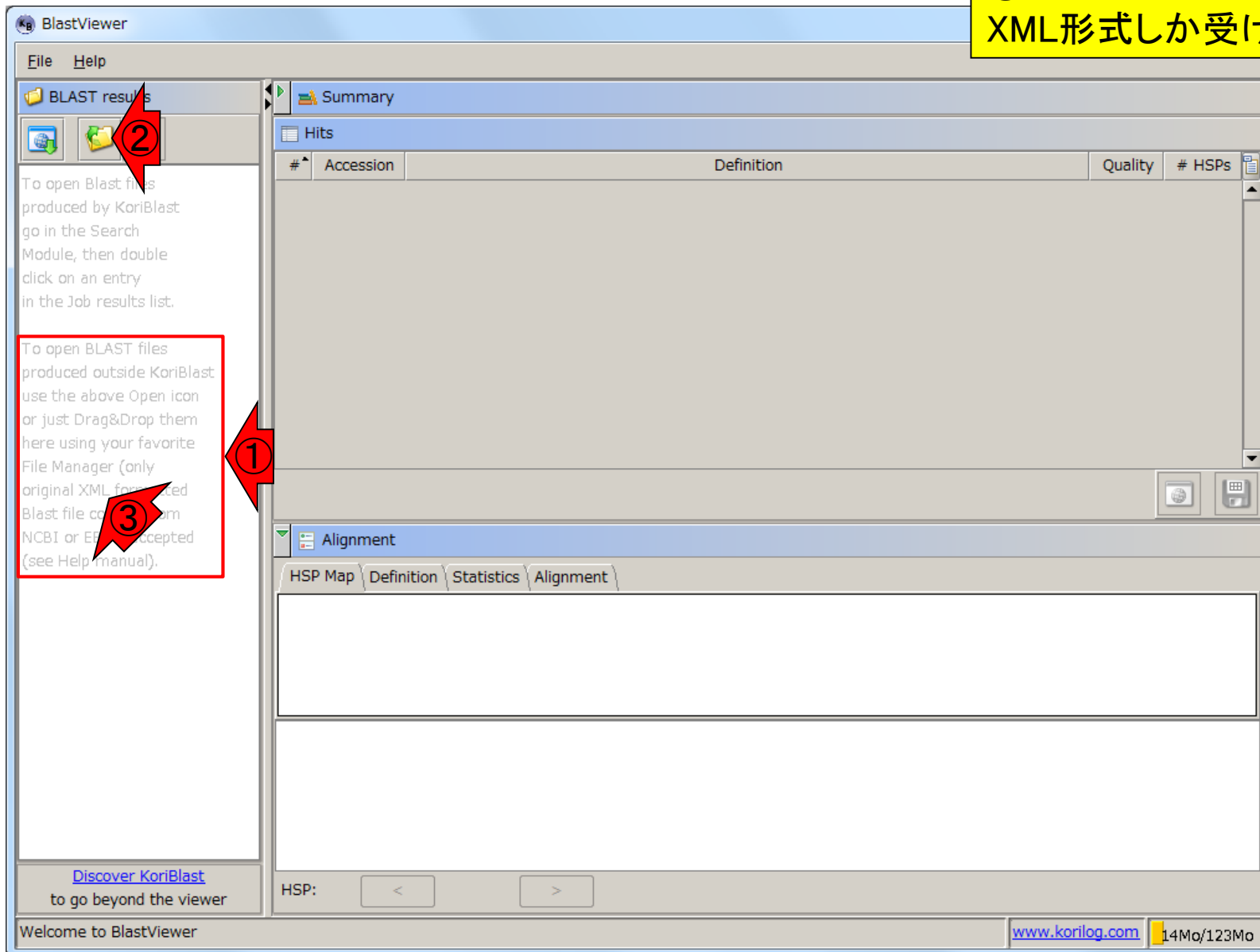
Java関連でエラーが出たら

BlastViewer (ver. 2.2)が起動できず、①Java関連でエラーが出たら、③どうにかして自力でインストールしてください。アグリバイオ貸与PCはJavaインストール済み(なはず)



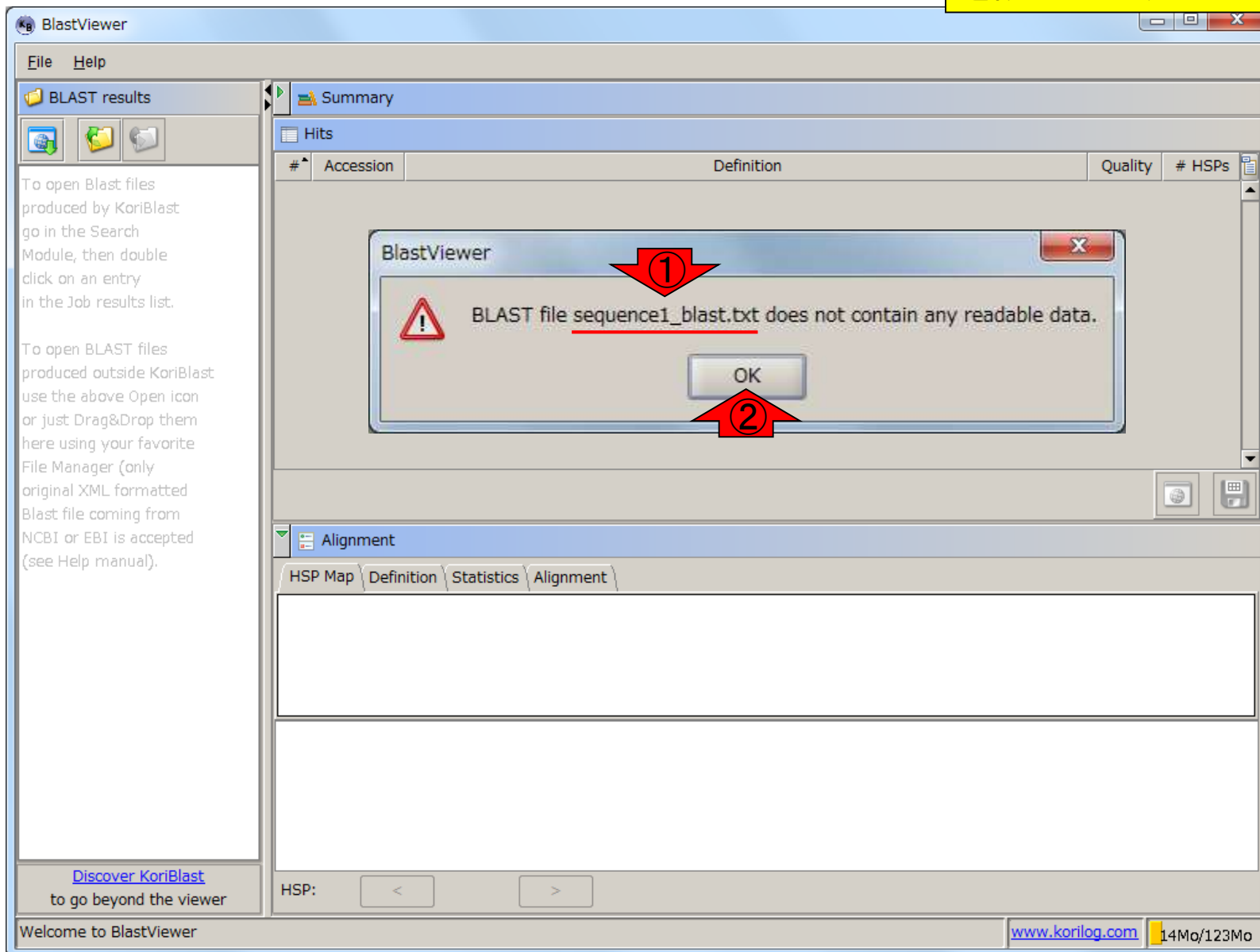
W7-2: BlastViewer

①を見ることで、Blast結果ファイルを開く際には、②を押せばよいことがわかる。③をよく見るとBlast出力結果ファイルはXML形式しか受け付けていないようだ



W7-2: BlastViewer

①ためしにW6-2で作成したテキスト形式のBlast結果ファイル(sequence1_blast.txt)を読み込もうとしたらダメでした。②OK



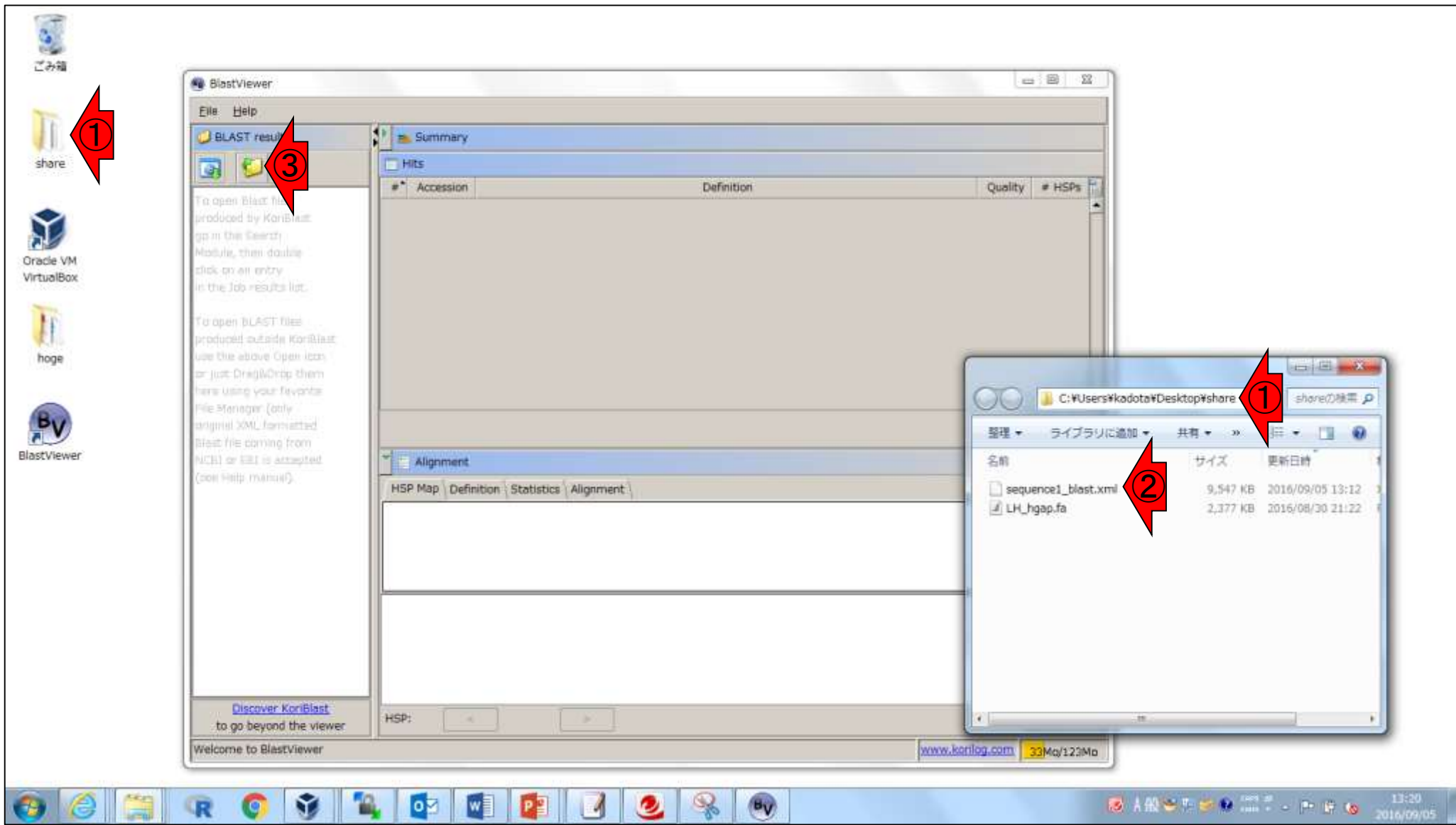
①XML形式のBlast結果ファイル(sequence1_blast.xml)を作成。②確かにあります

W7-3: Blast再実行

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 1:16午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1* [ 1:17午後 ]
-rwxrwxrwx 1 iu iu 12473982 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] blastn -db LH_hgap.fa -query sequence1.fa \
> -out sequence1_blast.xml -outfmt 5
iu@bielinux[result] ls -lh sequence1* [ 1:18午後 ]
-rwxrwxrwx 1 iu iu 12M 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 9.4M 6月 22 2017 sequence1_blast.xml
-rwxrwxrwx 1 iu iu 2.2M 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4.4M 6月 13 17:09 sequence1.fq
iu@bielinux[result] [ 1:18午後 ]
```

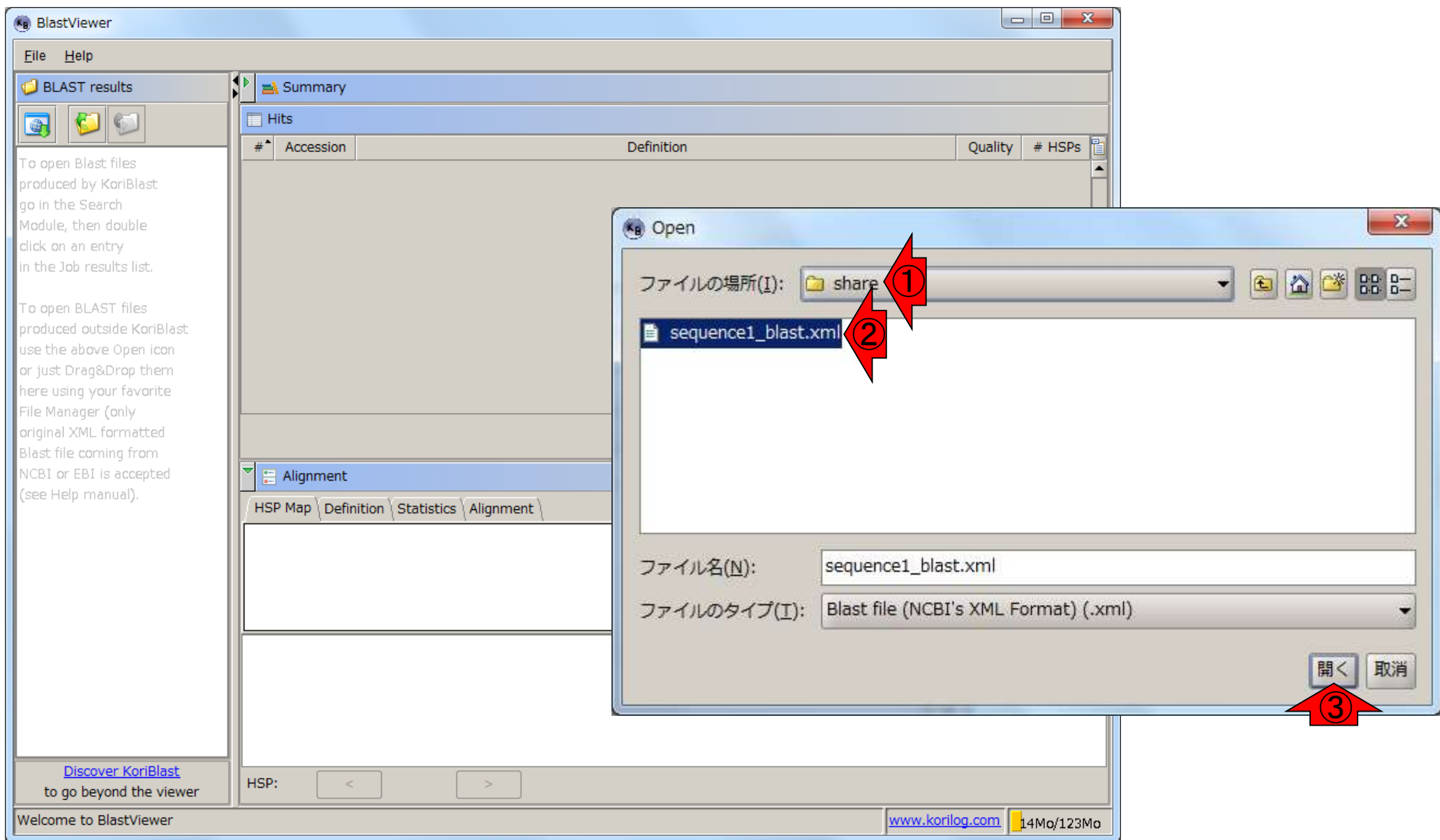


W7-4: BlastViewer



①共有フォルダ上の、②XML形式のBlast結果ファイル(sequence1_blast.xml)を、③開く

W7-4: BlastViewer



W7-4: BlastViewer

読み込み後の状態。①query側が sequence1.fa、②DB側がsequence1-4からなるLH_hgap.faだったことを思い出そう

The screenshot shows the BlastViewer application window. On the left, a file explorer shows 'sequence1_blast.xml' and 'blastn vs. LH_hgap.fa'. A red arrow labeled '1' points to the XML file, and another red arrow labeled '2' points to the FASTA file. The main window displays a 'Summary' tab with a table of hits:

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	😊	1347
2	3	sequence4	😊	2
3	2	sequence3	😊	1
4	1	sequence2	😊	1

Below the table, the 'Alignment' tab is active, showing 'Alignment: Query (2289497 nuc) vs. 0 (2,289,497 nuc)'. It displays a sequence alignment with a color-coded HSP map above it. The alignment shows a perfect match between the query and the database sequence. At the bottom, the status bar indicates 'HSP: 1/1347' and provides navigation buttons.

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



①テキスト形式のBLAST結果ファイル(sequence1_blast.txt)をlessで眺めて全体像を大まかに把握

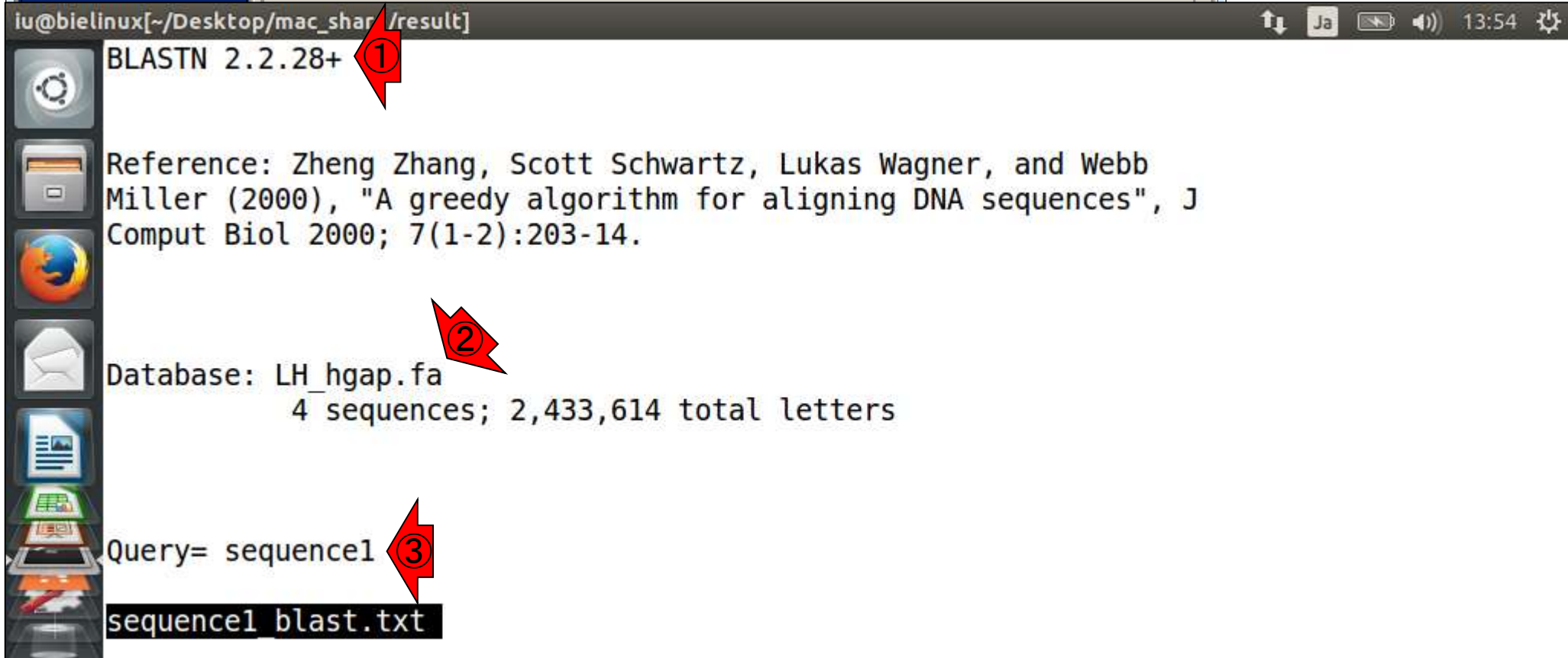
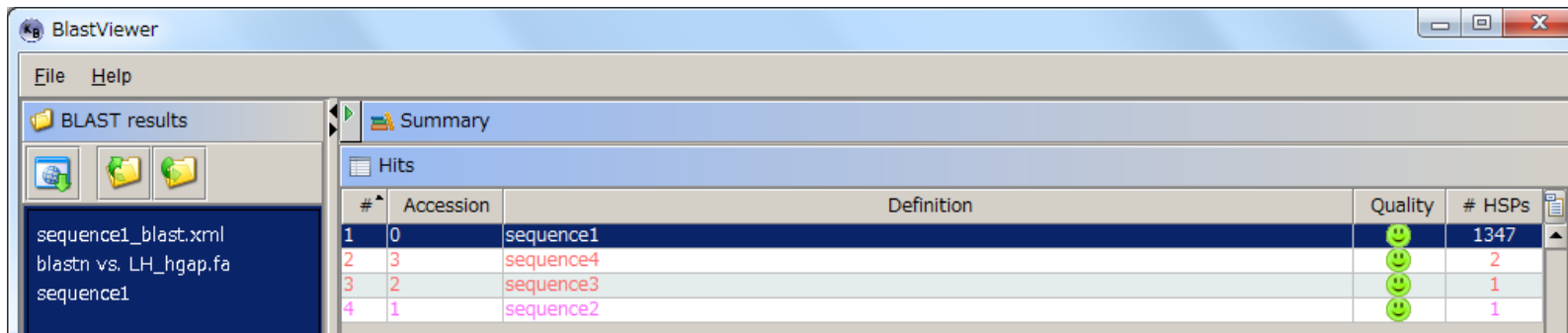
W7-5: 解釈2

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	1347	1
2	3	sequence4	2	1
3	2	sequence3	1	1
4	1	sequence2	1	1

```

iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1*
-rwxrwxrwx 1 iu iu 12473982  6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu  9775990  6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu  2289509  6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu  4579015  6月 13 17:09 sequence1.fq
iu@bielinux[result] less sequence1_blast.txt
    
```


W7-5: 解釈3



W7-5: 解釈4

1ページ分ほど下に移動(上下左右の矢印キーで移動)。①確かにBlastViewerと同じ並び(sequence1, 4, 3, 2)になっている

The screenshot shows the BlastViewer application window. The 'Summary' tab is active, displaying a table of hits. The table has columns for '#', 'Accession', 'Definition', 'Quality', and '# HSPs'. The data is as follows:

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	😊	1347
2	3	sequence4	😊	2
3	2	sequence3	😊	1
4	1	sequence2	😊	1

The terminal window shows the command prompt and the output of a BLAST search. The output is as follows:

```

iu@bielinux[~/Desktop/mac_share/result]
Sequences producing significant alignments:
sequence1      4.228e+06    0.0
sequence4     1.032e+04    0.0
sequence3      2113         0.0
sequence2      124          1e-25

> sequence1
Length=2289497

Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
    
```

A red box highlights the list of sequences and their scores/E-values. A red arrow with the number '1' points to the first sequence, 'sequence1'.

W7-5: 解釈5

①これがトップヒットの基本情報。query (sequence1)とDB (sequence1-4)の関係が分かっているならば、これがsequence1 vs. sequence1の100%一致の結果であることがわかる。スコアの計算方法がよくわかっていなくても、大まかに配列長(2,289,497 bp)の2倍程度の値がスコアっぽいなどと学習する

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	1347	1
2	3	sequence4	2	1
3	2	sequence3	1	1
4	1	sequence2	1	1

```

iu@bielinux[~/Desktop/mac_share/result]
Sequences producing significant alignments:

sequence1          4.228e+06  0.0
sequence4          1.032e+04  0.0
sequence3           2113       0.0
sequence2           124        1e-25

> sequence1
Length=2289497

Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
    
```



W7-5: 解釈6

となると、いくら①E-valueがそこそこ低くても、② sequence2にヒットしているsequence1の断片配列の領域は、スコアが124なので60塩基程度だろうと予想。また、③sequence3にヒットしているsequence1の断片配列の領域は、スコアが2113なので1000塩基程度だろうと予想

BlastViewer Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	😊	1347
2	3	sequence4	😊	2
3	2	sequence3	😊	1
4	1	sequence2	😊	1


```

iu@bielinux[~/Desktop/mac_share/result]
Sequences producing significant alignments:

sequence1          4.228e+06  0.0
sequence4          1.032e+04  0.0
sequence3          2113      0.0
sequence2          124       1e-25

> sequence1
Length=2289497

Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
    
```

Annotations in the terminal window:

- ①: Points to the settings icon in the terminal window's title bar.
- ②: Points to the score '124' for sequence2.
- ③: Points to the E-value '0.0' for sequence3.

① sequence2でキーワード検索し、
②のアラインメント結果を眺めます

W7-5: 解釈7

BlastViewer Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	😊	1347
2	3	sequence4	😊	2
3	2	sequence3	😊	1
4	1	sequence2	😊	1

```

iu@bielinux[~/Desktop/mac_share/result]
Sequences producing significant alignments:

sequence1          4.228e+06  0.0
sequence4          1.032e+04  0.0
sequence3           2113       0.0
sequence2          124        1e-25

> sequence1
Length=2289497

Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
/sequence2
    
```

Red arrow ② points to the row for sequence2 in the alignment list.

Red arrow ① points to the prompt for sequence2 in the detailed view.

W7-5: 解釈8

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	0	1347
2	3	sequence4	3	2
3	2	sequence3	2	1
4	1	sequence2	1	1

```

iu@bielinux[~/Desktop/mac_share/result]
sequence2
124 1e-25
> sequence1
Length=2289497

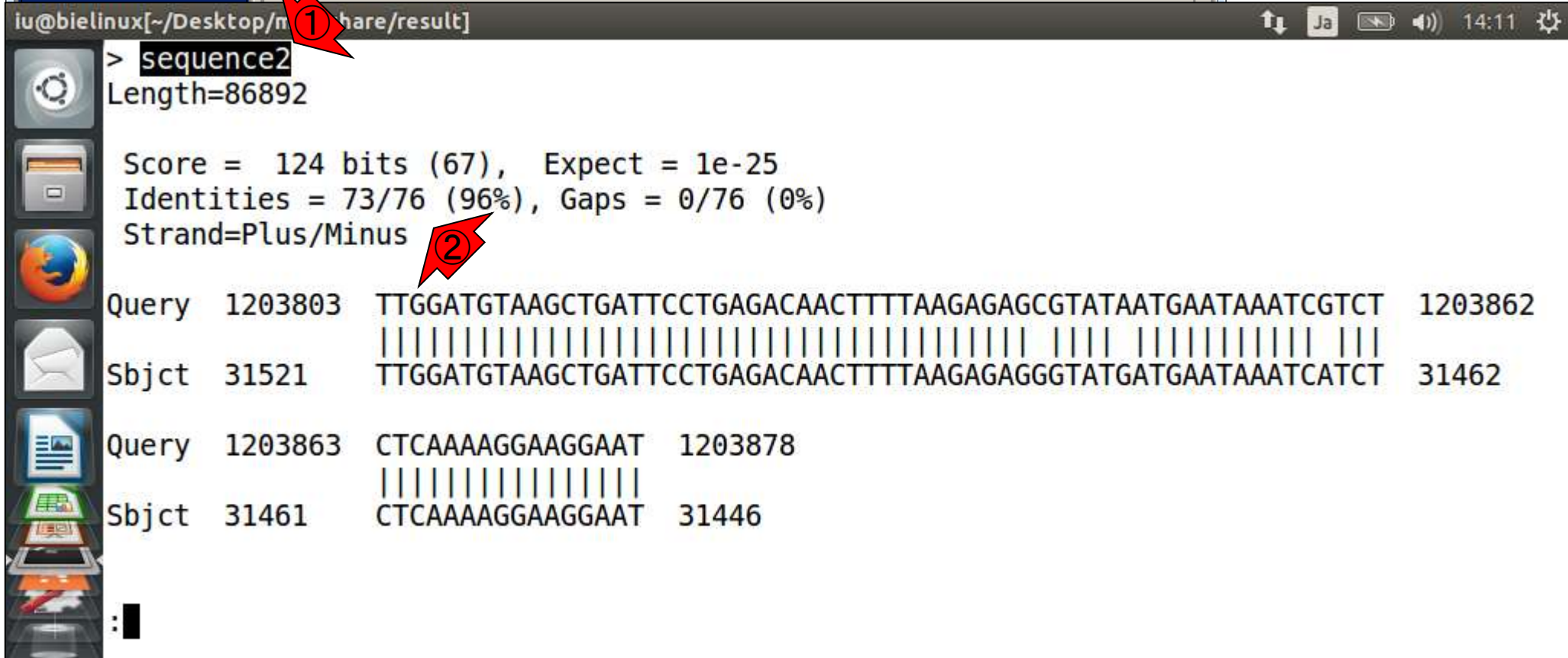
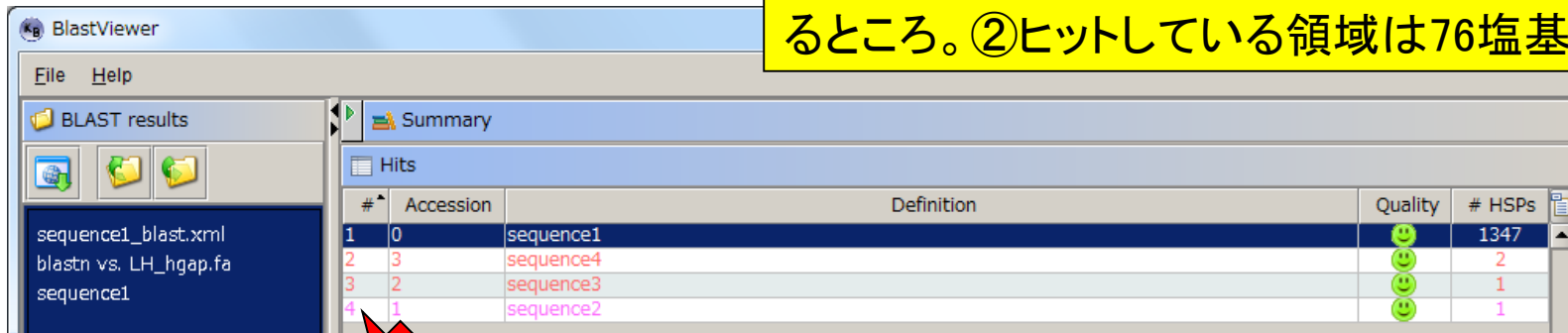
Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus

Query 1 TGTTGGGCTGCTGAATGAACATAGCGAATTTGCCCGGAACTACTTTTTGGCGGTGGCAA 60
|
Sbjct 1 TGTTGGGCTGCTGAATGAACATAGCGAATTTGCCCGGAACTACTTTTTGGCGGTGGCAA 60

Query 61 TCGCGCTGACAGATTTACGCTCAAAGGAAACCATGATGATGGTAGTGGCAGGATTCTGCA 120
|
:
    
```


W7-5: 解釈9

今は、テキスト形式のBLAST結果ファイル(sequence1_blast.txt)をlessで眺めている。①sequence2でキーワード検索(nで順方向に検索、Nで逆方向に検索)し、アラインメントを表示させているところ。②ヒットしている領域は76塩基



W7-5: 解釈10

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

```

iu@bielinux[~/Desktop/mac_share/result]
> sequence2
Length=86892

Score = 124 bits (67), Expect = 1e-25
Identities = 73/76 (96%), Gaps = 0/76 (0%)
Strand=Plus/Minus

Query 1203803 TTGGATGTAAGCTGATTCCTGAGACAACCTTTTAAGAGAGCGTATAATGAATAAATCGTCT 1203862
          |||
Sbjct 31521   TTGGATGTAAGCTGATTCCTGAGACAACCTTTTAAGAGAGGGTATGATGAATAAATCATCT 31462

Query 1203863 CTCAAAAGGAAGGAAT 1203878
          |||
Sbjct 31461   CTCAAAAGGAAGGAAT 31446

/sequence3
    
```



W7-5: 解釈11

最初はこんな感じになるかもしれないが、「nで順方向に検索、Nで逆方向に検索」なので、Nと打ってページ上部を探すと…

BLAST results Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

```

iu@bielinux[~/Desktop/mac_share/result]
> sequence2
Length=86892

Score = 124 bits (67), Expect = 1e-25
Identities = 73/76 (96%), Gaps = 0/76 (0%)
Strand=Plus/Minus

Query 1203803 TTGGATGTAAGCTGATTCCTGAGACAAC...TTTAAGAGAGCGTATAATGAATAAATCGTCT 1203862
      |||
Sbjct 31521   TTGGATGTAAGCTGATTCCTGAGACAAC...TTTAAGAGAGGGTATGATGAATAAATCATCT 31462

Query 1203863 CTCAAAGGAAGGAAT 1203878
      |||
Sbjct 31461   CTCAAAGGAAGGAAT 31446

Pattern not found (press RETURN)
    
```


こんな感じになって、①sequence3のアラインメント結果を見ることができます。②ヒットしている領域は1276塩基

W7-5: 解釈12

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	1347	1
2	3	sequence4	2	1
3	2	sequence3	1	1
4	1	sequence2	1	1

```

iu@bielinux[~/Desktop/nshare/result]
> sequence3
Length=45853

Score = 2113 bits (1144), Expect = 0.0
Identities = 1232/1276 (97%), Gaps = 0/1276 (0%)
Strand=Plus/Minus

Query 1203879 TTTTACATGCCAACTCGTTACGACAAAGAATTCAAACAAAACATCATCAACCTATATAAA 1203938
          |||
Sbjct 36875 TTTTACATGCCAACTCGTTACGACAAAGAATTCAAACAAAACATTATCAACCTATATAAG 36816

Query 1203939 CAAGGCGAATCAGCCGCCCAACTGGCCAGAGAATATGGCATTGGCTATTCAACCGTTCAT 1203998
          |||
Sbjct 36815 CAAGGCGAATCAGCTGCCCAACTGGCCAGAGAATATGGCATTGGCTATTCAACCGTTCAT 36756

Query 1203999 AAGTGGATCCAGGGCCAAGCCAAAACCTCAATCCGGTAAATCGCCAGACGAAATTAAGCG 1204058
          :
    
```

トータルのヒット数が(1347 + 2 + 1 + 1) = 1351個だったということは、①「Score = 」を含む行数も1351個あるのだろうと予想(スライド215)。grepで確認すべく、qを押してlessから一旦抜ける

W7-6: grep

The image shows a screenshot of a Linux desktop environment. At the top, there is a yellow text box with Japanese text explaining the total number of hits and the goal of using grep. Below this, the BlastViewer application is open, displaying a table of BLAST hits. The table has columns for #, Accession, Definition, Quality, and # HSPs. The first hit (Accession 0) has a quality score of 1347. Below the BlastViewer, a terminal window is open, showing the execution of a grep command on a file named 'sequence3'. The terminal output shows the BLAST results for 'sequence3', including the score, identities, and sequence alignments. A red arrow points to the 'Score = 2113 bits (1144)' line in the terminal output.

BlastViewer Hits Table:

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	1347	1
2	3	sequence4	2	1
3	2	sequence3	1	1
4	1	sequence2	1	1

Terminal Output:

```

> sequence3
Length=45853

Score = 2113 bits (1144), Expect = 0.0
Identities = 1232/1276 (97%), Gaps = 0/1276 (0%)
Strand=Plus/Minus

Query 1203879 TTTTACATGCCAACTCGTTACGACAAAGAATTCAAACAAAACATCATCAACCTATATAAA 1203938
      |||
Sbjct 36875 TTTTACATGCCAACTCGTTACGACAAAGAATTCAAACAAAACATTATCAACCTATATAAG 36816

Query 1203939 CAAGGCGAATCAGCCGCCCAACTGGCCAGAGAATATGGCATTGGCTATTCAACCGTTCAT 1203998
      |||
Sbjct 36815 CAAGGCGAATCAGCTGCCCAACTGGCCAGAGAATATGGCATTGGCTATTCAACCGTTCAT 36756

Query 1203999 AAGTGGATCCAGGGCCAAGCCAAAACCTCAATCCGGTAAATCGCCAGACGAAATTAAGCG 1204058
:
    
```

①「Score = 」を含む行数をgrepで調査。確かに1351個ある(スライド202)

W7-6: grep

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:06午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1* [ 3:06午後 ]
-rwxrwxrwx 1 iu iu 12473982 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 9775990 6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
① iu@bielinux[result] grep -c "Score = " sequence1_blast.txt [ 3:06午後 ]
1351
iu@bielinux[result] █ [ 3:06午後 ]
```


W7-6: grep

①「Score = 」を含む最初の10行分を表示。②のように同じスコアのもの2個ずつ表示されている状態を見て、「sequence1も両末端に重複領域がある環状コンティグなのだろう」と予想する

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1*
-rwxrwxrwx 1 iu iu 12473982 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 9775990 6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] grep -c "Score = " sequence1_blast.txt
1351
iu@bielinux[result] grep "Score = " sequence1_blast.txt | head
Score = 4.228e+06 bits (2289497), Expect = 0.0
Score = 1.162e+04 bits (6294), Expect = 0.0 }
Score = 1.162e+04 bits (6294), Expect = 0.0 }
Score = 1.075e+04 bits (5823), Expect = 0.0 }
Score = 1.075e+04 bits (5823), Expect = 0.0 }
Score = 1.052e+04 bits (5694), Expect = 0.0 }
Score = 1.052e+04 bits (5694), Expect = 0.0 }
Score = 1.043e+04 bits (5647), Expect = 0.0 }
Score = 1.043e+04 bits (5647), Expect = 0.0 }
Score = 9679 bits (5241), Expect = 0.0
iu@bielinux[result]
```



[3:06午後]
[3:06午後]
[3:06午後]
[3:06午後]
[3:06午後]
[3:07午後]

①「Score = 」を含む最初の3行分を表示。②grep -Aオプションで一致した行を含め後ろの3行分を表示。③が今注目しているところ

W7-6: grep

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] grep "Score = " sequence1_blast.txt | head -n 3
Score = 4.228e+06 bits (2289497), Expect = 0.0
Score = 1.162e+04 bits (6294), Expect = 0.0
Score = 1.162e+04 bits (6294), Expect = 0.0
iu@bielinux[result] grep -A 3 "Score = " sequence1_blast.txt | head -n 15
Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
--
Score = 1.162e+04 bits (6294), Expect = 0.0
Identities = 6587/6732 (98%), Gaps = 6/6732 (0%)
Strand=Plus/Plus
-
Score = 1.162e+04 bits (6294), Expect = 0.0
Identities = 6588/6733 (98%), Gaps = 8/6733 (0%)
Strand=Plus/Plus
-
iu@bielinux[result] █
```



[3:31午後]

W7-6: grep -A

①Plus/Plusになっているので、おそらく両端に重複領域があるのだろう。sequence3同士のBlast実行結果(第7回W15-6)も両端でPlus/Plusになっていたことを思い出せば納得できるだろう。本当に両端かどうかはBlastViewerなり、このテキスト形式のBlast結果ファイル(sequence1_blast.txt)を詳細に眺めて確認する

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] grep "Score = "
Score = 4.228e+06 bits (2289497),
Score = 1.162e+04 bits (6294), Expect = 0.0
Score = 1.162e+04 bits (6294), Expect = 0.0
iu@bielinux[result] grep -A 3 "Score = " sequence1_blast.txt | head -n 15
Score = 4.228e+06 bits (2289497), Expect = 0.0
Identities = 2289497/2289497 (100%), Gaps = 0/2289497 (0%)
Strand=Plus/Plus
--
Score = 1.162e+04 bits (6294), Expect = 0.0
Identities = 6587/6732 (98%), Gaps = 6/6732 (0%)
Strand=Plus/Plus
--
Score = 1.162e+04 bits (6294), Expect = 0.0
Identities = 6588/6733 (98%), Gaps = 8/6733 (0%)
Strand=Plus/Plus
--
iu@bielinux[result] █
```



[3:31午後]

W7-6: grep -n

①grep -nで検索文字列(この場合"> sequence")を含む行番号を表示。例えば②sequence4とヒットしたという結果情報は、入力ファイル(sequence1_blast.txt)の205785行目からスタートしていることがわかる。第7回W15-5

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1*
-rwxrwxrwx 1 iu iu 12473982 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 9775990 6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
① iu@bielinux[result] grep -n "> sequence" sequence1_blast.txt
27:> sequence1
205785:> sequence4 ②
206506:> sequence3
206602:> sequence2
iu@bielinux[result] █
```

W7-6: grep -

①grep -nで検索文字列(この場合"Score =")を含む行番号も表示。②赤枠の上下関係(sequence1の最後の③205776とsequence3の④206509)から、sequence4とヒットした領域は2か所あり、いずれもスコア(約10320と8907)がそこそこ高い。スコアの合計(2万弱)とsequence4の長さ(11,372 bp)の関係から、sequence4の大部分の領域がsequence1と類似していると判断できる

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_shar
iu@bielinux[result] ls -l
-rwxrwxrwx 1 iu iu 12473982 6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu 9775990 6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu 2289509 6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu 4579015 6月 13 17:09 sequence1.fq
iu@bielinux[result] grep -n "> sequence" sequence1_blast.txt [ 3:36午後 ]
27:> sequence1
205785:> sequence4
206506:> sequence3
206602:> sequence2
① iu@bielinux[result] grep -n "Score =" sequence1_blast.txt | tail -n 6
205767: Score = 54.7 bits (29), Expect = 2e-04
③ 205776: Score = 54.7 bits (29), Expect = 2e-04
205788: Score = 1.032e+04 bits (5588), Expect = 0.0
206173: Score = 8907 bits (4823), Expect = 0.0 ②
④ 206509: Score = 2113 bits (1144), Expect = 0.0
206605: Score = 124 bits (67), Expect = 1e-25
iu@bielinux[result] █ [ 3:36午後 ]
```


ちなみに

行番号の関係性から、(queryであるsequence1と)
①sequence3の一致が②のスコアであり、(sequence1と)
③sequence2の一致が④のスコアであることがわかる。こんな感じでLinuxコマンドオプションを駆使して、BlastViewerのような可視化ソフトで表示される内容の確認を(私は)行います

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l sequence1*
-rwxrwxrwx 1 iu iu 12473982  6月 21 15:57 sequence1_blast.txt
-rwxrwxrwx 1 iu iu  9775990  6月 22 13:18 sequence1_blast.xml
-rwxrwxrwx 1 iu iu  2289509  6月 13 11:30 sequence1.fa
-rwxrwxrwx 1 iu iu  4579015  6月 13 17:09 sequence1.fq
iu@bielinux[result] grep -n "> sequence" sequence1_blast.txt [ 3:36午後 ]
27:> sequence1
205785:> sequence4
206506:> sequence3
206602:> sequence2
iu@bielinux[result] grep -n "Score = " sequence1_blast.txt | tail -n 6
205767: Score = 54.7 bits (29), Expect = 2e-04
205776: Score = 54.7 bits (29), Expect = 2e-04
205788: Score = 1.032e+04 bits (5588), Expect = 0.0
206173: Score = 8907 bits (4823), Expect = 0.0
206509: Score = 2113 bits (1144), Expect = 0.0
206605: Score = 124 bits (67), Expect = 1e-25
iu@bielinux[result] █ [ 3:36午後 ]
```



Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p189の左上

①今はこのあたり。だんだんマニアックな話になっていくが、Blast結果とアノテーション結果を眺めることで②のようなことが判明することもある、といったことをご自身の研究に活かせると…いいですね

は、BlastViewer を利用する [W7-2]。BlastViewer は Windows 用と Macintosh 用のみが提供されているため、ホスト OS 上でインストールして利用する。XML 形式の BLAST 実行結果ファイル (sequence1_blast.xml) しか受け付けないが、DB 側の配列ごとにヒット数 (配列類似領域数: HSP 数) が示されているなど、全体的な操作感がよい [W7-4]。例えば、sequence1 に対するヒット数が 1,347 個、sequence4 が 2 個、sequence3 と sequence2 がそれぞれ 1 個であったことがわかる。また、ヒット数やスコア分布の全体像を眺めることで、sequence1 にいくつかの重複領域が存在することや、11,372 bp からなる sequence4 の大部分の領域が sequence1 と類似していることなどがわかる [W8-1]。

560027
の全長
W8-5)。

アラインメントされなかった sequence4 の領域 [4853, 5705 bp] に対するアノテーション結果を眺めると、transposase がコードされていたことがわかる (図 1b: W8-5)。これは、該当領域が挿入配列 (insertion sequence; IS) であることを示唆する。これはおそらく、乳酸菌の培養途中で一部の細胞に IS の挿入が起こったためであろう。結果として、シーケンスされた細胞集団の中に IS を含むものと含まないものが混在することになり、sequence4 が独立したコンティグとして出力されたものと思われる。sequence4 は全体的にクオリティスコアが低く (第 7 回 W11-7)、また、後述の Illumina によるシーケンス結果には当該部分が確認できなかったことから、IS が挿入された細胞の存在比率は高くないと考え、sequence4 は除外した。

sequence4 は sequence 1 の一部



BlastViewer で sequence4 (11,372 bp) に対する sequence1 (2,289,497 bp) のヒット領域を眺める [W8]。

第8回原稿p189の左中

は、BlastViewer を利用する [W7-2]。BlastViewer は Windows 用と Macintosh 用のみが提供されているため、ホスト OS 上でインストールして利用する。XML 形式の BLAST 実行結果ファイル (sequence1_blast.xml) しか受け付けないが、DB 側の配列ごとにヒット数 (配列類似領域数 : HSP 数) が示されているなど、全体的な操作感がよい [W7-4]。例えば、sequence1 に対するヒット数が 1,347 個、sequence4 が 2 個、sequence3 と sequence2 がそれぞれ 1 個であったことがわかる。また、ヒット数やスコア分布の全体像を眺めることで、sequence1 にいくつかの重複領域が存在することや、11,372 bp からなる sequence4 の大部分の領域が sequence1 と類似していることなどがわかる [W8-1]。

sequence4 は sequence1 の一部

BlastViewer で sequence4 (11,372 bp) に対する sequence1 (2,289,497 bp) のヒット領域を眺める [W8]。

560027 bp] と、領域 [4853, 5705 bp] を除く sequence4 の全長がほぼ一致していることを意味する (図 1a : W8-5)。

アラインメントされなかった sequence4 の領域 [4853, 5705 bp] に対するアノテーション結果を眺めると、transposase がコードされていたことがわかる (図 1b : W8-5)。これは、該当領域が挿入配列 (insertion sequence; IS) であることを示唆する。これはおそらく、乳酸菌の培養途中で一部の細胞に IS の挿入が起こったためであろう。結果として、シーケンスされた細胞集団の中に IS を含むものと含まないものが混在することになり、sequence4 が独立したコンティグとして出力されたものと思われる。sequence4 は全体的にクオリティスコアが低く (第7回 W11-7)、また、後述の Illumina によるシーケンス結果には当該部分が確認できなかったことから、IS が挿入された細胞の存在比率は高くないと考え、sequence4 は除外した。



①

W8-1 : seq1 vs. seq4

BlastViewerに戻り、①を押してsequence4とのヒット領域を眺める。ここでは、「sequence1 (query側の配列)のどのあたりが、sequence4 (DB側の配列)のどのあたりと一致しているか」を眺めようとしています。思考停止禁止!!



BlastViewer

File Help

BLAST results

sequence1_blast.xml
blastn vs. LH_hgap.fa
sequence1

Summary

Hits

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query

HSP(s)

3

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

11370 11360 11350 11340 11330

HSP: < 1/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 71Mo/123Mo

W8-1 : seq1 vs. seq4

大まかな見方を説明。①3という数字は、②Accessionのところの数字と同じものであり、3'末端などという意味ではない。③DB側配列(sequence4; 11,372 bp)、④query側配列(sequence1; 2,289,497 bp)

BLAST results

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	😊	1347
2	3	sequence4	😊	2
3	2	sequence3	😊	1
4	1	sequence2	😊	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map | Definition | Statistics | Alignment

Query: [Progress bar]

HSP(s): [Red bar]

3: [Black bar]

① ② ③ ④

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

11370 11360 11350 11340 11330

HSP: < 1/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 71Mo/123Mo

W8-1 : seq1 vs. seq4

①DB側配列(sequence4; 11,372 bp)と、② query側配列(sequence1; 2,289,497 bp)は実際の長さは異なるが、長さを揃えて表示している

BLAST results

sequence1_blast.xml
blastn vs. LH_hgap.fa
sequence1

Summary

Hits

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query

HSP(s)

3

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

11370 11360 11350 11340 11330

HSP: < 1/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 71Mo/123Mo

②

①

W8-1 : seq1 vs. seq4

①(sequence4; 11,372 bp)中には、配列類似領域が2つ(HSPが2個)あったことを思い出そう。②をクリックして最初に見ているのは、③スコアが高いほうの「Score = 1.032e+04 bits (5588)」のHSP。④対応するquery側のアラインメント領域

BLAST results

sequence1_blast.xml
blastn vs. LH_hgap.fa
sequence1

Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query: [Progress bar]

HSP(s): [Red bars]

3: [Black bar]

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T
T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T
11370 11360 11350 11340 11330

HSP: < 1/2 >

①2つあるHSPのうち、②最初の1個目という意味。③次のHSPのアラインメントが見られる

W8-1 : seq1 vs. seq4

BLAST results

- sequence1_blast.xml
- blastn vs. LH_hgap.fa
- sequence1

Summary

Hits

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query

HSP(s)

3

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

11370 11360 11350 11340 11330

HSP: < 1/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 71Mo/123Mo

これは、①query側配列(sequence1)が②左から右の方向(Plus鎖)に並んでいるのに対して…

W8-1 : seq1 vs. seq4

BLAST results

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query

HSP(s)

3

555170 555180 555190 555200 555210

T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

4850 4840 4830 4820 4810

HSP: < 2/2 >

Discover KoriBlast to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 82Mo/123Mo

W8-1 : seq1 vs. seq4

これは、①query側配列(sequence1)が②左から右の方向(Plus鎖)に並んでいるのに対して、③DB側配列(sequence4)が④右から左の方向(Minus鎖)に並んでいることから納得できる。テキスト形式のBlast結果ファイル(sequence1_blast.txt)を詳細に眺めて確認してもいいだろう

BLAST results

- sequence1_blast.xml
- blastn vs. LH_hgap.fa
- sequence1

Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map | Definition | Statistics | Alignment

Query: [Progress bar]

HSP(s): [Progress bar]

3: [Progress bar]

555170 555180 555190 555200 555210

T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

4850 4840 4830 4820 4810

T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

HSP: < 2/2 >



①grep実行時に②-nと③-A 3を組み合わせて、“Score = ”と一致した行を含めた後ろの3行分を行番号とともに表示させて、もう少し詳細情報を表示

W8-2: seq1 vs. seq4

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] grep -n -A 3 "Score = " sequence1_blast.txt | tail -n 19
205788: Score = 1.032e+04 bits (5588), Expect = 0.0
205789- Identities = 5656/5684 (99%), Gaps = 23/5684 (0%)
205790- Strand=Plus/Minus
205791-
--
206173: Score = 8907 bits (4823), Expect = 0.0
206174- Identities = 4851/4862 (99%), Gaps = 11/4862 (0%)
206175- Strand=Plus/Minus
206176-
--
206509: Score = 2113 bits (1144), Expect = 0.0
206510- Identities = 1232/1276 (97%), Gaps = 0/1276 (0%)
206511- Strand=Plus/Minus
206512-
--
206605: Score = 124 bits (67), Expect = 1e-25
206606- Identities = 73/76 (96%), Gaps = 0/76 (0%)
206607- Strand=Plus/Minus
206608-
iu@bielinux[result] █
```

[5:59午後]

sequence4上の配列類似領域の①1つめと②2つめ。
③確かにストランド情報として、query側(sequence1)がPlus鎖、DB側(sequence4)がMinus鎖となっている

W8-2: seq1 vs. seq4

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] grep -n -A 3 "Score = " sequence1_blast.txt | tail -n 19
205788: Score = 1.032e+04 bits (5588), Expect = 0.0
205789- Identities = 5656/5684 (99%), Gaps = 23/5684 (0%)
205790- Strand=Plus/Minus
205791-
--
206173: Score = 8907 bits (4823), Expect = 0.0
206174- Identities = 4851/4862 (99%), Gaps = 11/4862 (0%)
206175- Strand=Plus/Minus
206176-
--
206509: Score = 2113 bits (1144), Expect = 0.0
206510- Identities = 1232/1276 (97%), Gaps = 0/1276 (0%)
206511- Strand=Plus/Minus
206512-
--
206605: Score = 124 bits (67), Expect = 1e-25
206606- Identities = 73/76 (96%), Gaps = 0/76 (0%)
206607- Strand=Plus/Minus
206608-
iu@bielinux[result] █
```

[5:59午後]

①1つめのHSP (以下、HSP1)に切り替えて、②Alignmentタブをクリック。ここでアラインメント結果の詳細情報がわかる

W8-3:HSP1

The screenshot shows the BlastViewer interface. On the left, a sidebar lists BLAST results: sequence1_blast.xml, blastn vs. LH_hgap.fa, and sequence1. The main window is divided into several sections:

- Summary:** Contains a 'Hits' table with the following data:

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1
- Alignment:** Shows 'Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)'. Below this are tabs for 'HSP Map', 'Definition', 'Statistics', and 'Alignment'. The 'Alignment' tab is active, displaying:
 - Query: from 549,494 to 555,171, strand '+'
 - 3: from 11,372 to 5,706, strand '-'
 - length: 5,684
 A sequence alignment is shown with two lines of nucleotides (T, C, A, G) and vertical bars indicating matches. The top line is labeled with coordinates 549500, 549510, 549520, 549530, 549540. The bottom line is labeled with coordinates 11370, 11360, 11350, 11340, 11330.
- Navigation:** At the bottom, there are navigation buttons for HSPs, including a left arrow, a '1/2' indicator, and a right arrow. A red arrow labeled '1' points to the left arrow.

At the bottom of the window, there is a footer with 'Welcome to BlastViewer', a URL 'www.korilog.com', and a file size indicator '99Mo/123Mo'. A page number '248' is visible in the bottom right corner.

①query側(sequence1)の549,494番目と、②DB側(sequence4)の11,372番目から…③を一番右まで移動

W8-3:HSP1

BLAST results

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Def **①** Statistics Alignment

Query: from 549,494 to 555,171, strand '+'
 3: from 11,372 to 5,706, strand '-'

length: 5,684 **②**

549500 549510 549520 549530 549540

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

T T T C C A G T G C T G G T G T A T A A A C G G C A A T C G G T T T A A T C G C T G A T C C T G G T T

11370 11360 11350 11340 11330

HSP: **③** < 1/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer www.korilog.com 99Mo/123Mo

W8-3:HSP1

①query側(sequence1)の555,171番目と、②DB側(sequence4)の5,706番目の領域がHSP1。③を一番右まで移動させた結果で見えています

The screenshot shows the BlastViewer interface. On the left, the 'BLAST results' pane lists 'sequence1_blast.xml', 'blastn vs. LH_hgap.fa', and 'sequence1'. The main area displays the 'Summary' and 'Hits' table.

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

The 'Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)' section shows the 'HSP Map' tab selected. It displays the alignment coordinates and sequence:

Query: from 549,494 to 555,171, strand '+'
3: from 11,372 to 5,706, strand '-'
length: 5,684

The alignment sequence is shown as two rows of nucleotides with vertical bars indicating matches. The sequence is: C G C C T G G G T G T A A A C C C A A C G C A G T T A C A T T T T C T T G G G G A C C C A T T A A G A. The alignment is shown from position 5750 to 5710.

At the bottom, the 'HSP:' section shows navigation buttons and the current HSP number '1/2'. The status bar at the bottom right shows 'www.korilog.com' and '87Mo/123Mo'.

W8-4:HSP2

BLAST results

sequence1_blast.xml
blastn vs. LH_hgap.fa
sequence1

Summary

Hits

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)

HSP Map Definition Statistics Alignment

Query: from 555,167 to 560,027, strand '+'
3: from 4,852 to 1, strand '-'
length: 4,862

```

555170          555180          555190          555200          555210
T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T
4850          4840          4830          4820          4810
    
```

HSP: < 2/2 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer

www.korilog.com 68Mo/123Mo

①query側(sequence1)の555,167番目と、②DB側(sequence4)の4,852番目から…③を一番右まで移動

W8-4:HSP2

The screenshot shows the BlastViewer interface with the following components:

- Left Panel:** BLAST results summary including 'sequence1_blast.xml', 'blastn vs. LH_hgap.fa', and 'sequence1'.
- Summary Panel:** A table of hits with columns for #, Accession, Definition, Quality, and # HSPs.
- Alignment Panel:** Shows 'Alignment: Query (2289497 nuc) vs. 3 (11,372 nuc)'. It includes tabs for HSP Map, Def, Statistics, and Alignment. The query sequence is from 555,167 to 560,027, and the subject sequence is from 4,852 to 1. The alignment length is 4,862.
- Sequence View:** Displays two DNA sequences aligned. The top sequence (Query) starts at position 555170 and ends at 555210. The bottom sequence (Subject) starts at position 4850 and ends at 4810. Red arrows point to the start of the alignment (1), the subject start position (2), and the HSP control (3).
- Bottom Panel:** Includes a 'Discover KoriBlast' link, a 'Welcome to BlastViewer' message, a URL 'www.korilog.com', and a date '68Mo/123Mo'.

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Query: from 555,167 to 560,027, strand '+'
3: from 4,852 to 1, strand '-'
length: 4,862

555170 555180 555190 555200 555210

T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

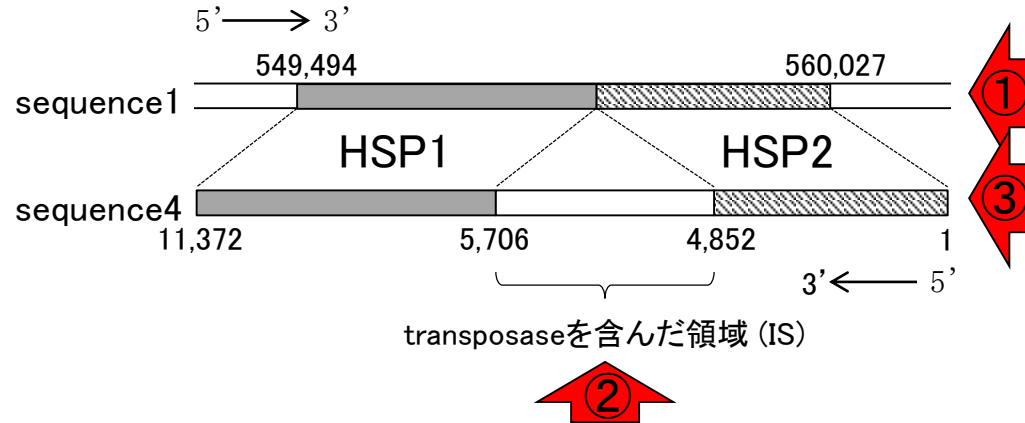
T A A G A G T T C C A T A C T T T T G A C T G T T T C A G T A C C C A T G T C A C C A T G T G T A A T

4850 4840 4830 4820 4810

HSP: < 2/2 >

W8-5: 模式図

sequence1とsequence4のアラインメント模式図。① sequence1の一続きの領域[549494, 560027 bp]と、②領域[4853, 5705 bp]を除く、③sequence4の全長がほぼ一致。そして、④アラインメントされなかった領域[4853, 5705 bp]には、transposaseがコードされていた(W4-9)。第8回の図1と同じ



sequence4	4133..4753	CDS	mannose-specific PTS system IID component
sequence4	4781..4873	CDS	mannose-specific PTS system IIA component
sequence4	4912..5220	CDS	transposase
sequence4	5361..5699	CDS	transposase
sequence4	5711..6058	CDS	mannose-specific PTS system IIA component
sequence4	6114..6596	CDS	mannose/fructose/sorbose-specific PTS system IID component

第8回原稿p189の右上

①このあたりまでの話が終了。
sequence4除外後、残るはsequence1
の詳細な解析。②sequence1は(ちよつ
とややこしいが)環状染色体、が結論

BLAST 実行結果ファイル (sequence1_blast.xml) しか受け付けないが、DB 側の配列ごとにヒット数 (配列類似領域数; HSP 数) が示されているなど、全体的な操作感がよい [W7-4]。例えば、sequence1 に対するヒット数が 1,347 個、sequence4 が 2 個、sequence3 と sequence2 がそれぞれ 1 個であったことがわかる。また、ヒット数やスコア分布の全体像を眺めることで、sequence1 にいくつかの重複領域が存在することや、11,372 bp からなる sequence4 の大部分の領域が sequence1 と類似していることなどがわかる [W8-1]。

sequence4 は sequence1 の一部

BlastViewer で sequence4 (11,372 bp) に対する sequence1 (2,289,497 bp) のヒット領域を眺める [W8]。スコアの高いほう (Score = 10,320) の 1 つめの HSP (以下、HSP1) は、sequence1 の [549494, 555171 bp] と、sequence4 の [11372, 5706 bp] の領域から形成されてい

アラインメント [W8-1] に対するアノテーション結果を眺めると、transposase がコードされていたことがわかる (図 1b: W8-5)。これは、該当領域が挿入配列 (insertion sequence; IS) であることを示唆する。これはおそらく、乳酸菌の培養途中で一部の細胞に IS の挿入が起こったためであろう。結果として、シーケンスされた細胞集団の中に IS を含むものと含まないものが混在することになり、sequence4 が独立したコンティグとして出力されたものと思われる。sequence4 は全体的にクオリティスコアが低く (第 7 回 W11-7)、また、後述の Illumina によるシーケンス結果には当該部分が確認できなかったことから、IS が挿入された細胞の存在比率は高くないと考え、sequence4 は除外した。

① sequence1 はプロファージ領域を含む環状染色体 ②

BlastViewer で sequence1 (2,289,497 bp) に対する

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p187

①第8回原稿のアブストラクト。特にsequence1の解釈は赤下線部分を実感できます。得られた結果をパズルゲームのように組み合わせて合理的な結論を導き出す考え方は、バクテリア以外の分野のヒトにとっても有意義かも。取捨選択してご自身の研究に活かしてください

de novo ゲノムアセンブリ結果から、概要・完全配列 (draft and complete genome sequences) にする作業は、基本的な塩基配列解析用プログラムの活用や自作、プログラム実行結果の検証や合理的な解釈など、ウェットとドライ両面の幅広い知識とスキル、そして精神力を要する。第8回は、PacBio データの *de novo* ゲノムアセンブリの後処理として、特に染色体ゲノムに相当する長いコンティグの検証作業を解説する。具体的には、DFAST による乳酸菌に特化したアノテーション、BLAST の実行と可視化、環状染色体の完成、Illumina データのマッピングによる検証と修正について述べる。ウェブサイト (R で) 塩基配列解析 (URL: http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html) 中に本連載をまとめた項目 (URL: http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#about_book_JSLAB) が存在する。ウェブ資料 (以下、W) や関連ウェブサイトなどを効率的に活用してほしい。

W9-1 : seq1 vs. seq1

query側(sequence1)は固定で、①DB側が sequence1のBlast結果を表示。②配列類似領域(HSP)は1,347個あったことを思い出そう。③スコアトップのHSP (HSP1)が、④seq1 vs. seq1の全長が100%一致のHSPとなるのは当たり前

The screenshot shows the BlastViewer interface. On the left, the 'BLAST results' panel lists 'sequence1_blast.xml', 'blastn vs. LH_hgap.fa', and 'sequence1'. The main window is divided into 'Summary' and 'Alignment' sections.

Summary: A table of hits is shown:

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: The 'Alignment' tab is active, showing the HSP Map for 'Query (2289497 nuc) vs. 0 (2,289,497 nuc)'. The alignment details are:

```

Query: from 1 to 2,289,497, strand '+'
      0: from 1 to 2,289,497, strand '+'
length: 2,289,497
    
```

Below the text, a sequence alignment is displayed with positions 60, 70, 80, 90, and 100 marked. The sequences are identical: G C G G T G G C A A T C G C G C T G A C A G A T T T A C G C T C A A A G G A A A C C A T G A T G A T G C.

At the bottom, the 'HSP:' section shows '1/1347' with navigation arrows.

Annotations: Red arrows with numbers 1-4 point to specific elements: 1 points to 'sequence1' in the hits table; 2 points to the '# HSPs' column value '1347'; 3 points to the sequence alignment; 4 points to the alignment text box.

W9-1 : seq1 vs. seq1

BLAST results

- sequence1_blast.xml
- blastn vs. LH_hgap.fa
- sequence1

Summary

#	Accession	Definition	Quality	# HSPs
1	0	sequence1	☺	1347
2	3	sequence4	☺	2
3	2	sequence3	☺	1
4	1	sequence2	☺	1

Alignment: Query (2,289,497 nuc)

HSP Map | Definition | Statistics | Alignment

Query: from 1 to 2,289,497, strand '+'
0: from 1 to 2,289,497, strand '+'
length: 2,289,497

```
      60      70      80      90     100
G C G G T G G C A A T C G C G C T G A C A G A T T T A C G C T C A A A G G A A A C C A T G A T G A T G C
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
G C G G T G G C A A T C G C G C T G A C A G A T T T A C G C T C A A A G G A A A C C A T G A T G A T G C
      60      70      80      90     100
```

HSP: < 1/1347 >

Discover KoriBlast
to go beyond the viewer

Welcome to BlastViewer www.korilog.com 77Mo/123Mo

W9-2: HSP1-33

これです。①括弧内のHSPは、カッコ外のHSPとqueryとDB側的一致領域が入れ替わっているだけで、実質的に同じものです。この表作成は1つ1つ確認しながらの手作業

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,900	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP4 (HSP5)	10,754	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP8 (HSP9)	10,429	2,283,820	2,289,497	plus	5,839	11,509	plus	5,679	②'	②
HSP10 (HSP11)	9,679	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP14 (HSP15)	9,362	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106		
HSP16 (HSP17)	8,385	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,301	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP22 (HSP23)	6,047	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365		
HSP24 (HSP26)	6,043	999,670	1,002,969	plus	711,886	715,184	plus	3,300		
HSP25 (HSP27)	6,043	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294		
HSP28 (HSP29)	6,038	2,275,808	2,279,100	plus	1,091,466	1,088,173	minus	3,295		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		



W9-2:HSP1-33

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,900	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP4 (HSP5)	10,754	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP24 (HSP26)	6,043	999,670	1,002,969	plus	711,886	715,184	plus	3,300		
HSP14 (HSP15)	9,362	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106		
HSP25 (HSP27)	6,043	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294		
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP16 (HSP17)	8,385	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,301	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		
HSP10 (HSP11)	9,679	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301		
HSP22 (HSP23)	6,047	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365		
HSP28 (HSP29)	6,038	2,275,808	2,279,100	plus	1,091,466	1,088,173	minus	3,295		
HSP8 (HSP9)	10,429	2,283,820	2,289,497	plus	5,839	11,509	plus	5,679	②'	②



W9-2:HSP1-33

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,901	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP4 (HSP5)	10,741	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP24 (HSP26)	6,045	999,670	1,002,969	plus	711,886	715,184	plus	3,300		
HSP14 (HSP15)	9,362	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106		
HSP25 (HSP27)	6,043	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294		
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP16 (HSP17)	8,385	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,301	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		
HSP10 (HSP11)	9,679	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301		
HSP22 (HSP23)	6,047	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365		
HSP28 (HSP29)	6,031	2,275,808	2,279,100	plus	1,091,466	1,088,173	minus	3,295		
HSP8 (HSP9)	10,412	2,283,820	2,289,497	plus	5,839	11,509	plus	5,679	②'	②



W9-2: HSP1-33

右側の①' が領域[37329, 43187 bp]、①が[1, 5860 bp]、②' が [2283820, 2289497 bp]、②が[5839, 11509 bp]に相当。この「①' , ①, ②' , ②」が後の議論対象

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,900	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP4 (HSP5)	10,754	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP24 (HSP26)	6,043	999,670	1,002,969	plus	711,886	715,184	plus	3,300		
HSP14 (HSP15)	9,362	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106		
HSP25 (HSP27)	6,043	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294		
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP16 (HSP17)	8,385	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,301	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		
HSP10 (HSP11)	9,679	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301		
HSP22 (HSP23)	6,047	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365		
HSP28 (HSP29)	6,038	2,275,808	2,279,100	plus	1,091,466	1,088,173	minus	3,295		
HSP8 (HSP9)	10,429	2,283,820	2,289,497	plus	5,839	11,509	plus	5,679	②'	②

③のあたりはribosomal RNAがほとんど。DFAST
で得られたアノテーション情報を追加しています

W9-3: アノテーション

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,900	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP4 (HSP5)	10,754	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP24 (HSP26)	6,041	999,670	1,002,969	plus	711,886	715,184	plus	3,300	ribosomal RNA	
HSP14 (HSP15)	9,300	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106	ribosomal RNA	
HSP25 (HSP27)	6,041	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294	ribosomal RNA	
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP16 (HSP17)	8,385	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,301	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		
HSP10 (HSP11)	9,671	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301	ribosomal RNA	
HSP22 (HSP23)	6,035	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365	ribosomal RNA	
HSP28 (HSP29)	6,035	2,275,808	2,279,100	plus	1,091,466	1,088,173	minus	3,295	ribosomal RNA	
HSP8 (HSP9)	10,429	2,283,820	2,289,497	plus	5,839	11,509	plus	5,679	②'	②

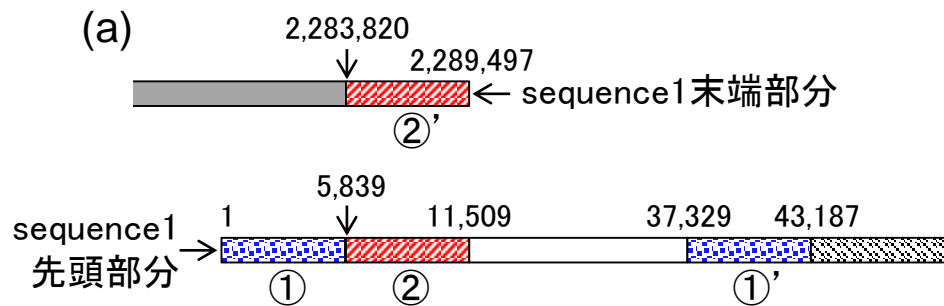
W9-3: アノテーション

④のあたりは、「adhesion exoprotein、mucus-binding protein、hypothetical protein」となっており、はっきり言ってよくわからない

HSP番号	スコア	一致領域(query側)			一致領域(DB側)			Length	query側の領域	DB側の領域
		start	end	strand	start	end	strand			
HSP1	4,227,900	1	2,289,497	plus	1	2,289,497	plus	2,289,497		
HSP4 (HSP5)	10,754	37,329	43,187	plus	1	5,860	plus	5,865	①'	①
HSP24 (HSP26)	6,043	999,670	1,002,969	plus	711,886	715,184	plus	3,300	ribosomal RNA	
HSP14 (HSP15)	9,362	1,088,170	1,093,274	plus	999,673	1,004,778	plus	5,106	ribosomal RNA	
HSP25 (HSP27)	6,043	1,088,173	1,091,466	plus	711,892	715,184	plus	3,294	ribosomal RNA	
HSP2 (HSP3)	11,624	2,058,465	2,065,193	plus	2,057,850	2,064,578	plus	6,732		
HSP6 (HSP7)	10,516	2,059,080	2,065,193	plus	2,057,850	2,063,963	plus	6,117		
HSP12 (HSP13)	9,524	2,059,672	2,065,193	plus	2,057,830	2,063,348	plus	5,525		
HSP16 (HSP17)	8,388	2,060,310	2,065,194	plus	2,057,850	2,062,735	plus	4,887		
HSP18 (HSP19)	7,366	2,060,901	2,065,194	plus	2,057,830	2,062,120	plus	4,296		
HSP20 (HSP21)	6,186	2,061,516	2,065,193	plus	2,057,830	2,061,504	plus	3,680		
HSP30 (HSP31)	5,050	2,062,154	2,065,193	plus	2,057,850	2,060,889	plus	3,042		
HSP32 (HSP33)	4,021	2,062,769	2,065,197	plus	2,057,850	2,060,279	plus	2,431		
HSP10 (HSP11)	9,679	2,273,804	2,279,100	plus	717,190	711,892	minus	5,301	ribosomal RNA	
HSP22 (HSP23)	6,047	2,275,808	2,279,168	plus	1,002,969	999,608	minus	3,365	ribosomal RNA	
HSP28 (HSP29)	6,038	2,275,808		sequence1	2055735..2060864	CDS		adhesion exoprotein		NA
HSP8 (HSP9)	10,429	2,283,820		sequence1	2060942..2063323	CDS		mucus-binding protein		②
				sequence1	2063401..2066928	CDS		hypothetical protein		



W10-1 : sequence1

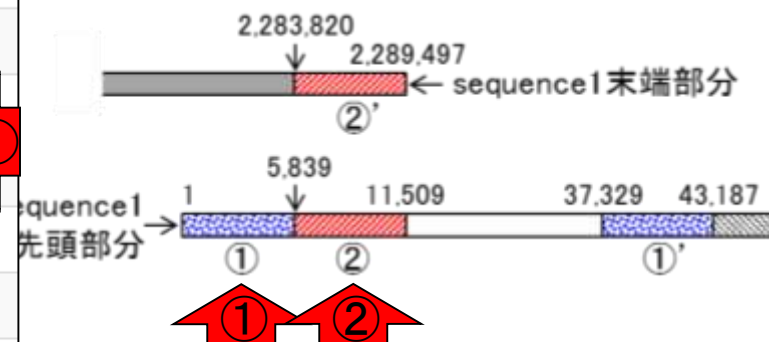


sequence1の末端付近の重複は、HSP8の一致領域(②と②')に相当。これにHSP4の一致領域(①と①')を含めた模式図。第8回の図2aと同じ。後の検証により、①の領域は染色体上に存在しないことが確認された

W10-2: アノテーション

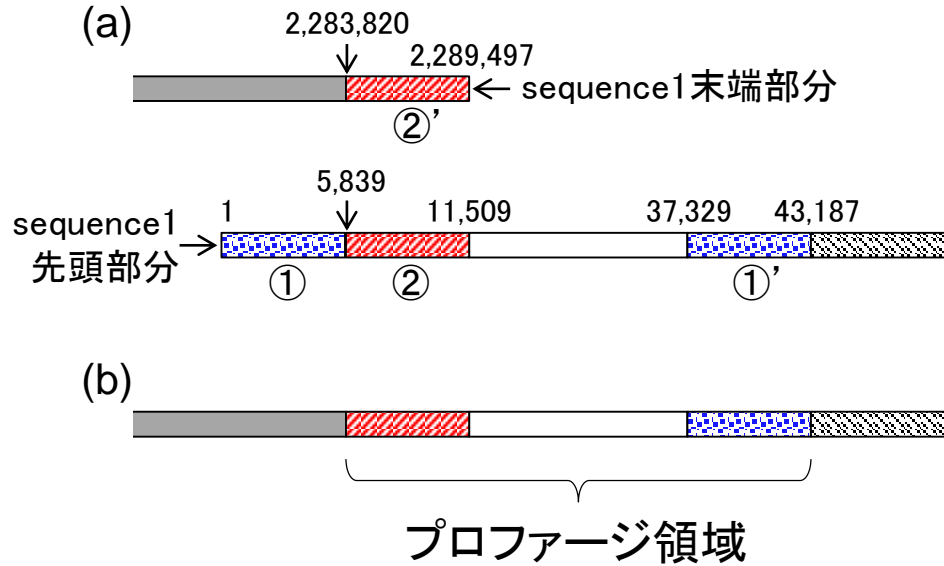
HSP4の一致領域①[1, 5860 bp]のDFASTアノテーション。HSP8の一致領域②[5839, 11509 bp]のアノテーションの一部。マニアックすぎるので、このあたりは飛ばします。第8回ウェブ資料に詳細情報あり

LOCUS_00001	sequence1	151..384	CDS	hypothetical protein	
LOCUS_00002	sequence1	350..886	CDS	hypothetical protein	
LOCUS_00003	sequence1	883..1311	CDS	hypothetical protein	
LOCUS_00004	sequence1	1637..1849	CDS	hypothetical protein	
LOCUS_00005	sequence1	1968..2165	CDS	hypothetical protein	
LOCUS_00006	sequence1	2355..2732	CDS	prophage protein	
LOCUS_00007	sequence1	2725..2940	CDS	hypothetical protein	①
LOCUS_00008	sequence1	2930..3031	CDS	hypothetical protein	
LOCUS_00009	sequence1	3067..3534	CDS	holin	
LOCUS_00010	sequence1	3547..4578	CDS	1,4-beta-N-acetylmuramidase	
LOCUS_00011	sequence1	4765..4935	CDS	hypothetical protein	
LOCUS_00012	sequence1	4936..5517	CDS	hypothetical protein	
LOCUS_00013	sequence1	complement (6059..7228)	CDS	integrase	
LOCUS_00014	sequence1	complement (7407..8267)	CDS	hypothetical protein	②
LOCUS_00015	sequence1	complement (8301..8813)	CDS	hypothetical protein	



W10-3: 染色体構造

(b) sequence1の両末端である①と②'の領域をトリム。得られた領域[5839, 2283819 bp]の両末端を結合した環状コンティグが実際の染色体構造であると予想した



第8回原稿p190右上

sequence1 (2,289,497 bp) のヒット領域を眺める [W9]。最もスコアの高い HSP (HSP1) は sequence1 同士の全長が 100% 一致のものであるため、2 番目の HSP (HSP2) 以降のアラインメントが精査対象となる。総ヒット数 (1,347 個) が非常に多いため、ここではスコアが 4,000 以上という条件を満たす HSP1-33 の一致領域をまとめた [W9-2]。ゲノム配列決定論文¹⁾ 中では、結論として下記の計 4 領域が議論の対象となっているが、どこまでの HSP を眺め、どのように解釈して結論づけるかは、利用可能な情報を徹底的に調べ、試行錯誤しながら整理していく以外にはないだろう。

- ・ HSP4 (HSP5) の一致領域① [1, 5860 bp] と①' [37329, 43187 bp]
- ・ HSP8 (HSP9) の一致領域② [5839, 11509 bp] と②' [2283820, 2289497 bp]

上記以外の HSP のアノテーションとして、HSP10, 14, 22, 24, 25, 28 の領域には、ribosomal RNA がコードされ

改めて強調しておきたい点は、配列相同性とアノテーションの併用の重要性である。BLAST 結果だけではわからないこともある。しかし、どのような遺伝子がコードされているかなど、アノテーション情報と合わせて総合的に判断すればわかる場合もある。また、シーケンス対象サンプルは均一な集団ではない点も胆に銘じておかねばならない。実際、今回の乳酸菌サンプル中には、図 2b で示すようなプロファージ領域を含む環状染色体だけでなく、図 2c で示すような (i) プロファージ領域が切り出されてきた環状ファージ DNA や、(ii) プロファージ領域が切り出されてなくなった残りの環状染色体も含まれていた。このあたりが、本稿の冒頭で述べた幅広い知識や合理的な解釈に相当する。



① 乳酸菌ゲノム概要配列の作成

ここまで *de novo* アセンブリ結果ファイル (LH_hgap.fa) を入力として、アノテーションや配列相同性検索を行った。得られた方針は下記の通りである：

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



第8回原稿p190右中

・ HSP8 (HSP9) の一致領域② [5839, 11509 bp] と②' [2283820, 2289497 bp]

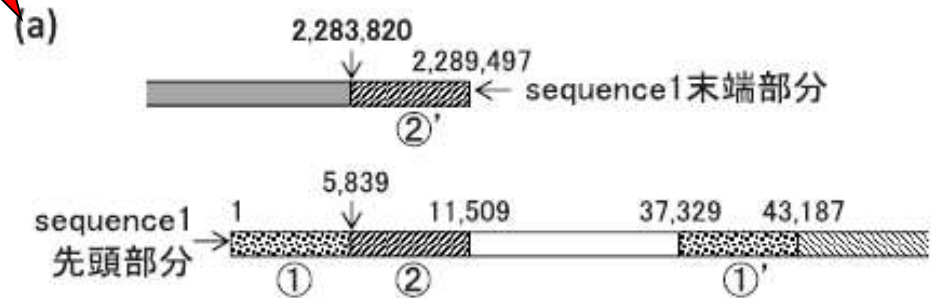
上記以外の HSP のアノテーションとして、HSP10, 14, 22, 24, 25, 28 の領域には、ribosomal RNA がコードされていた。また、残りの HSP2, 6, 12, 16, 18, 20, 30, 32 は、[2057850, 2065197 bp] にかけての反復構造を含んだ領域の中に全て含まれ、この中には3つの遺伝子 (adhesion exoprotein, mucus-binding protein, and hypothetical protein) がコードされていた [W9-3]。遺伝子名からは細胞接着に関わる表層タンパクと推察され、ここに見られた反復構造が何らかの働きを持っているのかもしれない。いずれも sequence1 末端から 10,000 塩基以上離れた領域であることから、アセンブリ結果の検証という点では無関係であろう。

sequence1 末端付近の重複は、HSP8 の一致領域 (②と②') に相当する。これに HSP4 の一致領域 (①と①') を含めた模式図を示す (図 2a; W10-1)。もし①の領域がなければ、sequence2 や 3 と同様に「両末端の重複 → 環状コンテナ」と結論づけられる。現在利用可能な他の情

乳酸菌ゲノム概要配列の作成

ここまで *de novo* アセンブリ結果ファイル (LH_hgap.fa) を入力として、アノテーションや配列相同性検索を行った。得られた方針は下記の通りである：

sequence1 : 環状染色体 (図 2)。主に①や②' の重複部分を除けばよい。ここでは、W10-4 のアラインメントを参



第8回原稿p191左上

考にして、領域 [5839, 2283819 bp] を抽出する。
 sequence2: 環状プラスミド [W5-1]。BLAST で重複領域のアラインメントをとり、領域 [2641, 84270 bp] を抽出する [W11-1]。
 sequence3: 環状プラスミド。領域 [2450, 43422 bp] を抽出する (第7回 W16)。
 sequence4: sequence1 の一部なので却下 (図1)。

この方針に従って重複除去を行い、概要配列 (LH_draft.fa) を作成する [W11-2]。どのレベルを「概要 (ドラフト)」と呼ぶかはヒトそれぞれであるが、少なくとも重複除去でおしまいではないため、ここでは重複除去後の状態を概要と呼ぶ。

概要配列への MiSeq データのマッピング

乳酸菌ゲノム配列決定の原著論文¹⁾では、概要配列 (LH_draft.fa) の元となった PacBio データ (DRR054113) 以外に、paired-end Illumina MiSeq データ (DRR024501) も存在

DDBJ Pipeline 上での BWA 実行時のオプションは、基本的にデフォルト設定のままでよい [W14-5]。補足として、オプション画面の Step3 (ユニーク化処理) は、ペアのリードがともにリファレンス配列の 1 か所にのみマップされたリード (uniquely mapped read; unique mapper) を残し、それ以外のリードを除去している。これは、反復配列のような領域が存在すると、1つのリードが複数個所にマップされて解析結果の解釈が難しくなるからである。変異解析では、これらのリードは除外して行う場合が多い。

BWA (ver. 0.6.1-r104) 実行結果として、297,633 リード中 281,303 個 (94.513%) がマップされた [W15-2]。リファレンス配列 (LH_draft.fa) のゲノムサイズは 2,400,584 bp であるが、そのうち 2,400,552 bp 分がマップされたリードで覆われていた。つまり被覆率 (coverage) は、 $2,400,552 / 2,400,584 = 99.99867\%$ である。また、マップされたリードの総塩基数 (130,565,653 bp) をマップされたリードで覆われている領域 (2,400,552 bp) で割れば、平均してどれだけの厚み (depth) でマップされているかがわかる。この場合は、depth = 54.390 である。



W11-2:ドラフト配列作成

①アセンブリ結果ファイル(LH_hgap.fa)を入力として、重複除去を行う一連のコマンド。
②出力ファイルはLH_draft.fa。③ファイルサイズの減少度合い的に妥当。ここでは示さないがlessで配列の最初と最後を狙い通りに抽出できているか確認しておこう

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa
-rwxrwxrwx 1 iu iu 2433662  6月 12 15:25 LH_hgap.fa
iu@bielinux[result] echo ">chromosome" > LH_draft.fa [ 3:50午後]
iu@bielinux[result] head -2 LH_hgap.fa | tail -1 | cut -c 5839-2283819 >> LH_draft.fa [ 3:50午後]
iu@bielinux[result] echo ">plasmid1" >> LH_draft.fa [ 3:50午後]
iu@bielinux[result] head -4 LH_hgap.fa | tail -1 | cut -c 2641-84270 >> LH_draft.fa [ 3:50午後]
iu@bielinux[result] echo ">plasmid2" >> LH_draft.fa [ 3:50午後]
iu@bielinux[result] head -6 LH_hgap.fa | tail -1 | cut -c 2450-43422 >> LH_draft.fa [ 3:50午後]
iu@bielinux[result] ls -l LH*.fa [ 3:50午後]
-rwxrwxrwx 1 iu iu 2400619  6月 27 2017 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662  6月 12 15:25 LH_hgap.fa
iu@bielinux[result] █
```



Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



この後の展開(第8回)

①自分は変異解析をやるつもりがなく、VCFファイルの理解はこれまで拒否していました。今回ドラフト配列の修正という局面で用いることになり、嫌々ながら必要最小限の勉強をしました

■ W12: bwaでマッピング

- (Bio-Linuxにプレインストールされている)bwaを利用し、PacBioデータから得られたドラフト配列をリファレンスとして、Illumina MiSeqデータをマップ。
- やり方のみを示し、この結果は使わない。後述のDDBJ Pipelineの結果のほうがVCFファイルまで出力してくれて便利だから。

■ W13: CyberduckのインストールとMiSeqデータのアップロード

- CyberduckはFTPソフト。DDBJ Pipelineへのデータアップロード用として利用。

■ W14: DDBJ Pipeline上でbwaを実行

■ W15: 実行結果の概観とダウンロード

■ W16~W17: マッピング結果ファイル(SAM/BAM形式)の説明

■ W18: VCFファイルの説明

- PacBioデータから得られたリファレンス配列と違っている部分を示したものの。リード数およびクオリティスコアの点でIllumina MiSeqデータのほうが優位。VCFファイルの結果をもとに手作業でドラフト配列を修正するための基礎知識や基本的な考え方を身につけることを目的として、VCFファイルの読み取り方を解説。VCF形式については、(平成27-28年度のNGSハンズオン講習会の)変異解析」でも紹介。



この後の展開(第8回)

① Tabletの解説も兼ねています。W12からW27の範囲は、PacBioデータしかない場合は行いません。MiSeqデータもあるので援用しています

■ W19: Tabletのインストールと利用

- Tabletは、リファレンス配列へのマッピング結果をローカル環境で可視化するViewer。**VCFファイルでみた変異箇所は、ドラフト配列の修正候補箇所に相当する。** MiSeqデータのマッピング結果のほうを採用するかどうかを、Tabletで修正候補箇所を実際に眺めて判断する。

■ W20: plasmid2上の変異箇所を確認

■ W21: plasmid1上の変異箇所を確認

■ W22: chromosome上の変異箇所を確認

■ W23: マップされたリードの詳細情報を知るTips

■ W24: ドラフト配列修正方針のまとめとVCFファイルのおさらい

■ W26: 変異の反映…が意外と文字列置換でうまくいかないことを学ぶ

■ W27: テキストエディタ(EmEditor)を用いてベタで修正

- 文字列検索と該当塩基位置情報を組み合わせて、ドラフト配列を修正



この後の展開(第9回)

連載第9回以降は、Illumina MiSeqデータの有無に関係なく行う。①は、環状バクテリアゲノムの場合にしばしば慣例として行われる。GC skewで眺めたときに、時計の0時あたりで分布が切り替わるようにするのが主目的

- W3: Ori-Finderで転写開始点を同定
- W4: 配列の“回転”
 - *dnaA*遺伝子が配列の先頭となるように“回転”
- W5: INSDCフラットファイルの説明
 - entry, feature, qualifierからなる3つの階層構造の話など
- W6: locus_tagの説明
- W7とW8: DFASTアノテーションの実行と結果の確認、Tips
- W9とW10: DFAST結果画面からのDDBJへの登録の概要
- W11とW12: 描画ソフトDNAPlotterのインストールと利用
- W13: grepを駆使してgenbank(.gbk)形式を理解
- W14: awkコマンドの利用(最初の配列のみ抽出)
- W15とW16: DNAPlotterを再度利用(プラスミド配も描画し)
- W17: 描画結果を原著論文の図と比較
- W18: ovaファイルの再インストール関連情報
- W19: 配列情報つきgffファイルを作成してDNAPlotterで描画



連載第10回は、*de novo*アセンブリ結果ファイルをDFASTアノテーション結果とともに、DDBJに登録する一連の作業を解説

この後の展開(第10回)

- W2: DDBJの登録用アカウント作成
- W3: DDBJにログイン
- W4: NSSSとMSSの説明など(今回はMSSでの登録)
- W5とW6: BioProjectの登録。Locus tag prefixのおさらいなど
- W7とW8: BioSampleの登録。サンプル属性のテンプレートファイルの解説
- W9: アクセッション番号取得とDDBJによる編集結果の確認
- W10: MSS (Mass Submission System)での申込
- W11: DFAST上でDDBJ登録用ファイルを作成
- W12: DDBJとのメールのやりとり
- W13: 公開

Contents (主に第8回)

- W4: ゲノムアノテーション (DFAST)
- W5: dotterの実行 (sequence4)
- W6: Blastの実行 (DB配列はLH_hgap.fa、query配列はsequence1)
- W7: BlastViewerでBlast実行結果を眺める (W7-4まで)
- W7: lessとgrepでも確認 (W7-6まで)
- W8: Blast実行結果のsequence1 vs. sequence4を眺める
- W9~W10: sequence1を詳細に調べる (Blastとアノテーション結果を併用)
- W11: 乳酸菌ゲノム概要配列の作成
- この後の展開 (第8回のW12以降と第9-10回の概要)
- W3: シェルスクリプト



W3-0: ファイル分割1

①第8回W11-2(スライド273)では、3つの配列からなるmulti-FASTAファイル(LH_draft.fa)を作成した。よって、②配列数は3、③行数は6

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:48午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] echo ">chromosome" > LH_draft.fa [ 3:50午後 ]
iu@bielinux[result] head -2 LH_hgap.fa | tail -1 | cut -c 5839-2283819 >> LH_draft.fa [ 3:50午後 ]
① iu@bielinux[result] echo ">plasmid1" >> LH_draft.fa [ 3:50午後 ]
iu@bielinux[result] head -4 LH_hgap.fa | tail -1 | cut -c 2641-84270 >> LH_draft.fa [ 3:50午後 ]
iu@bielinux[result] echo ">plasmid2" >> LH_draft.fa [ 3:50午後 ]
iu@bielinux[result] head -6 LH_hgap.fa | tail -1 | cut -c 2450-43422 >> LH_draft.fa [ 3:50午後 ]
iu@bielinux[result] ls -l LH*.fa [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 2400619 6月 27 2017 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
② iu@bielinux[result] grep ">" LH_draft.fa [ 3:50午後 ]
>chromosome
>plasmid1
>plasmid2
③ iu@bielinux[result] wc LH_draft.fa [ 4:05午後 ]
6 6 2400619 LH_draft.fa
iu@bielinux[result] █ [ 4:05午後 ]
```


W3-0: ファイル分割2

第7回W10-3からW10-4(スライド56-58)と同じようなファイル分割方法だと、①こんな感じになる。この例のように配列数が10個以下程度なら、私は手作業でやります。しかし、配列数が100とか1000個程度などの局面では、シェルスクリプトがおそらくよく使われます

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa
-rwxrwxrwx 1 iu iu 2400619  6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662  6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa > mongee1.fa      [ 4:51午後 ]
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > mongee2.fa
iu@bielinux[result] tail -n 2 LH_draft.fa > mongee3.fa     [ 4:51午後 ]
iu@bielinux[result] ls -l LH_draft.fa mongee*             [ 4:51午後 ]
-rwxrwxrwx 1 iu iu 2400619  6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2277994  6月 27  2017 mongee1.fa
-rwxrwxrwx 1 iu iu  81641  6月 27  2017 mongee2.fa
-rwxrwxrwx 1 iu iu  40984  6月 27  2017 mongee3.fa
iu@bielinux[result] █                                     [ 4:51午後 ]
```



W3-1: シェルスクリプト

先程とは異なり、①contig[0-9].fa作成のためのスクリプトを**統一的に記述**している。②や③の記述法は**一見無駄に見える**が、このような書き方でも目的を達成できるところがポイント

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa
-rwxrwxrwx 1 iu iu 2400619  6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662  6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] █
```



W3-1: シェルスクリプト

①で作成した、②contig[0-9].fa
と、先に作成したmongee[0-9].fa
のファイルサイズは確かに同じ

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:19午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 5:19午後]
-rwxrwxrwx 1 iu iu 2400619 6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] ls -l mongee* contig* [ 5:19午後]
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 16:51 mongee1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 16:51 mongee2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 16:51 mongee3.fa
iu@bielinux[result] [ 5:28午後]
```



①ファイルサイズだけでなく、中見も完全に同じであることがわかります

W3-1: シェルスクリプト

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:19午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2400619 6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] ls -l mongee* contig* [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 16:51 mongee1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 16:51 mongee2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 16:51 mongee3.fa
iu@bielinux[result] diff mongee1.fa contig1.fa [ 5:28午後 ]
iu@bielinux[result] diff mongee2.fa contig2.fa [ 5:48午後 ]
iu@bielinux[result] diff mongee3.fa contig3.fa [ 5:48午後 ]
iu@bielinux[result] [ 5:48午後 ]
```



ここでは、配列数が100とか1000個程度からなるmulti-FASTAファイルの分割を想定して、シェルスクリプトで記述するやり方を伝授しようとしています。①のようなコマンドを100とか1000個書かなくて済むので覚える価値があります

目的を再確認

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:19午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2400619 6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] ls -l mongee* contig* [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 16:51 mongee1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 16:51 mongee2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 16:51 mongee3.fa
iu@bielinux[result] diff mongee1.fa contig1.fa [ 5:28午後 ]
iu@bielinux[result] diff mongee2.fa contig2.fa [ 5:48午後 ]
iu@bielinux[result] diff mongee3.fa contig3.fa [ 5:48午後 ]
iu@bielinux[result] █ [ 5:48午後 ]
```



W3-2: 参考資料

平成27年度NGSハンズオン講習会

①

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

H27年度概要

H27年度講義日程・参考資料

H26年度講習会の情報については[こちら](#)をご覧ください。

H27年度実施報告書・講義資料・動画等

● [講習会実施報告書 \(PDF: 2.17MB\)](#) および [受講者アンケート集計結果 \(データ集\) \(PDF: 662KB\)](#)

● [講義資料・動画](#) *講義資料一覧のファイル名をクリックすると資料ファイル (PDF等) がダウンロードできます。

実施日	実施時間	大項目	項目	レベル	習得技術	担当講師(敬称略)	講義資料・動画(統合TV)				
7月22日 (水)	10:30-12:00	PC環境の構築	Bio-Linux8とRのインストール状況確認		<ul style="list-style-type: none"> Linux導入 R導入 NGS解析に必要な環境構築技術 	門田 幸二 (東京大学)	事前予習資料一覧 (PDF:52KB)				
	13:15-14:45										
	15:00-16:30									寺田 透 (東京大学)	講義資料一覧 (PDF:108KB)
	16:45-18:15										
7月23日 (木)	10:30-12:00	UNIX/Linuxとスクリプト言語	Linux基礎	初級	UNIXの基礎の理解	門田 幸二 (東京大学)	講義資料一覧 (PDF:32KB)				
	13:15-14:45			中級							
	15:00-16:30								統合TV		
	16:45-18:15										
7月24日 (金)	10:30-12:00		スクリプト言語	中級	シェルスクリプト	服部 恵美 (アメリエフ)	講義資料 (PDF:1.8MB)				
	13:15-14:45										
	15:00-16:30									統合TV	

②

まず、①の部分は2で固定。理由は、出力ファイルがsingle-FASTA形式だから

W3-3: 戦略を練る

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:19午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2400619 6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] ls -l mongee* contig* [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 16:51 mongee1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 16:51 mongee2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 16:51 mongee3.fa
iu@bielinux[result] diff mongee1.fa contig1.fa [ 5:28午後 ]
iu@bielinux[result] diff mongee2.fa contig2.fa [ 5:48午後 ]
iu@bielinux[result] diff mongee3.fa contig3.fa [ 5:48午後 ]
iu@bielinux[result] █ [ 5:48午後 ]
```


W3-3: 戦略を練る

変わっているのは①と②の部分。①1-3までのループを回して、その2倍の値として②を表現できるだろうと妄想する

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 5:19午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l LH*.fa [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2400619 6月 27 15:50 LH_draft.fa
-rwxrwxrwx 1 iu iu 2433662 6月 12 15:25 LH_hgap.fa
iu@bielinux[result] head -n 2 LH_draft.fa | tail -n 2 > contig1.fa
iu@bielinux[result] head -n 4 LH_draft.fa | tail -n 2 > contig2.fa
iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 > contig3.fa
iu@bielinux[result] ls -l mongee* contig* [ 5:19午後 ]
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 16:51 mongee1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 16:51 mongee2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 16:51 mongee3.fa
iu@bielinux[result] diff mongee1.fa contig1.fa [ 5:28午後 ]
iu@bielinux[result] diff mongee2.fa contig2.fa [ 5:48午後 ]
iu@bielinux[result] diff mongee3.fa contig3.fa [ 5:48午後 ]
iu@bielinux[result] █ [ 5:48午後 ]
```

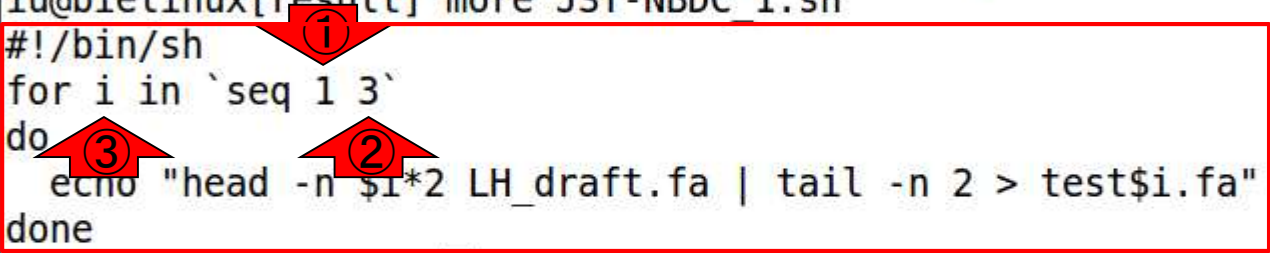

W3-4: 基本形

```
iu@bielinux[~/Desktop/mac_share/result] [ 3:27午後]
iu@bielinux[result] pwd [ 3:27午後]
/home/iu/Desktop/mac_share/result
② iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
③ iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後]
#!/bin/sh
for i in `seq 1 3`
do
    echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
① iu@bielinux[result] [ 3:27午後]
```

①1から②3までの1刻みのループを回し、③iという変数で取り扱う。別にiでなくてもよいが、変数名として、i, j, kといった順番で使うヒトも一定数存在する

W3-5: 解説

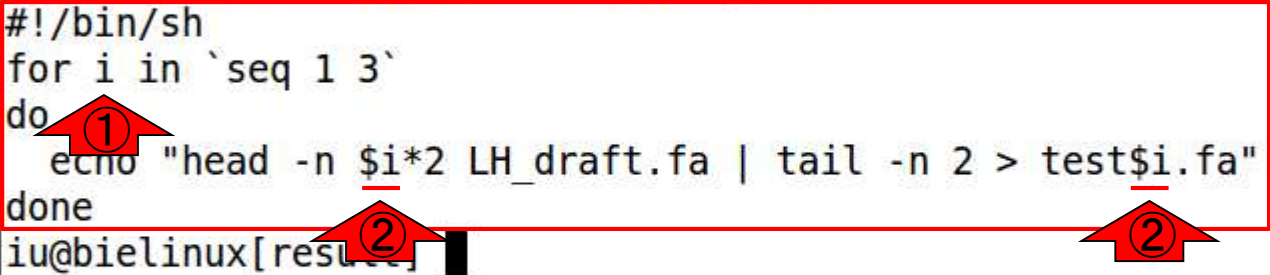
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:27午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後 ]
#!/bin/sh
do
  echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] [ 3:27午後 ]
```



①変数iは、②\$iとして取り扱う。つまり\$を追加する。なぜ?という類のものではなく、お約束です

W3-5: 解説

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:27午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result]
```



①「echo “…”」で困っていますが、これはLinuxコマンドのechoであり、“”内部の実際に行いたいコマンドがどんな感じになっているかを実行前に確認するためにつけています

W3-5: 解説

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:27午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] [ 3:27午後 ]
```



W3-6: echoで確認

- ①shコマンドでJST-NBDC_1.shを実行。
- ②echoで囲った中身に相当する、赤下線部分が表示されていることがわかる

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:27午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] sh JST-NBDC_1.sh [ 3:27午後 ]
head -n 1*2 LH_draft.fa | tail -n 2 > test1.fa
head -n 2*2 LH_draft.fa | tail -n 2 > test2.fa
head -n 3*2 LH_draft.fa | tail -n 2 > test3.fa
iu@bielinux[result] [ 3:28午後 ]
```



W3-7: head部分

実際に全体を実行してエラーに遭遇してもよいが一応説明。headコマンド部分の①は最初のy行という数値を指定するところだが、1*2とか3*2という掛け算の*が含まれている。このような指定法はダメ(だということのエラーに遭遇して学習する)

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 7:19午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 7:19午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] sh JST-NBDC_1.sh [ 7:19午後 ]
head -n 1*2 LH_draft.fa | tail -n 2 > test1.fa
head -n 2*2 LH_draft.fa | tail -n 2 > test2.fa
head -n 3*2 LH_draft.fa | tail -n 2 > test3.fa
iu@bielinux[result] █ [ 2:51午後 ]
```



W3-7: head部分

①や②を実際に実行してみても、赤下線部分が原因でエラーが出ます。エラーが出ないようにするには、③のように3*2の結果である6を与える必要があります

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:27午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_1.sh
iu@bielinux[result] ls -l *.sh [ 3:27午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
iu@bielinux[result] more JST-NBDC_1.sh [ 3:27午後 ]
#!/bin/sh
for i in `seq 1 3`
do
    echo "head -n $i*2 LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] sh JST-NBDC_1.sh [ 3:27午後 ]
head -n 1*2 LH_draft.fa | tail -n 2 > test1.fa
head -n 2*2 LH_draft.fa | tail -n 2 > test2.fa
head -n 3*2 LH_draft.fa | tail -n 2 > test3.fa
① iu@bielinux[result] head -n 3*2 LH_draft.fa [ 3:28午後 ]
zsh: no matches found: 3*2
② iu@bielinux[result] head -n 3*2 LH_draft.fa | tail -n 2 [ 3:29午後 ]
zsh: no matches found: 3*2
③ iu@bielinux[result] head -n 6 LH_draft.fa | tail -n 2 [ 3:29午後 ]
```


W3-7: head部分

```
iu@bielinux[~/Desktop/mac_share/result] 15:30
CCCAAGCAGTCGGCATGCTAGTTTGTAAATCAACGCCGGGCAGTCGTGAGTTCACTGACATTCGGTAATTTGG
CTGACATTTCCAATGAAGTTCGACAATCTGAGAAGCAGCGATTCGGTAAATTAAGCTATCTTTACCGTGCTA
TTCGCCATATTGGTCACAATAAGTCACTGCCAATTCGATATCAATTCAAACACGAAGGAAGCCACACCTTAA
AAACATGGTTCTGTTTGATTACAACGACCAAATCAGTCGGTGGGCACGTCTATAGCGCGTCTGCTCCAGGAA
AAATGCATATTAGTCTACTTAACAACATTGGCTGGCGGCAAGTAATTCATATATTTGGTTCGCACTAACGG
GGAAC TTGCAAACTCCAAGGCCATTACACAGCTAACGGCAACCAGTGCACGAATTACGAGTGCAACTGGTC
AAGCCGTGACGACACGTATTGACGGCGACCCAGCGGTTAACTGCCAATTGAATTGACCTATTTGACAGACC
GCTTCGAATTGATCGTACCAACAGTCATCGAATAACGACGTATTTTTTTATTAATATATACATATATTAATTG
CAAGAATTCATGTTATCGTCATTCTTATGAAGAGTATTTACCTTAACAAAAGCAAGCACAAATATACTTTAA
GACAAAATCACAAATTAATTTTTTATCTTGTCATTAATAATTGGATTTTCAGATACCAACCTCATGCCAAT
TAATGAACAGGATAATCATTGTACGCGGAAAGGCGCGATGAGACGCCGTCCGGAAAACAGGTATTAACGGA
CTACCTTTATTCGATATTGAAAGAAATTCGATGTACAACGATGATGCTTAAATGAGATAATAATGTAGATCC
AGTCTGGTAAGTGTCAAATATGCTATAAAAGCACGGGTTGAAAACGAGGATAGAAAAAGGGGAATGTGTTCA
TGTCTGTAGATTCTTGGTTTAAAAATTATGTTTCAGAAAACATAAATATAGATAGTAAAAAAAGCGCAAGGG
CAAGAAGTAGTAGAAATTGGTTAACTTCAAATATTAAGATCTTAGTCAAAAAACGAGGAAAACCTTGAAT
TATACTCGGACTCAGAATTTGCACTAAAAATGGGATCATTTGCTCGAAAGACACAGATTAGACCTCTTGACG
ATGTTGACCAAATGATTATCTTTTCGGCAAAGGGGAGCACCGCTAATTTAGATACGTCTCAATGGAATCAGG
TGTTTGTAATGTTCCAGATAGCGCTCCAGAATTAAGAAAATGGATGGAGAAAATGGGCTTAGTTCTATAA
AAGTCTTGAATTATCTTAAACAGCTATTGAATGGAATATCGCAATATCAATCGGCAGATATTAAGGAGTC
AGCAAGCACTTAGACTGGAATTATCAAGCTACGACTGGGGATTCGATATAGTCCCTGGATTTAGAACTGTTG
ATGAT
iu@bielinux[result] [ 3:30午後 ]
```


①シェルスクリプトの発展形ファイルJST-NBDC_2.sh
を、②wgetし、③moreで確認

W3-8: 発展形

```
iu@bielinux[~/Desktop/mac_share/result] [ 3:48午後]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
② iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
③ iu@bielinux[result] more JST-NBDC_2.sh [ 3:52午後]
#!/bin/sh
for i in `seq 1 3`
do
    j=`expr $i \* 2`
    echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
done
①
iu@bielinux[result] [ 3:52午後]
```

JST-NBDC_1.shとの違いは、**赤下線部分のみ**。jという別の変数を用いて*\$i*2*を表現したただけだが、シェルスクリプトの掟に従うと、このようになります

W3-8: 発展形

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:48午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] more JST-NBDC_2.sh [ 3:52午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  j=`expr $i \* 2`
  echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] [ 3:52午後 ]
```

W3-8: 発展形

①は $\$i*2$ の*に相当する部分ですが、「なんでもよい、という意味でのワイルドカードの*」と「掛け算の意味での*」を区別する必要があります。後者の意味として用いたい場合に、②¥(入力できませんが、バックスラッシュです)を添えます。本当にこの説明でいいのかは怪しいです

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] more JST-NBDC_2.sh [ 3:52午後 ]
#!/bin/sh
for i in `seq 1 3`
do
    j=`expr $i \* 2`
    echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] █ [ 3:52午後 ]
```



W3-8: 発展形

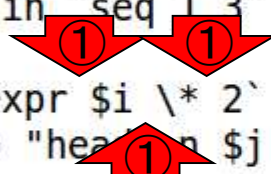
```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:48午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] more JST-NBDC_2.sh [ 3:52午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  j=`expr $i \* 2`
  "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] [ 3:52午後 ]
```



掟の続き。①のところにはスペースを入れないといけません

W3-8: 発展形

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 3:48午後 ]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] more JST-NBDC_2.sh [ 3:52午後 ]
#!/bin/sh
for i in `seq 1 3`
do
    j=`expr $i \* 2`
    echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
done
iu@bielinux[result] █ [ 3:52午後 ]
```



①の記号は、②Shiftキーを押しながら、③@のキーを押すと出ます

W3-8: 発展形

```
iu@bielinux[~/Desktop/mac_share/result] [ 3:48午後 ]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_2.sh
iu@bielinux[result] ls -l *.sh [ 3:50午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] more JST-NBDC_2.s
#!/bin/sh
for i in `seq 1 3`
do
  j=`expr $i \* 2`
  echo "head -n $j LH_draft.fa | tail
done
iu@bielinux[result] █
```



①shコマンドでJST-NBDC_2.shを実行。
②意図通りになっていることがわかる

W3-9: echoで確認

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:23午後]
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l *.sh [ 4:23午後]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
iu@bielinux[result] sh JST-NBDC_2.sh [ 4:23午後]
head -n 2 LH_draft.fa | tail -n 2 > test1.fa
head -n 4 LH_draft.fa | tail -n 2 > test2.fa
head -n 6 LH_draft.fa | tail -n 2 > test3.fa
iu@bielinux[result] [ 4:23午後]
```



W3-10: 発展形2

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd [ 4:41午後]
/home/iu/Desktop/mac_share/result
② iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_3.sh
iu@bielinux[result] ls -l *.sh [ 4:42午後]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
-rwxrwxrwx 1 iu iu 162 6月 27 18:02 JST-NBDC_3.sh
③ iu@bielinux[result] more JST-NBDC 3.sh [ 4:42午後]
#!/bin/sh
for i in `seq 1 3`
do
    j=`expr $i \* 2`
    #echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
    head -n $j LH_draft.fa | tail -n 2 > test$i.fa
done
iu@bielinux[result] [ 4:42午後]
```



W3-10: 発展形

JST-NBDC_2.shとの違いは、①と②の行の部分。①のecho行頭に#を入れてコメントアウト(実行されないように)している。削除するのと同じだが、エラーなど問題が起こったときの対処用などの目的で#をつけたまま残しておくことはよくやる。②は①の赤下線部分と同じもの。ここが実際に実行される部分

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_3.sh
iu@bielinux[result] ls -l *.sh [ 4:42午後 ]
-rwxrwxrwx 1 iu iu 95 6月 27 18:00 JST-NBDC_1.sh
-rwxrwxrwx 1 iu iu 112 6月 27 18:02 JST-NBDC_2.sh
-rwxrwxrwx 1 iu iu 162 6月 27 18:02 JST-NBDC_3.sh
iu@bielinux[result] more JST-NBDC_3.sh [ 4:42午後 ]
#!/bin/sh
for i in `seq 1 3`
do
  j=`expr $i \* 2`
  #echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
  head -n $j LH_draft.fa | tail -n 2 > test$i.fa
done
iu@bielinux[result] █ [ 4:42午後 ]
```



W3-11: 実行

①shコマンドでJST-NBDC_3.shを実行。echoをコメントアウトしているので、意図通り何も表示されない。また、②イメージ通りの出力ファイルtest[0-9].faが作成されていることがわかる

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls *.fa
contig1.fa  hoge.fa      mongee1.fa  sequence1.fa  sequence3_trimmed.fa
contig2.fa  LH_draft.fa mongee2.fa  sequence2.fa  sequence4.fa
contig3.fa  LH_hgap.fa  mongee3.fa  sequence3.fa
① iu@bielinux[result] sh JST-NBDC_3.sh
iu@bielinux[result] ls *.fa
contig1.fa  LH_draft.fa  mongee3.fa  sequence3_trimmed.fa  test3.fa ②
contig2.fa  LH_hgap.fa  sequence1.fa  sequence4.fa
contig3.fa  mongee1.fa  sequence2.fa  test1.fa
hoge.fa     mongee2.fa  sequence3.fa  test2.fa } ②
iu@bielinux[result] █
[ 5:33午後]
[ 5:33午後]
[ 5:33午後]
[ 5:33午後]
```

W3-12: 確認

①「ls -l」で詳細情報を表示。ファイルサイズの観点からは、contig[0-9].faとtest[0-9].faは全く同じ。②念のためdiffコマンドでも確認。何も表示されていないので中身も同一であることを確認したことになる。③違うもの同士だと、違いのある部分がババッと表示されます

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] ls -l contig*.fa test*.fa
-rwxrwxrwx 1 iu iu 2277994 6月 27 17:19 contig1.fa
-rwxrwxrwx 1 iu iu 81641 6月 27 17:19 contig2.fa
-rwxrwxrwx 1 iu iu 40984 6月 27 17:19 contig3.fa
-rwxrwxrwx 1 iu iu 2277994 6月 28 17:33 test1.fa
-rwxrwxrwx 1 iu iu 81641 6月 28 17:33 test2.fa
-rwxrwxrwx 1 iu iu 40984 6月 28 17:33 test3.fa
iu@bielinux[result] diff contig1.fa test1.fa
iu@bielinux[result] diff contig2.fa test2.fa
iu@bielinux[result] diff contig3.fa test3.fa
iu@bielinux[result] diff test2.fa test3.fa
```

[5:51午後]

[5:51午後]

[5:51午後]

[5:51午後]

[5:51午後]



W3-12: 確認

リターンキーを押して、ババッと表示された後の状態です。diffコマンドの挙動については、自分で納得できるようなファイルを用意して確認するのが一番ですので、ぜひやってみてください

iu@bielinux[~/Desktop/mac_share/result]

```
GACCCAAGCAGTCGGCATGCTAGTTTGTAAATCAACGCCGGGCAAGTCGTGAGTTCAC TGACATTCGGTAATTT
GGCTGACATTTCCAATGAAGTTCGACAATCTGAGAAGCAGCGATTTCGGTAAATTAAGCTATCTTTACCGTGC
TATTCGCCATATTGGTCACAATAAGTCACTGCCAATTCGATATCAATTCAAACACGAAGGAAGCCACACCTT
AAAAACATGGTTCTGTTTGATTACAACGACCAAATCAGTCGGTGGGCACGTCTATAGCGCGTCTGCTCCAGG
AAAAATGCATATTAGTCTACTTAACAACATTGGCTGGCGGCAAGTAATTCCATATATTTGGTTCGCACTAAC
GGGGAACCTTGCAAACCTCAAGGCCATTACACAGCTAACGGCAACCAGTGCACGAATTACGAGTGCAACTGG
TCAAGCCGTGACGACACGTATTGACGGCGACCCAGCGGTTAAACTGCCAATTGAATTGACCTATTTGACAGA
CCGCTTCGAATTGATCGTACCAACAGTCATCGAATAACGACGTATTTTTTTATTAATATATACATATATTAAT
TGCAAGAATTCATGTTATCGTCATTCTTATGAAGAGTATTTACCTTAACAAAAGCAAGCACAAATATACTTT
AAGACAAAATCACAAATTAATTTTTTATCTTGTCATTAATAATTGGATTTTCAGATACCAACCTCATGCCA
ATTAATGAACAGGATAATCATTGTACGCGGAAAGGCGCGATGAGACGCCGTCGGGAAAACAGGTATTAACG
GACTACCTTTATTCGATATTGAAAGAAATTCGATGTACAACGATGATGCTTAAATGAGATAATAATGTAGAT
CCAGTCTGGTAAGTGTCAAATATGCTATAAAAGCACGGGTTGAAAACGAGGATAGAAAAAGGGGAATGTGTT
CATGTCTGTAGATTCTTGGTTTAAAAATTATGTTTCAGAAAACATAAATATAGATAGTAAAAAAGCGCAAG
GGCAAGAACTAGTAGAAATTGGTTAACTTCAAATATTAAGATCTTAGTCAAAAAAACGAGGAAAACCTTGA
ATTATACTCGGACTCAGAATTTGCACTAAAAATGGGATCATTTGCTCGAAAGACACAGATTAGACCTCTTGA
CGATGTTGACCAAATGATTATCTTTTCGGCAAAGGGGAGCACCGCTAATTTAGATACGTCTCAATGGAATCA
GGTGTGTTGTAAATGTTCCAGATAGCGCTCCAGAATTAAGAAAATGGATGGAGAAAATGGGCTTAGTTCTAT
AAAAGTCTTGAATTATCTTAAACAGCTATTGAATGGAATATCGCAATATCAATCGGCAGATATTAAGAGGAG
TCAGCAAGCACTTAGACTGGAATTATCAAGCTACGACTGGGGATTTCGATATAGTCCCTGGATTTAGAAGTGT
TGATGAT
```

iu@bielinux[result] █

[5:54午後]

W3-13: 発展形3

「expr シェル 遅い」などでググればわかりますが、JST-NBDC_3.sh内で使われているexprは遅いらしいです。赤下線のように書くといいらしい。①高速版のJST-NBDC_4.shを②wgetし、③more。後は省略

```
iu@bielinux[~/Desktop/mac_share/result]
iu@bielinux[result] pwd
/home/iu/Desktop/mac_share/result
iu@bielinux[result] wget -cq http://www.iu.a.u-tokyo.ac.jp/~kadota/book/
JST-NBDC_4.sh
iu@bielinux[result] more JST-NBDC_4.sh
#!/bin/sh
for i in `seq 1 3`
do
  j=$((i \* 2))
  #echo "head -n $j LH_draft.fa | tail -n 2 > test$i.fa"
  head -n $j LH_draft.fa | tail -n 2 > test$i.fa
done
iu@bielinux[result] █
```

