

スライド8までは自習。当日はスライド9から始める予定。スライド131-186は当日省略予定。講習会後に各自で復習してください



## 第3部:NGS解析(中～上級)

～トランスクリプトームアセンブリ、発現量推定～

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/>



# 利用プログラムの簡単な解説

- Bowtie2 (Langmead and Salzberg, *Nat Methods*, 2012)
  - マッピングプログラム
- Bridger (Chang et al., *Genome Biol.*, 2015)
  - *de novo*トランスクリプトームアセンブラー
- DDBJ Pipeline (Nagasaki et al., *DNA Res.*, 2013)
  - マッピングや*de novo*アセンブリを行ってくれるウェブツール(クラウド解析環境)
- FaQCs (Lo and Chain, *BMC Bioinformatics*, 2014)
  - Quality Control用プログラム。クオリティフィルタリングやアダプター除去が主目的
- FastQC
  - Quality Control用プログラム。アダプターの混入などNGSデータのクオリティチェックが主目的
- QuasR (Gaidatzis et al., *Bioinformatics*, 2015)
  - (主に)マッピングからカウント情報取得まで行ってくれるRパッケージ
- Rockhopper2 (Tjaden, B., *Genome Biol.*, 2015)
  - バクテリア用*de novo*トランスクリプトームアセンブラー
- TIGAR2 (Nariai et al., *BMC Genomics*, 2014)
  - トランスクリプトーム発現量推定用プログラム、FPKM値とカウント情報が主な出力
- Trinity (Grabherr et al., *Nat Biotechnol.*, 2011)
  - *de novo*トランスクリプトームアセンブラー

# 2016.08.04は

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриптーム、正規化、発現変動、統計、モ<sup>ル</sup>  
(last modified 2016/06/03, since 2011)

- 基本的な利用法 (last modified 2015/04/03)

- サンプルデータ (last modified 2016/05/13)

- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | NGSハンズオン講習会2016

- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | NGSハンズオン講習会2015 (last modified 2015/06/03)

- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース2014 (last modified 2014/05/12)

- 書籍 | トランскриптーム解析 | について (last modified 2014/05/12)

- 書籍 | トランскриптーム解析 | について (last modified 2014/05/12)

- 書籍 | トランскриптーム解析 | について (last modified 2014/05/12)

- 書籍 | トランскриптーム解析 | について (last modified 2014/05/12)

- 書籍 | トランскриптーム解析 | について (last modified 2014/05/12)

### バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | NGSハンズオン講習会2016

平成28年度NGSハンズオン講習会の実習で用いるリンク先、データファイル、コピペ用コード集などはここで示します。

- はじめに(講習会参加者必読) (last modified 2016/06/21)

- 事前準備 | Bio-Linux 8とRのインストール状況確認(2016.07.19) (last modified 2016/06/19)

- 第1部 | 統計解析 | について (last modified 2016/06/18)

- 第1部 | 統計解析 | ゲノム解析、塩基配列解析(2016.07.20) (last modified 2016/05/07)

- 第1部 | 統計解析 | トランскриптーム解析1(2016.07.21) (last modified 2016/06/18)

- 第1部 | 統計解析 | トランскриптーム解析2(2016.07.22) (last modified 2016/06/02)

- 第2部 | NGS解析(初～中級) | について (last modified 2016/06/16)

- 第2部 | NGS解析(初～中級) | NGS解析基礎(2016.07.25) (last modified 2016/04/26)

- 第2部 | NGS解析(初～中級) | ゲノムReseq、変異解析(2016.07.26) (last modified 2016/04/26)

- 第2部 | NGS解析(初～中級) | RNA-seq(2016.07.27) (last modified 2016/04/26)

- 第2部 | NGS解析(初～中級) | ChIP-seq(2016.07.28) (last modified 2016/04/26)

- 第3部 | NGS解析(中～上級) | について (last modified 2016/06/21)

- 第3部 | NGS解析(中～上級) | Linux環境でのデータ解析: JavaやRの利用法(2016.08.01) (last modified 2016/06/13)

- 第3部 | NGS解析(中～上級) | 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(2016.08.02) (last modified 2016/06/21)

- 第3部 | NGS解析(中～上級) | クラウド環境との連携、ロングリードデータの解析(2016.08.03) (last modified 2016/06/20)

- 第3部 | NGS解析(中～上級) | トランскриптームアセンブリ、発現量推定(2016.08.04) (last modified 2016/06/22)

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



乳酸菌RNA-seqデータ解析のおさらい。①FaQCsとFastQCはゲノム配列既知or未知とは無関係なので、アダプター配列除去部分は一律にやるところ

# おさらい1

## ■ paired-end Illumina HiSeqデータ(SRR616268)

- サンプルは、*Lactobacillus casei* 12A (第3回W21)。ゲノム配列既知
  - Ensembl Bacteria (release 22)当時はコンティグレベル (第3回W11; 第1回の図2)  
ファイル名 : *Lactobacillus\_casei\_12a.GCA\_000309565.1.22.dna.toplevel.fa.gz* (genome.fa)
  - Ensembl Bacteria (release 30)頃は染色体レベル (第5回W13)  
ファイル名 : *Lactobacillus\_casei\_12a.GCA\_000309565.2.30.dna.toplevel.fa.gz*
- オリジナルのリード数(第3回W23とW24)
  - sraファイルのリード数は135,073,834
  - FASTQファイルのリード数は134,755,996
  - bzip2圧縮FASTQファイルで合計約15GB
- サブセット(最初の100万リード)の取得(第4回W6)
  - gzip圧縮FASTQファイルで合計約140MB
  - forward側(*SRR616268sub\_1.fastq.gz*) : リード長は107 bp
  - reverse側(*SRR616268sub\_2.fastq.gz*) : リード長は93 bp
- ① □ FaQCs実行(第5回W1; 2016.08.01)
  - 1,000,000リード → 977,202リード (第5回W1-3)
  - forward側(*QC.1.trimmed.fastq.gz*) : リード長ばらばら
  - reverse側(*QC.2.trimmed.fastq.gz*) : リード長ばらばら
  - FastQC上で見られるIllumina adapterは消滅状態



① *de novo* transcriptome assemblyをやって、うまくいかなかつた原因が、②ゲノムへのマッピングでわかつたというストーリー展開に結果的になっている

# おさらい2

## ■ *de novo*ランスクリプトームアセンブリ(第5回W5とW6; 2016.08.01前半)

①

- バクテリア用のアセンブラRockhopper2を実行して変な結果に遭遇した

- paired-end (QC.1.trimmed.fastqとQC.2.trimmed.fastq) : 0 transcript or contig (W5-2)
- single-end (forward側のみ; QC.1.trimmed.fastq) : 1 transcript (W6-2)
- single-end (reverse側のみ; QC.2.trimmed.fastq) : 423 transcripts (W6-4)

## ■ ゲノムマッピング(第5回W13からW15; 2016.08.01後半から2016.08.02前半)

②

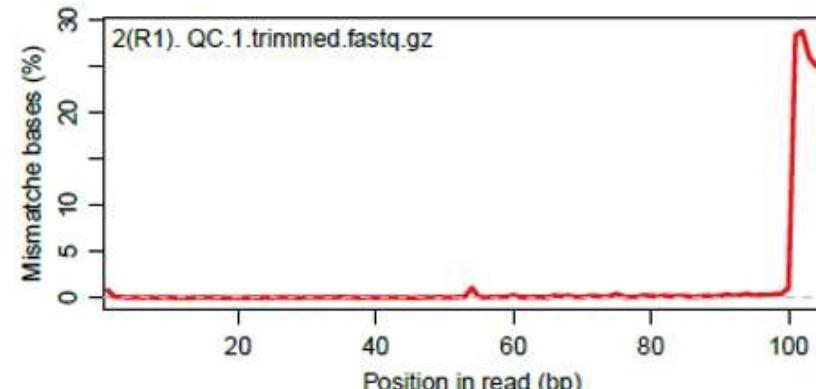
- マップされる側: Lactobacillus\_casei\_12a.GCA\_000309565.2.30.dna.toplevel.fa.gz

- マップする側: paired-endのgzip圧縮FASTQファイル

- (SRR616268sub\_1.fastq.gzとSRR616268sub\_2.fastq.gz) ← この結果は重要ではない
- QC.1.trimmed.fastq.gzとQC.2.trimmed.fastq.gz

- QuasRパッケージでやってみたら偶然原因が判明した(第5回W15-5)

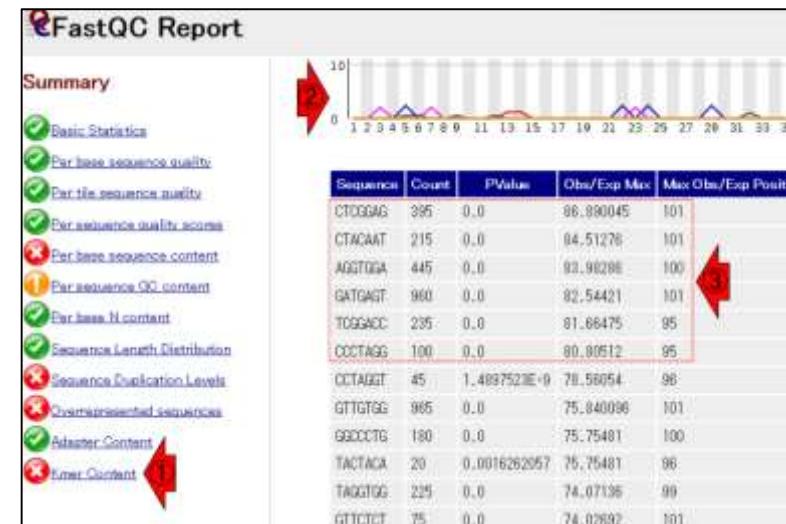
- 乳酸菌に由来しない配列を多く含む領域(forward側の100–107bp付近;  $f_{100-107}$ 問題)が見つかった



# おさらい3

問題の箇所をトリムしたらうまくいきましたね  
という話だが、ここまででは、Linuxテクやこう  
いうこともあるという事例紹介が主目的

- トリミング(第5回W16; 2016.08.02前半)
  - FaQCs実行前のforward側(SRR616268sub\_1.fastq.gz)のみに適用(W16)
    - forward側(hoge\_1.fastq.gz): リード長は107 → 99 bp
- トリム後のデータで再解析(第5回W17とW18; 2016.08.02前半)
  - Rockhopper2でアセンブリ再実行(第5回W17)
    - paired-end (hoge\_1.fastq.gzとSRR616268sub\_2.fastq.gz): 0 → 794 transcripts (W17-4)
  - QuasRでマッピング再実行(第5回W18)
    - トリミングの効果で乳酸菌ゲノム配列へのリードのマップ率が0.4 → 34.6%と劇的に改善
- FastQC段階で $f_{100-107}$ 問題に気づくことはできた(第5回W19; 2016.08.02前半)
  - --nogroupオプションつきでFastQCを実行し、Kmer\_Content項目を眺めるのも大事



①常識的にはこの論理展開は矛盾している。②*de novo transcriptome assembly*をやるのは、通常リファレンスゲノム配列がない場合。それゆえ、Rockhopper2の失敗原因を③ゲノムマッピング結果から突き止めるという流れは通常ない(近縁種へのマッピングなどの手段を除く)

# おさらい2

## ■ *de novo*トランスクriプトームアセンブリ

### ② □ バクテリア用のアセンブリRockhopper2

- paired-end (QC.1.trimmed.fastqとQC.2.trimmed.fastq) : 0 transcript or contig (W5-2)
- single-end (forward側のみ; QC.1.trimmed.fastq) : 1 transcript (W6-2)
- single-end (reverse側のみ; QC.2.trimmed.fastq) : 423 transcripts (W6-4)

## ■ ゲノムマッピング(第5回W13からW15; 2016.08.01後半から2016.08.02前半)

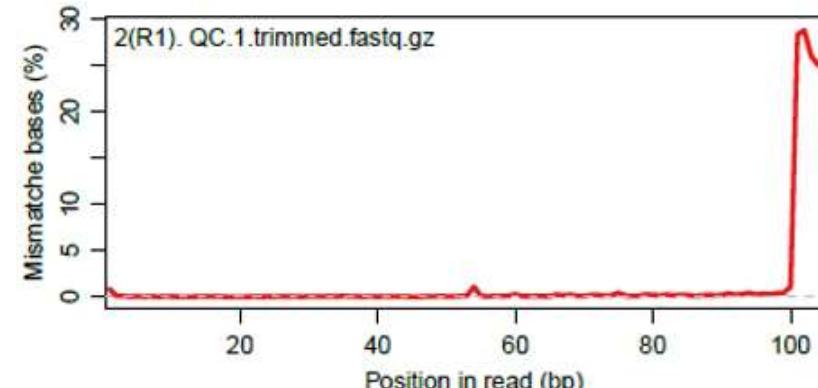
### ③ □ マップされる側: Lactobacillus\_casei\_12a.GCA\_000309565.2.30.dna.toplevel.fa.gz

### □ マップする側: paired-endのgzip圧縮FASTQファイル

- (SRR616268sub\_1.fastq.gzとSRR616268sub\_2.fastq.gz) ← この結果は重要ではない
- QC.1.trimmed.fastq.gzとQC.2.trimmed.fastq.gz

### □ QuasRパッケージでやってみたら偶然原因が判明した(第5回W15-5)

- 乳酸菌に由来しない配列を多く含む領域(forward側の100–107bp付近;  $f_{100-107}$ 問題)が見つかった



近縁種などゲノム配列を利用せずに、①*de novo transcriptome assembly*でそこそこのアセンブリ結果を得るための思考回路や手順を解説

# 問題設定

## ■ *de novo*トランスクリプトームアセンブリ(第5回W5とW6; 2016.08.01前半)

① □

バクテリア用のアセンブラRockhopper2を実行して変な結果に遭遇した

- paired-end (QC.1.trimmed.fastqとQC.2.trimmed.fastq) : 0 transcript or contig (W5-2)
- single-end (forward側のみ; QC.1.trimmed.fastq) : 1 transcript (W6-2)
- single-end (reverse側のみ; QC.2.trimmed.fastq) : 423 transcripts (W6-4)

もちろん実質的には答えが既知の状態ではある。つまり、forward側の100–107bp付近をトリムすれば794 transcripts程度にはなる(第5回W17–4)というのが1つの到達目標である。我々は「過去に $f_{100-107}$ 問題を経験した」ので、②の無様な結果と③FastQCの結果をじっくり眺めて解析戦略を練る

# 問題設定

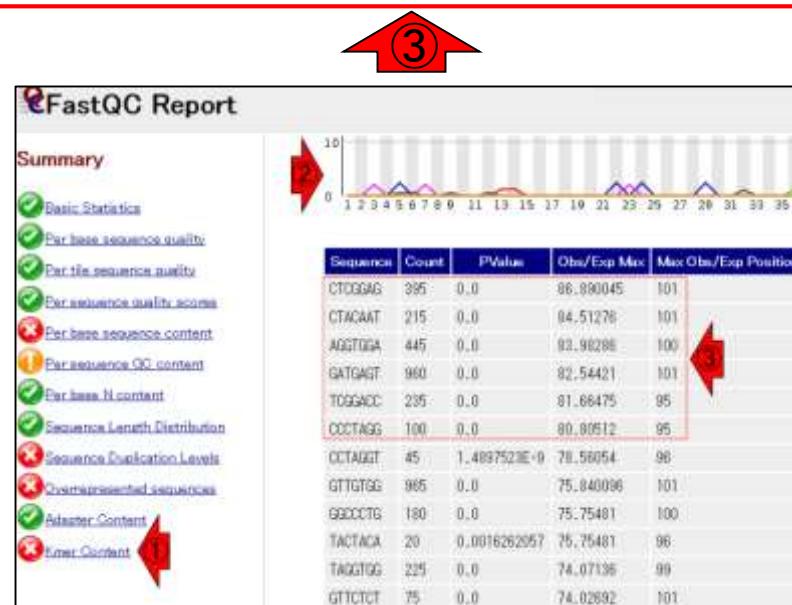
## ■ de novoranscripトームアセ

①

- バクテリア用のアセンブラーRockhopper2を実行して異なる結果に遭遇した
  - paired-end (QC.1.trimmed.fastqとQC.2.trimmed.fastq) : 0 transcript or contig (W5–2)
  - single-end (forward側のみ; QC.1.trimmed.fastq) : 1 transcript (W6–2)
  - single-end (reverse側のみ; QC.2.trimmed.fastq) : 423 transcripts (W6–4)

## ■ FastQC段階で $f_{100-107}$ 問題に気づくことはできた(第5回W19; 2016.08.02前半)

- --nogroupオプションつきでFastQCを実行し、Kmer\_Content項目を眺めるのも大事



# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 事前準備

## • 講義資料PDF(2016.07.xx版; 約xxMB)

## • おさらい1: paired-end Illumina HiSeqデータ(SRR616268)

下記のファイルサイズは環境によって異なります。目安です。

- サンプルは、*Lactobacillus casei* 12A (第3回W21)。ゲノム配列既知

- Ensembl Bacteria (release 22)当時はコンティグレベル (第3回W11; 第1回の図2)

ファイル名: *Lactobacillus\_casei\_12a.GCA\_000309565.1.22.dna.toplevel.fa.gz* (第3回W14-1で*genome.fai*に名前を変更)  
場所: ~/Desktop/hoge

- Ensembl Bacteria (release 30)頃は染色体レベル (第5回W13)

ファイル名: *Lactobacillus\_casei\_12a.GCA\_000309565.2.30.dna.toplevel.fa.gz*  
場所: ~/Documents/genomes

- オリジナルのリード数やファイルサイズ(第3回W23とW24)

DRAから取得したbz2圧縮ファイル。2つとも行数は539,023,984行、リード数は134,755,996リード(約1.35億)。大きすぎるの  
でBio-Linux8上には存在しない。

- forward側(*SRR616268\_1.fastq.bz2*): リード長は107 bp。ファイルサイズは7,662,128,101 bytes (約7.2GB)。

- reverse側(*SRR616268\_2.fastq.bz2*): リード長は93 bp。ファイルサイズは7,017,031,734 bytes (約6.6GB)。

- サブセット(最初の100万リード)の取得(第4回W6)

gzip圧縮FASTQファイルで合計約140MB。第4回W6でサブセット抽出を行い、第4回W9-7とW12-3でgzip圧縮しています。

- forward側(*SRR616268sub\_1.fastq.gz*): リード長は107 bp。ファイルサイズは76,659,501 bytes (約74MB)。

- reverse側(*SRR616268sub\_2.fastq.gz*): リード長は93 bp。ファイルサイズは68,682,959 bytes (約66MB)。

- 場所: ~/Documents/srp017156

- FaQCs実行(第5回W1; 2016.08.01)

1,000,000リード → 977,202リード (第5回W1-3)。リード長はばらばら。第5回W14-1でgzip圧縮しています。

- forward側(*QC.1.trimmed.fastq.gz*)。ファイルサイズは73,669,994 bytes (約71MB)。

- reverse側(*QC.2.trimmed.fastq.gz*)。ファイルサイズは65,909,777 bytes (約63MB)。

- 場所: ~/Documents/srp017156/result2

## • 事前準備

~/Documents/srp017156上に20160804ディレクトリを作成し、そこにgzip圧縮後のFaQCs実行結果ファイルをコピーして解析を行う。

```
cd ~/Documents/srp017156
mkdir 20160804
cd 20160804
pwd

#FaQCs実行結果ファイルのコピー
cp ~/Documents/srp017156/result2/QC.[0-9].*.gz .
#cp ~/Desktop/backup/QC.[0-9].*.gz .
ls -l

#ファイル名変更
mv QC.1.trimmed.fastq.gz data1.fq.gz
mv QC.2.trimmed.fastq.gz data2.fq.gz
ls -l
```

# 事前準備

## 事前準備

~/Documents/srp017156上に20160804ディレクトリを作成し、そこにgzip圧縮後のFaQCs実行結果ファイルをコピーして解析を行う。

```
cd ~/Documents/srp017156  
mkdir 20160804  
cd 20160804  
pwd
```

#FaQCs実行結果ファイルのコピー

```
cp ~/Documents/srp017156/result2/QC.[0-9].*.gz .  
#cp ~/Desktop/backup/QC.[0-9].*.gz .  
ls -l
```

#ファイル名変更

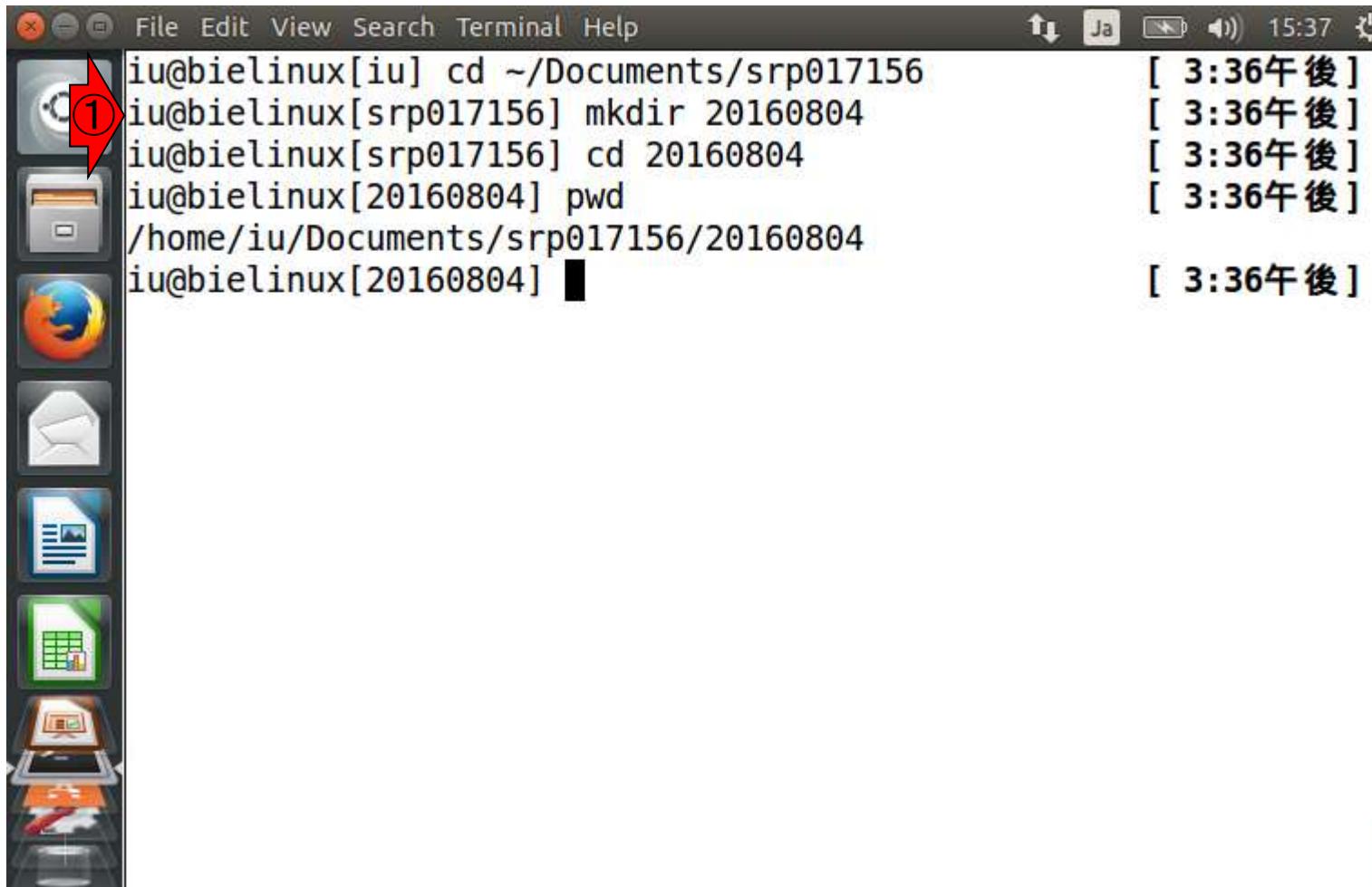
```
mv QC.1.trimmed.fastq.gz data1.fq.gz }  
mv QC.2.trimmed.fastq.gz data2.fq.gz }  
ls -l
```

詳細は省くが、①20160804ディレクトリを作成し、②FaQCs実行結果ファイルをコピーし、③ファイル名を短く変更しているだけ。次の3枚のスライドで実際の作業を示す

何らかの理由で②がうまくいかない場合やファイルがない場合は~/Desktop/backupのもので代用してください

詳細は省くが、①20160804ディレクトリを作成し、

# 事前準備



The screenshot shows a terminal window with a dark background and light-colored text. On the left, there is a vertical dock with icons for various applications, including a terminal icon which has a red circle with the number '1' over it, indicating a new message or notification.

```
iu@bielinux[iu] cd ~/Documents/srp017156 [ 3:36午後]
iu@bielinux[srp017156] mkdir 20160804 [ 3:36午後]
iu@bielinux[srp017156] cd 20160804 [ 3:36午後]
iu@bielinux[20160804] pwd [ 3:36午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] [ 3:36午後]
```

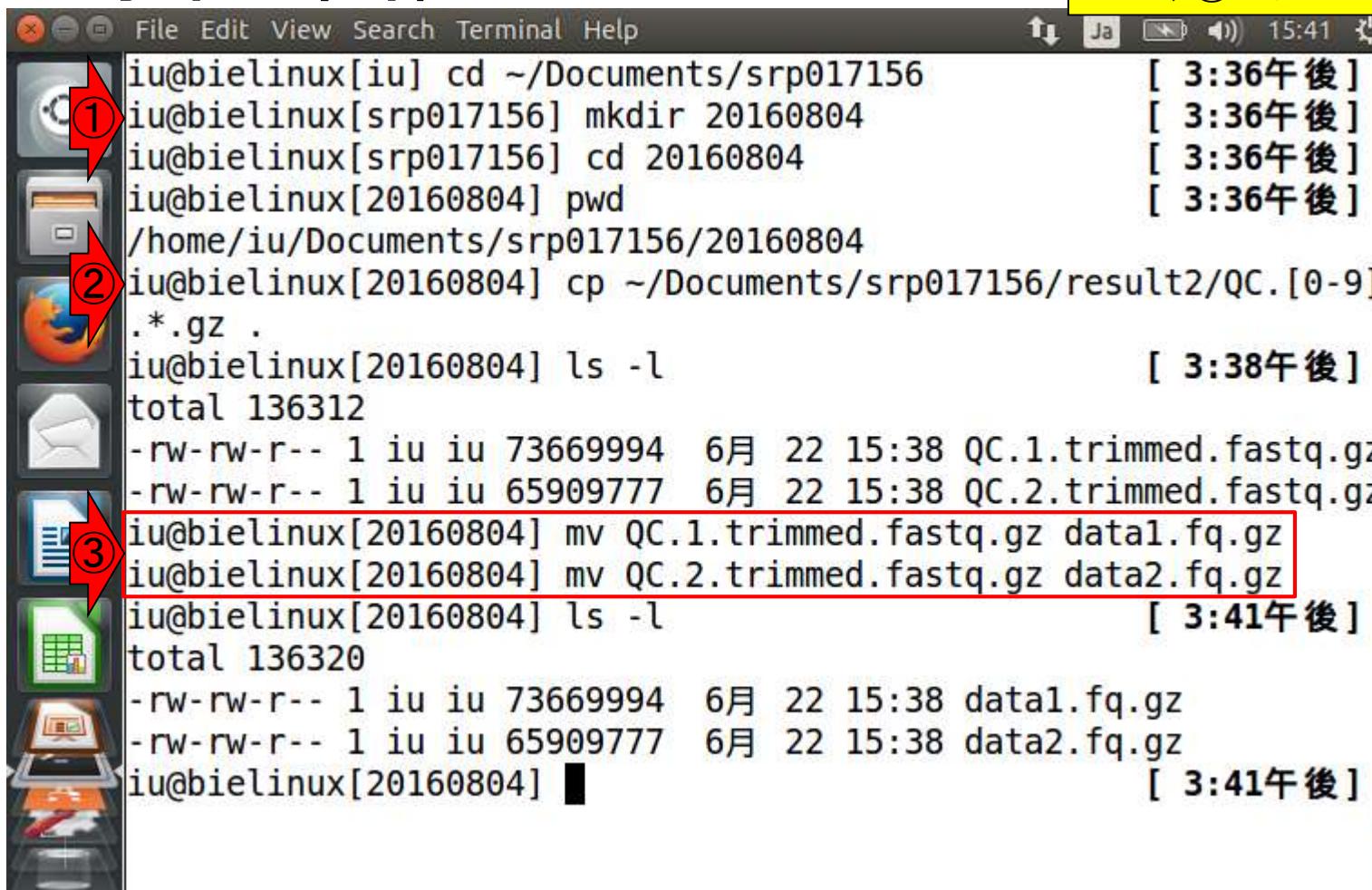
# 事前準備

詳細は省くが、①20160804ディレクトリを作成し、②FaQCs実行結果ファイルをコピーし、

```
iu@bielinux[iu] cd ~/Documents/srp017156
iu@bielinux[srp017156] mkdir 20160804
iu@bielinux[srp017156] cd 20160804
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] cp ~/Documents/srp017156/result2/QC.[0-9]
.*.gz .
iu@bielinux[20160804] ls -l
total 136312
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:38 QC.1.trimmed.fastq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:38 QC.2.trimmed.fastq.gz
iu@bielinux[20160804]
```

# 事前準備

詳細は省くが、①20160804ディレクトリを作成し、②FaQCs実行結果ファイルをコピーし、③ファイル名を短く変更しているだけ



```
iu@bielinux[iu] cd ~/Documents/srp017156 [ 3:36午後]
iu@bielinux[srp017156] mkdir 20160804 [ 3:36午後]
iu@bielinux[srp017156] cd 20160804 [ 3:36午後]
iu@bielinux[20160804] pwd [ 3:36午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] cp ~/Documents/srp017156/result2/QC.[0-9]*.gz .
iu@bielinux[20160804] ls -l [ 3:38午後]
total 136312
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:38 QC.1.trimmed.fastq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:38 QC.2.trimmed.fastq.gz
iu@bielinux[20160804] mv QC.1.trimmed.fastq.gz data1.fq.gz [ 3:41午後]
iu@bielinux[20160804] mv QC.2.trimmed.fastq.gz data2.fq.gz
iu@bielinux[20160804] ls -l [ 3:41午後]
total 136320
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:38 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:38 data2.fq.gz
iu@bielinux[20160804] █ [ 3:41午後]
```

# ちなみに…

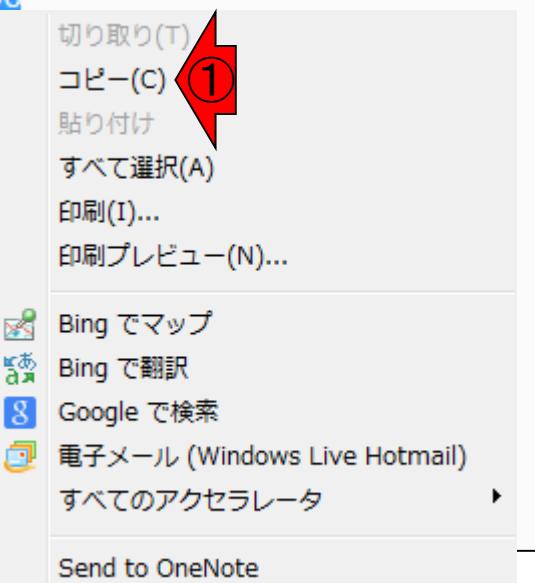
## 事前準備

~/Documents/srp017156上に20160804ディレクトリを作成し、そこにgzip圧縮後のFaQCs実行結果ファイルをコピーして解析を行う。

```
cd ~/Documents/srp017156  
mkdir 20160804  
cd 20160804  
pwd
```

```
#FaQCs実行結果ファイルの  
cp ~/Documents/srp0171!  
#cp ~/Desktop/backup/Q  
ls -l
```

```
#ファイル名変更  
mv QC.1.trimmed.fastq.  
mv QC.2.trimmed.fastq.  
ls -l
```



コード部分を全選択(Internet Explorerの場合)  
はCTRL + ALT + 左クリック;またはトリプルク  
リック)して、Bio-Linuxのターミナル画面上で  
ペーストしても構いません。スライドを見るだけ



# ちなみに…

## 事前準備

~/Documents/srp017156上に20160804ディレクトリを作成し、そこにgzip圧縮後のFaQCs実行結果ファイルをコピーして解析を行う。

```
cd ~/Documents/srp017156
```

```
mkdir 20160804
```

```
cd 20160804
```

```
pwd
```

```
#FaQCs実行結果ファイルの
```

```
cp ~/Documents/srp0171!
```

```
#cp ~/Desktop/backup/QC
```

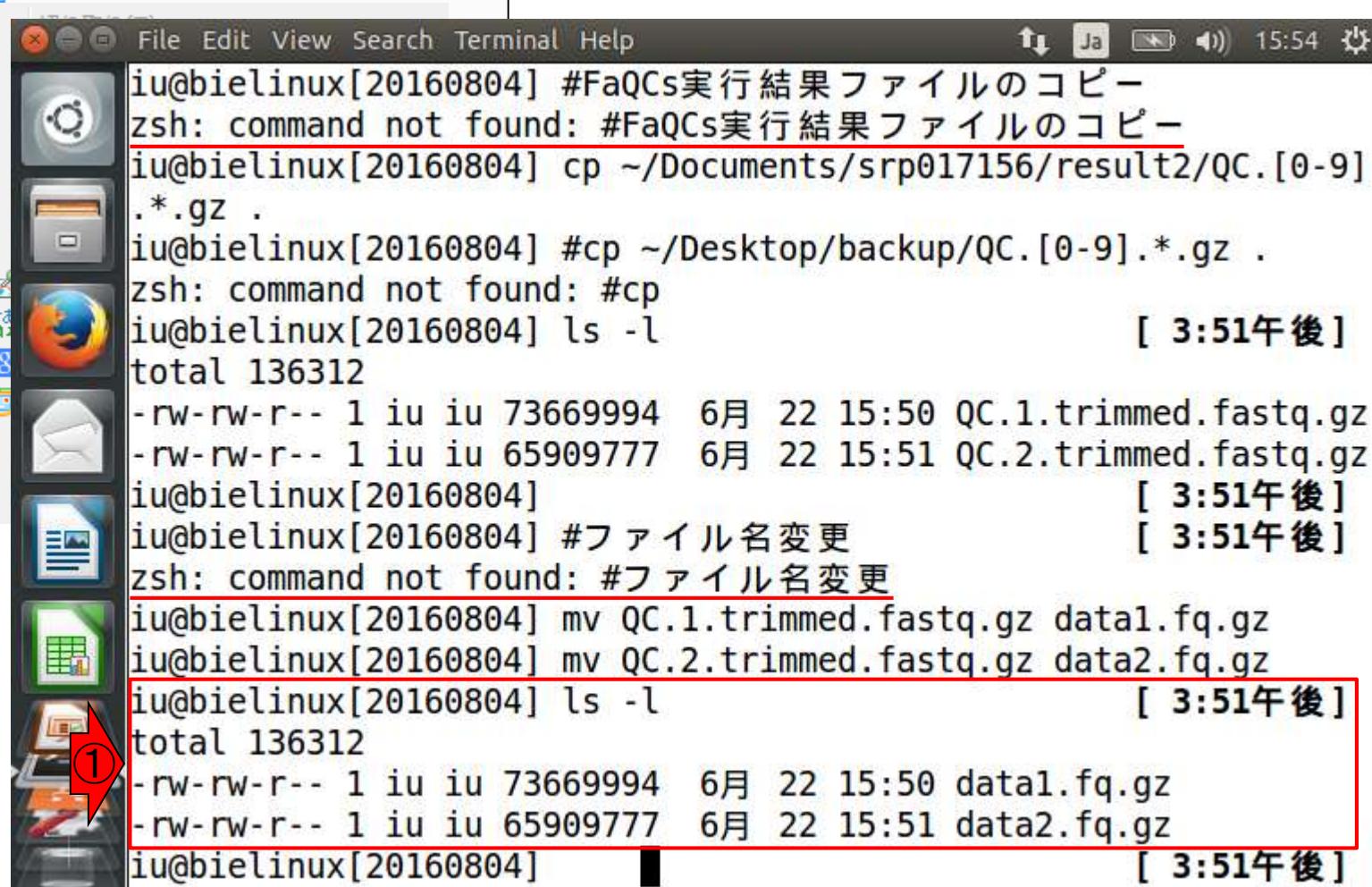
```
ls -l
```

```
#ファイル名変更
```

```
mv QC.1.trimmed.fastq.
```

```
mv QC.2.trimmed.fastq.
```

```
ls -l
```



The terminal window shows the command history and output of the user's actions. A red arrow points to the number 1 at the bottom left of the terminal window, indicating the final state of the command line.

```
iu@bielinux[20160804] #FaQCs実行結果ファイルのコピー  
zsh: command not found: #FaQCs実行結果ファイルのコピー  
iu@bielinux[20160804] cp ~/Documents/srp017156/result2/QC.[0-9].*.gz .  
iu@bielinux[20160804] #cp ~/Desktop/backup/QC.[0-9].*.gz .  
zsh: command not found: #cp  
iu@bielinux[20160804] ls -l [ 3:51午後]  
total 136312  
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 QC.1.trimmed.fastq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 QC.2.trimmed.fastq.gz  
iu@bielinux[20160804] [ 3:51午後]  
iu@bielinux[20160804] #ファイル名変更 [ 3:51午後]  
zsh: command not found: #ファイル名変更  
iu@bielinux[20160804] mv QC.1.trimmed.fastq.gz data1.fq.gz  
iu@bielinux[20160804] mv QC.2.trimmed.fastq.gz data2.fq.gz  
iu@bielinux[20160804] ls -l [ 3:51午後]  
total 136312  
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz  
iu@bielinux[20160804] [ 3:51午後]
```

赤下線部分で示されているように、#から始まる行の部分で「そんなコマンドはない」と怒られるだけで、①最後の状態は同じです

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# FastQC

## • FastQC

FastQCを実行し、結果を共有フォルダ(`~/Desktop/mac_share`)に保存する。実体はFastQC(ver. 0.11.4)であり、第4回W9のリードの場合に10番目以降の塩基を一定幅で処理するオプション(第5回W19-1)。約30秒。

(Bio-Linuxのコピペ実行結果のターミナル画面だと見づらいので)  
左上のコピペ用コード部分で説明。  
①-qは、途中経過を表示させないオプション。画面が流れで全体像がわからなくなるのでつけて  
いるだけ。通常利用時はつけない。  
②--nogroupオプションをつけることでKmer Contents項目の結果を正確に評価する(のが目的)。  
③FastQC実行結果は、共有フォルダに保存するよう指定している

```
cd ~/Documents/srp017156/20160804
```

```
pwd
ls -l
fastqc2 -v ②
fastqc2 -q --nogroup data1.fq.gz --outdir=/home/iu/Desktop/mac_share
fastqc2 -q --nogroup data2.fq.gz --outdir=/home/iu/Desktop/mac_share
ls -l ~/Desktop/mac_share/data*
```

# FastQC

赤枠部分のコピペ実行結果はこんな感じ。赤下線部分のhtmlファイルがFastQC 実行結果。<sup>①</sup>forward側、<sup>②</sup>reverse側

## • FastQC

FastQCを実行し、結果を共有フォルダ([~/Desktop/mac\\_share](#))に出力する。fastqc2コマンドの実体はFastQC (ver. 0.11.4)であり、第4回W9-5でパスを通しています。[--nogroup](#)は、「長いリードの場合に10番目以降の塩基に対するオプション(第5回W19-1)」

```
cd ~/Documents/srp017156/
```

```
pwd
```

```
ls -l
```

```
fastqc2 -v
```

```
fastqc2 -q --nogroup data1.fq.gz
```

```
fastqc2 -q --nogroup data2.fq.gz
```

```
ls -l ~/Desktop/mac_share
```

The screenshot shows a terminal window with a dark background and light-colored text. On the left, there's a vertical dock with icons for a terminal, file browser, and other applications. The terminal itself has a timestamped history:

- [4:20午後] `iu@bielinux[20160804] pwd`
- [4:20午後] `/home/iu/Documents/srp017156/20160804`
- [4:20午後] `iu@bielinux[20160804] ls -l`
- [4:20午後] `total 136320`
- [4:20午後] `-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz`
- [4:20午後] `-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz`
- [4:20午後] `iu@bielinux[20160804] fastqc2 -v`
- [4:20午後] `FastQC v0.11.4`
- [4:20午後] `iu@bielinux[20160804] fastqc2 -q --nogroup data1.fq.gz --outdir`
- [4:20午後] `=/home/iu/Desktop/mac_share`
- [4:20午後] `iu@bielinux[20160804] fastqc2 -q --nogroup data2.fq.gz --outdir`
- [4:20午後] `=/home/iu/Desktop/mac_share`
- [4:20午後] `iu@bielinux[20160804] ls -l ~/Desktop/mac_share/data*`
- [4:20午後] `-rwxrwxrwx 1 iu iu 421637 6月 22 16:20 /home/iu/Desktop/mac_share/data1_fastqc.html` (marked with red arrow ①)
- [4:20午後] `-rwxrwxrwx 1 iu iu 46123 6月 22 16:20 /home/iu/Desktop/mac_share/data1_fastqc.zip` (marked with red arrow ①)
- [4:20午後] `-rwxrwxrwx 1 iu iu 392817 6月 22 16:20 /home/iu/Desktop/mac_share/data2_fastqc.html` (marked with red arrow ②)
- [4:20午後] `-rwxrwxrwx 1 iu iu 42128 6月 22 16:20 /home/iu/Desktop/mac_share/data2_fastqc.zip` (marked with red arrow ②)
- [4:21午後] `iu@bielinux[20160804]`

# FastQC実行結果f

共有フォルダ上にある①forward側(data1.fq.gz)のFastQC実行結果を眺める。②リード数は977,202個。③リード長は50–107 bpの範囲。④目的はKmer Content項目。赤枠の部分が赤色なので「立ち止まって眺めるべきところ(第6回W4–2)」

C:\Users\kadota\Desktop\share\data1\_fastqc.html

Wed 22 Jun 2016  
data1.fq.gz

## FastQC Report

### Summary

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

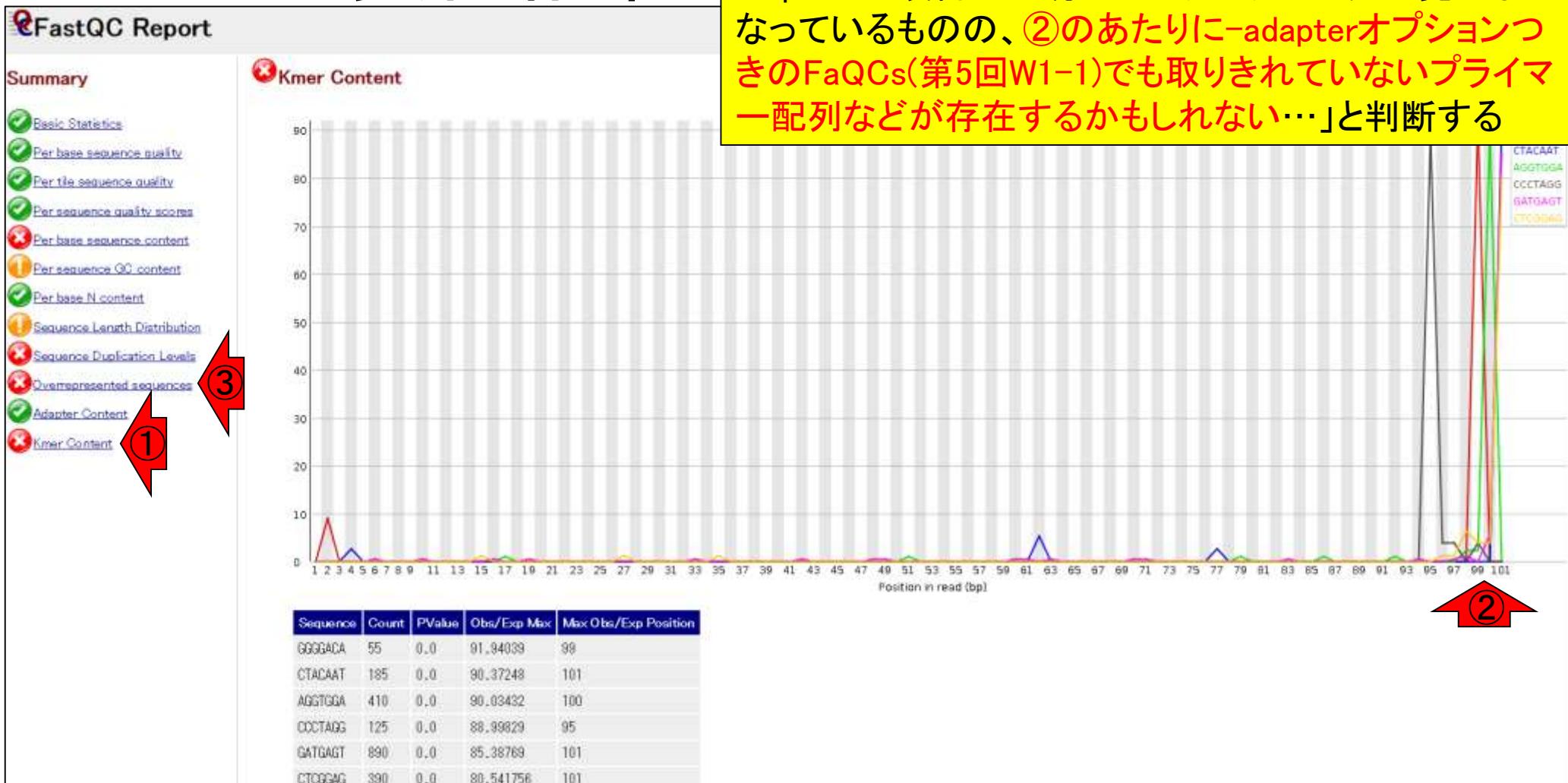
### Basic Statistics

Measure	Value
Filename	data1.fq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	977202
Sequences flagged as poor quality	0
Sequence length	50-107
%GC	50

### Per base sequence quality

Produced by FastQC (version 0.11.4)

# FastQC実行結果f



# FastQC実行結果

①reverse側(data2.fq.gz)のFastQC実行結果を眺める。  
 ②リード数は977,202個。③リード長は50–93 bpの範囲。④目的はKmer Content項目。赤枠の部分が赤色なので「立ち止まって眺めるべきところ(第6回W4-2)」

C:\Users\kadota\Desktop\share\data2\_fastqc.html

data2.fq.gz FastQC Report

Wed 22 Jun 2016  
data2.fq.gz

## FastQC Report

### Summary

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

### Basic Statistics

Measure	Value
Filename	data2.fq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	977202
Sequences flagged as poor quality	0
Sequence length	50-93
%GC	49

### Per base sequence quality

Produced by [FastQC](#) (version 0.11.4)

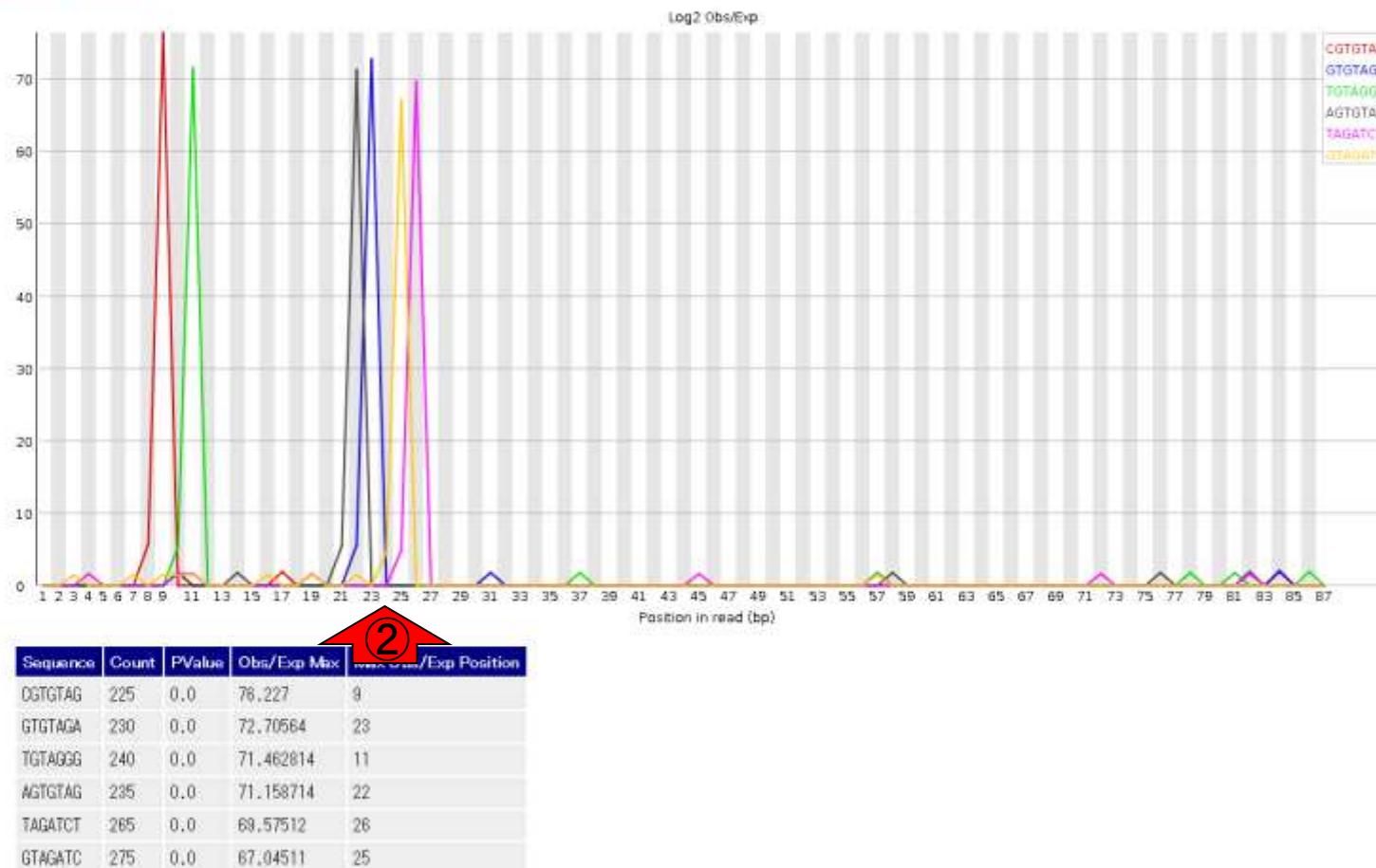
# FastQC実行結果

## FastQC Report

### Summary

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

### Kmer Content



①Kmer Content項目の全体像。②のあたりにピークがあるものの、リードの両端にはないようなので「reverse側はたぶん大丈夫だろう(解析サンプルに由来しない配列はなさそう)」と判断する

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# Rockhopper2おさらい

## • Rockhopper2おさらい

最大メモリを2GBにして paired-end で実行(第5回W5-2)。約1分。

```
cd ~/Documents/srp017156/20160804
```

```
pwd  
ls -l  
java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz
```

```
ls  
ls -l Rockhopper_Results
```

#ファイルの行頭と行末の8行分を表示

```
head -n8 Rockhopper_Results/summary.txt  
tail -n8 Rockhopper_Results/summary.txt
```

paired-end (data1.fq.gz と data2.fq.gz) で Rockhopper2 を実行し、以前と同じ結果(第5回 W5-2)になるかを確認しておく。赤枠内をコピー。① paired-end (data1.fq.gz と data2.fq.gz) としてメモリを 2GB (2000MB) にして実行するところ。2016.08.02 欠席者はクラスパスの設定ができないのでエラーが出ると思われます。2016.08.02 のスライド 60あたり(第5回 W17)を参考にして ~/.zshrc ファイルをいじってください

①

# Rockhopper2おさらい

iu@bielinux[~/Documents/srp017156/20160804]

```
iu@bielinux[20160804] pwd  
/home/iu/Documents/srp017156/20160804  
iu@bielinux[20160804] ls -l  
total 136320  
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz  
iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz
```

Assembling transcripts from reads in files:  
  data1.fq.gz  
  data2.fq.gz

Aligning reads to assembled transcripts using files:  
  data1.fq.gz  
  data2.fq.gz

Total reads in files:	976468
Perfectly aligned reads:	35      0%

①paired-end (data1.fq.gzとdata2.fq.gz)としてメモリを2GBにして実行。②赤枠がターミナル上に出力されている途中経過。③のあたりの数値は、第5回W5-2と同じなので安心

[ 9:57 午前 ]

[ 9:58 午前 ]

①

②

③

# Rockhopper2おさらい

```
File Edit View Search Terminal Help
Total number of assembled transcripts: 0
    Average transcript length: 0
    Median transcript length: 0
    Total number of assembled bases: 0

Summary of results written to file: Rockhopper_Result
s/summary.txt
Details of assembled transcripts written to file: Rockhopper_Result
s/transcripts.txt
FINISHED. ①

iu@bielinux[20160804] ls [10:03午前]
data1.fq.gz data2.fq.gz Rockhopper_Results
iu@bielinux[20160804] ls -l Rockhopper_Results [10:04午前]
total 16
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 genomeBrowserFiles
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 intermediary
-rw-rw-r-- 1 iu iu 580 6月 23 10:03 summary.txt
-rw-rw-r-- 1 iu iu 29 6月 23 10:03 transcripts.txt
iu@bielinux[20160804] [10:04午前]
```

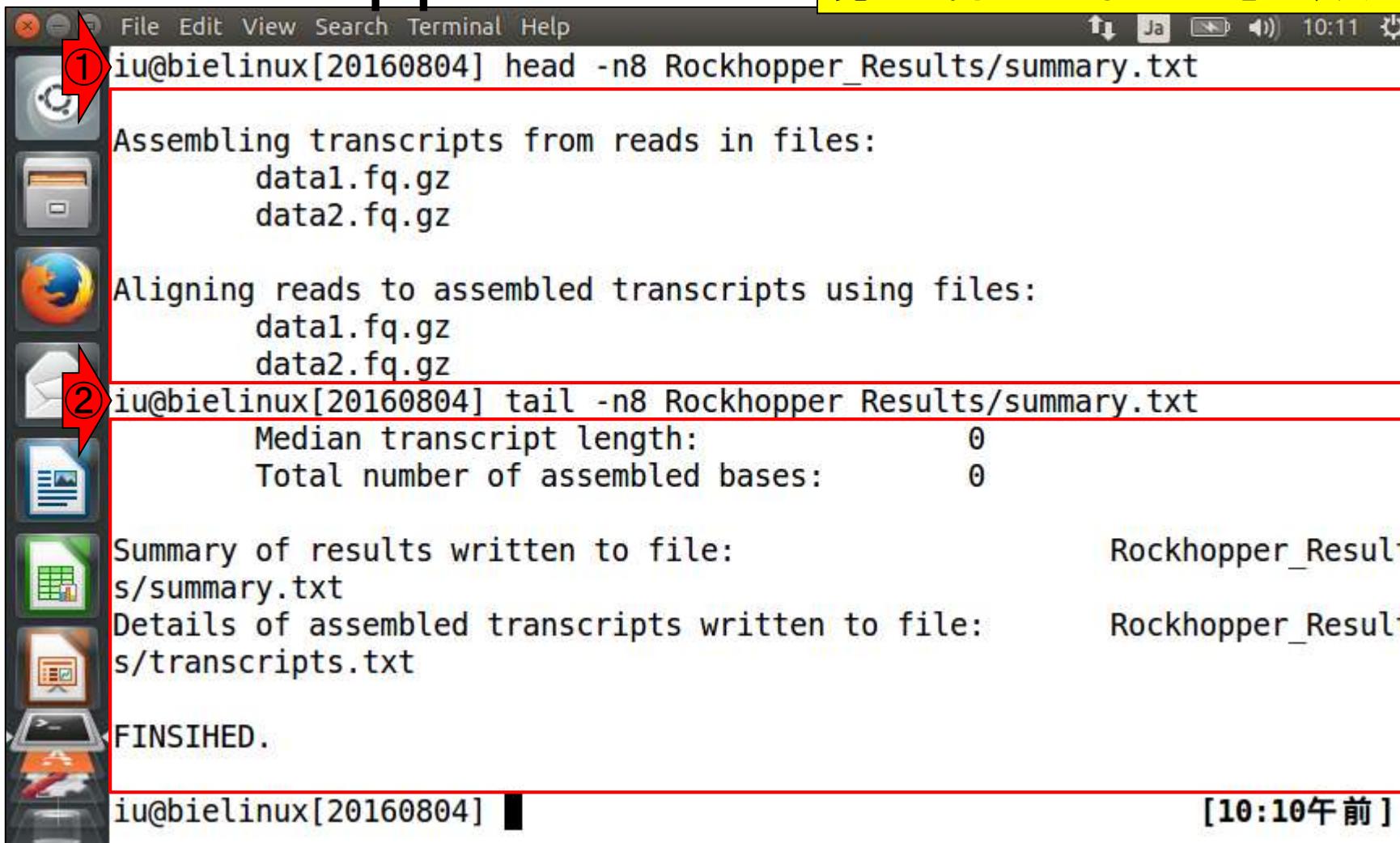
③transcripts.txtがアセンブリ結果、④summary.txtが赤枠とほぼ同じ内容でした

# Rockhopper2おさらい

```
File Edit View Search Terminal Help  
Total number of assembled transcripts: 0  
    Average transcript length: 0  
    Median transcript length: 0  
    Total number of assembled bases: 0  
  
Summary of results written to file: Rockhopper_Result  
s/summary.txt  
Details of assembled transcripts written to file: Rockhopper_Result  
s/transcripts.txt  
  
FINISHED.  
  
iu@bielinux[20160804] ls [10:03午前]  
data1.fq.gz data2.fq.gz Rockhopper_Results  
iu@bielinux[20160804] ls -l Rockhopper_Results [10:04午前]  
total 16  
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 genomeBrowserFiles  
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 intermediate  
-rw-rw-r-- 1 iu iu 580 6月 23 10:03 summary.txt ④  
-rw-rw-r-- 1 iu iu 29 6月 23 10:03 transcripts.txt ③  
iu@bielinux[20160804] [10:04午前]
```

summary.txtの①行頭と②行末の8行分を表示。  
Rockhopper2実行時のターミナル画面上の出力と  
完全に同じではないかと思い、次のスライドで確認

# Rockhopper2おさら



The screenshot shows a terminal window on an Ubuntu desktop. The terminal output is as follows:

```
iu@bielinux[20160804] head -n8 Rockhopper_Results/summary.txt
Assembling transcripts from reads in files:
    data1.fq.gz
    data2.fq.gz

Aligning reads to assembled transcripts using files:
    data1.fq.gz
    data2.fq.gz

iu@bielinux[20160804] tail -n8 Rockhopper_Results/summary.txt
    Median transcript length:          0
    Total number of assembled bases:  0

Summary of results written to file: Rockhopper_Result
s/summary.txt
Details of assembled transcripts written to file: Rockhopper_Result
s/transcripts.txt

FINISHED.

iu@bielinux[20160804] [10:10午前]
```

The terminal window has a red border around its content area. Two red arrows point to specific parts of the terminal window: one arrow labeled '1' points to the top line of the first block of text ('Assembling transcripts...'), and another arrow labeled '2' points to the top line of the second block of text ('Median transcript length: 0').

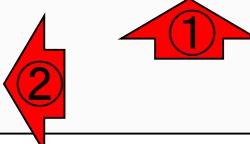
# Tips : diff

## • Tips: diff

Rockhopper実行時に画面上に出力されるものをresult.txtに保存してスッキリさせる。  
比較する2つのファイルの差分を表示させるdiffコマンドを実行し、何も表示されない  
(全く同じ)ことを確認。

```
cd ~/Documents/srp017156/20160804

pwd
ls -l
java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz > result.txt
ls
ls -l Rockhopper_Results
diff result.txt Rockhopper_Results/summary.txt
```



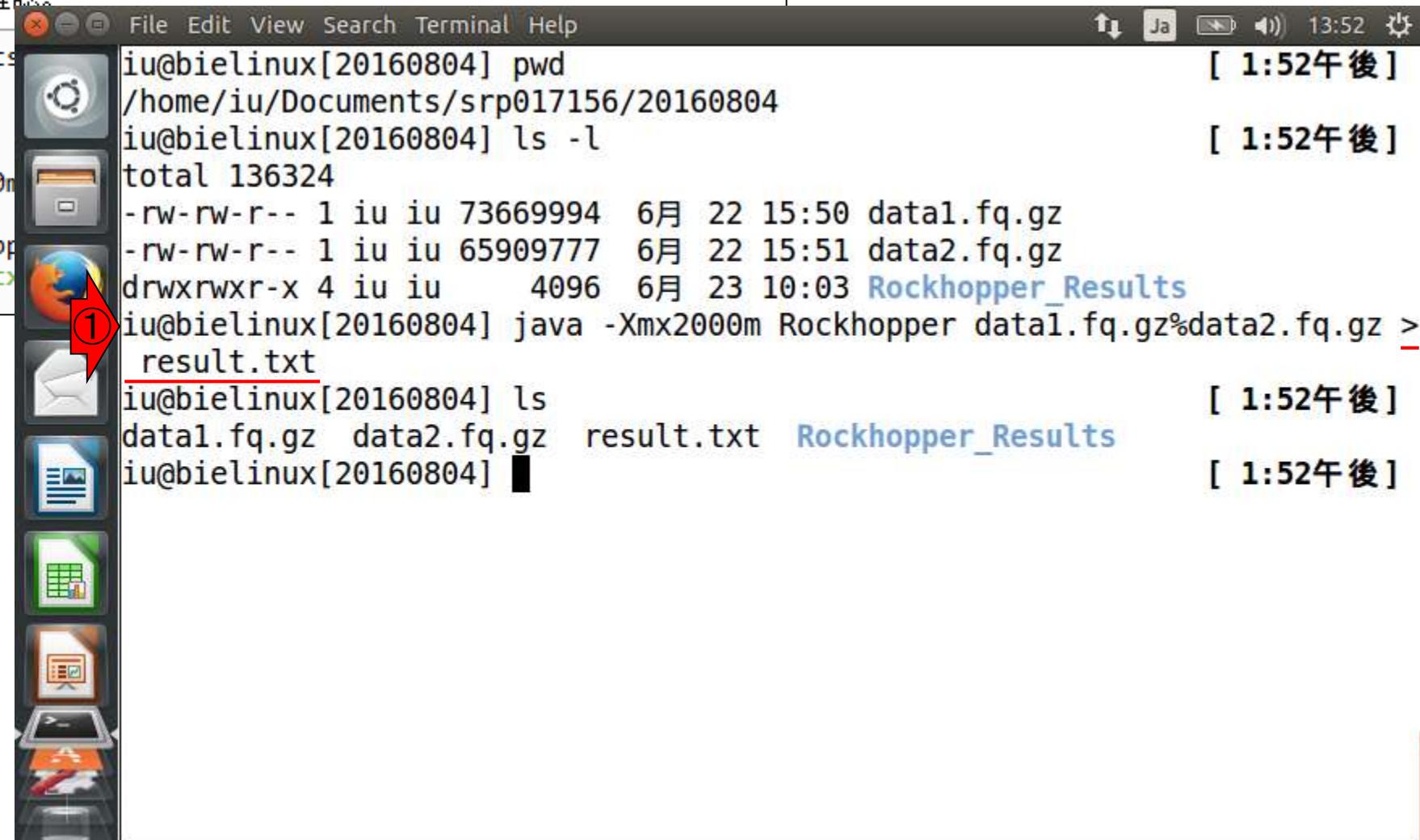
①Rockhopper2実行時にターミナル画面上に出力されるものをresult.txtに保存。②diffコマンドは2つのファイルの差分を出力する。比較している2つのファイルが全く同じ場合には何も起こらない

①Rockhopper2実行時にターミナル画面上に出力されるものをresult.txtに保存するところまで。計算は一瞬で終わるがresult.txtは確かにできている

# Tips: diff

## • Tips: diff

Rockhopper実行時に画面上に出力されるものをresult.txtに保存してスッキリさせる。  
比較する2つのファイルの差分を表示させるdiffコマンドを実行し、何も表示されない  
(全く同じ)ことを確認。



The screenshot shows a terminal window with the following session:

```

cd ~/Documents
pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l
total 136324
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
drwxrwxr-x 4 iu iu     4096 6月 23 10:03 Rockhopper_Results
iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz >
result.txt
iu@bielinux[20160804] ls
data1.fq.gz  data2.fq.gz  result.txt  Rockhopper_Results
iu@bielinux[20160804]

```

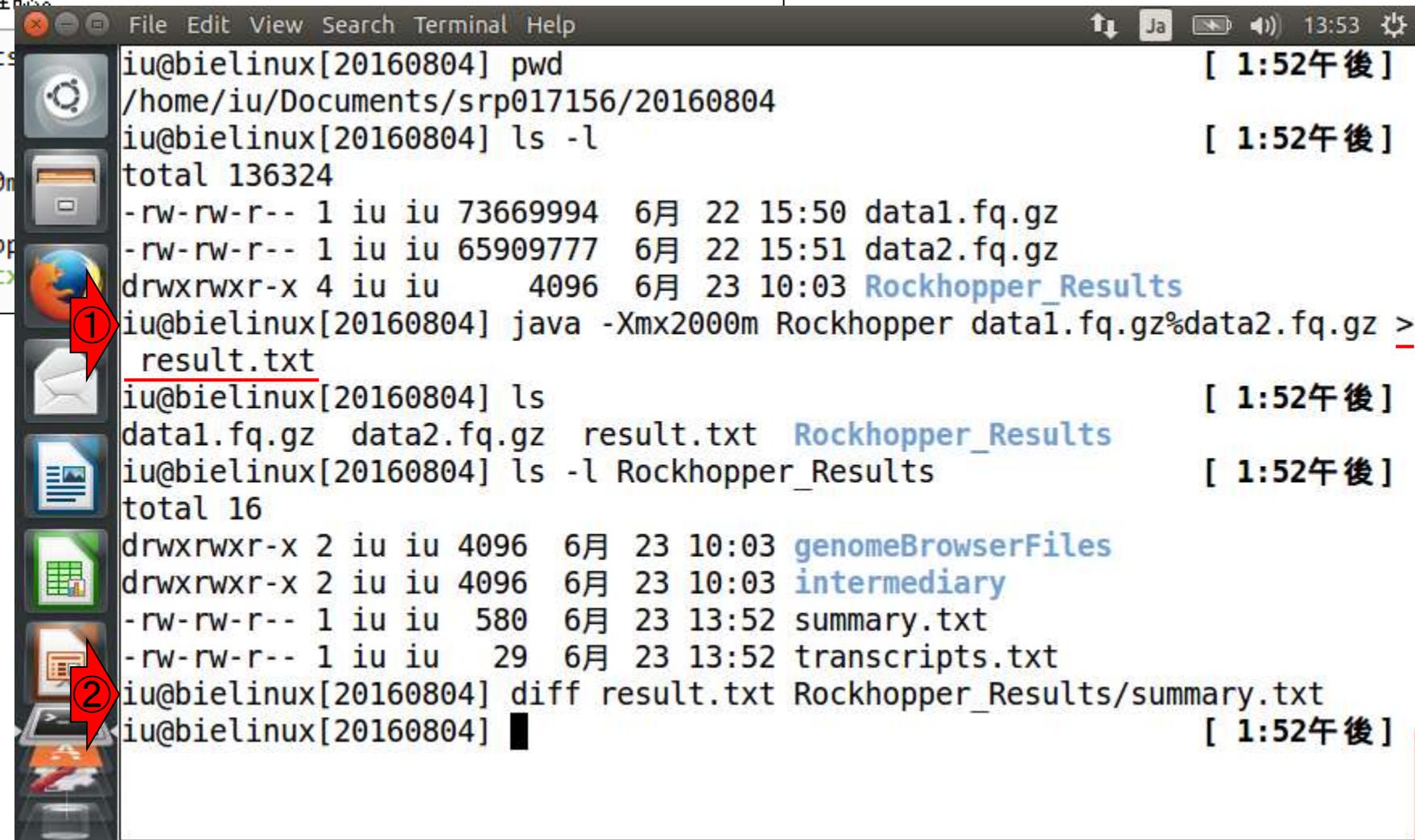
A red arrow labeled ① points to the terminal icon in the dock at the bottom left.

②diff実行結果は何も起こらなかった。つまり、result.txtとsummary.txtの中身は全く同じ

# Tips: diff

## Tips: diff

Rockhopper実行時に画面上に出力されるものをresult.txtに保存してスッキリさせる。  
比較する2つのファイルの差分を表示させるdiffコマンドを実行し、何も表示されない  
(全く同じ)ことを確認。



The screenshot shows a terminal window with the following session:

```

cd ~/Documents
pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l
total 136324
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
drwxrwxr-x 4 iu iu 4096 6月 23 10:03 Rockhopper_Results
iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz > result.txt
iu@bielinux[20160804] ls
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] ls -l Rockhopper_Results
total 16
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 genomeBrowserFiles
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 intermediary
-rw-rw-r-- 1 iu iu 580 6月 23 13:52 summary.txt
-rw-rw-r-- 1 iu iu 29 6月 23 13:52 transcripts.txt
iu@bielinux[20160804] diff result.txt Rockhopper_Results/summary.txt
iu@bielinux[20160804]

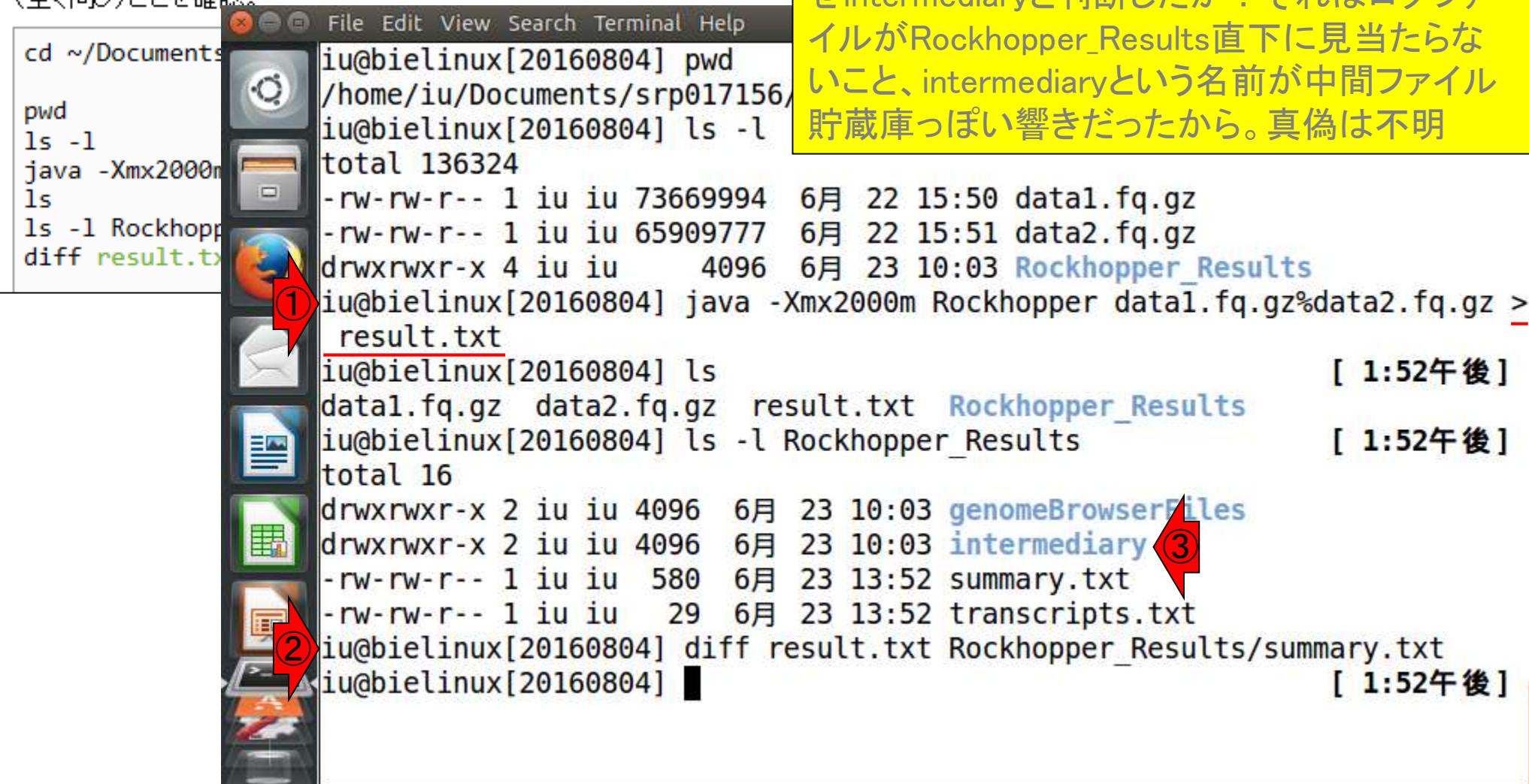
```

Two red arrows point to the icons for 'result.txt' (an envelope icon) and 'summary.txt' (a document icon) in the file list on the left side of the terminal window.

# Tips: diff

## • Tips: diff

Rockhopper実行時に画面上に出力されるものをresult.txtに保存して比較する2つのファイルの差分を表示させるdiffコマンドを実行し、何も表示(全く同じ)ことを確認。



The terminal window shows the following session:

```

cd ~/Documents
pwd
ls -l
java -Xmx2000m Rockhopper
diff result.txt

```

Output:

```

iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156
iu@bielinux[20160804] ls -l
total 136324
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
drwxrwxr-x 4 iu iu 4096 6月 23 10:03 Rockhopper_Results
iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz > result.txt
iu@bielinux[20160804] ls
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] ls -l Rockhopper_Results
total 16
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 genomeBrowserFiles
drwxrwxr-x 2 iu iu 4096 6月 23 10:03 intermediary
-rw-rw-r-- 1 iu iu 580 6月 23 13:52 summary.txt
-rw-rw-r-- 1 iu iu 29 6月 23 13:52 transcripts.txt
iu@bielinux[20160804] diff result.txt Rockhopper_Results/summary.txt
iu@bielinux[20160804]

```

Annotations:

- ① A red arrow points to the terminal icon in the dock.
- ② A red arrow points to the file icon in the dock.
- ③ A red arrow points to the "intermediary" folder in the listing.

[ 1:52 午後 ] [ 1:52 午後 ] [ 1:52 午後 ]

③ Rockhopper2の実行がほぼ一瞬で終わった  
理由は、おそらく最初にintermediary中のファイルなど以前の実行ログを見に行って、全く同じファイル名やパラメータのものがあれば、それを返すようにしているのであろうと解釈する。なぜintermediaryと判断したか？それはログファイルがRockhopper\_Results直下に見当たらぬこと、intermediaryという名前が中間ファイル貯蔵庫っぽい響きだったから。真偽は不明

# 確認

①以前の結果の影響をなくすため、一旦削除してから再実行。rm実行時のrオプションはディレクトリ削除時に利用(第3回W18-4; 第4回W7-7やW9-3)

## 確認

Rockhopper2の実行がほぼ一瞬で終わった理由は、おそらく最初にintermediaryなどの以前に実行したログ情報を見に行って、全く同じファイル名やパラメータのものがあれば、それを返すような仕様をしているのであろう。念には念を入れて、一旦 Rockhopper\_Resultsディレクトリを削除してもう一度実行する。それなりの実行時間がかかり、diff結果で何も出なければ安心。

```
cd ~/Documents/srp017156/20160804  
  
pwd  
ls  
rm -rf Rockhopper_Results  
rm -f result.txt  
①  
ls -l  
java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz > result.txt  
ls  
ls -l Rockhopper_Results  
diff result.txt Rockhopper_Results/summary.txt
```

赤枠以降のコードもコピペ。①(以前の実行結果が全てなくなっているので当たり前といえば当たり前だが)アセンブリ実行にそれなりに時間がかかった。②diffの結果は不变

# 確認

## 確認

Rockhopper2の実行がほぼ一瞬で終わった理由は、おそらく最初にintermediaryなどの以前に実行したログ情報を見に行って、全く同じファイル名やパラメータのものがあれば、それを返すような仕様をしているのであろう。今までは今を入れて一日

Rockhopper\_Results

かかり、diff結果で

cd ~/Documents

pwd

ls

rm -rf Rockhopper\_Results

rm -f result.txt

ls -l

java -Xmx2000m Rockhopper

ls

ls -l Rockhopper\_Results

diff result.txt

iu@bielinux[~/Documents/srp017156/20160804]

/home/iu/Documents/srp017156/20160804

iu@bielinux[20160804] ls

data1.fq.gz data2.fq.gz result.txt Rockhopper\_Results

iu@bielinux[20160804] rm -rf Rockhopper\_Results

iu@bielinux[20160804] rm -f result.txt

iu@bielinux[20160804] ls -l

total 136320

-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz

-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz

iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz%data2.fq.gz >

result.txt

iu@bielinux[20160804] ls

data1.fq.gz data2.fq.gz result.txt Rockhopper\_Results

iu@bielinux[20160804] ls -l Rockhopper\_Results

total 16

drwxrwxr-x 2 iu iu 4096 6月 23 14:16 genomeBrowserFiles

drwxrwxr-x 2 iu iu 4096 6月 23 14:17 intermediary

-rw-rw-r-- 1 iu iu 580 6月 23 14:17 summary.txt

-rw-rw-r-- 1 iu iu 29 6月 23 14:17 transcripts.txt

iu@bielinux[20160804] diff result.txt Rockhopper\_Results/summary.txt

iu@bielinux[20160804] ■

[ 2:16午後 ]

[ 2:16午後 ]

[ 2:16午後 ]

[ 2:16午後 ]

[ 2:17午後 ]

[ 2:17午後 ]



①

②

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 情報抽出(grep)

## 情報抽出(grep)

Rockhopper\_Results/summary.txtと同じ中身のファイル(result数(transcripts数)情報を抽出する。ここでは、"number of assembled transcripts"という文字列を含む行を表示させている。どういう文字列で抽出する眺めておいて試行錯誤するのが一般的。

```
cd ~/Documents/srp017156/20160804
```

```
pwd  
ls -l  
grep "number of assembled transcripts" result.txt
```

(このあたりはヒトによって大分流儀が異なるが)1つのアセンブリ結果から1行分のみで、全体像を把握したい場合にはgrepが便利(第3回W16-2; W18-1; W19; 第4回W9-9)。grepは、指定した文字列を含む行を出力するコマンド。ここではnumber of assembled transcriptsという文字列を含む行を表示させている。赤枠が実行結果

```
iu@bielinux[20160804] pwd  
/home/iu/Documents/srp017156/20160804  
iu@bielinux[20160804] ls -l  
total 136328  
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz  
-rw-rw-r-- 1 iu iu 580 6月 23 14:17 result.txt  
drwxrwxr-x 4 iu iu 4096 6月 23 14:17 Rockhopper_Results  
iu@bielinux[20160804] grep "number of assembled transcripts" result.txt  
Total number of assembled transcripts: 0  
iu@bielinux[20160804]
```

# 情報抽出2(grep)

①single-endでforward側(data1.fq.gz)のみ、reverse側(data2.fq.gz)のみのRockhopper2を実行し、②配列数(転写物数)を表示

## 情報抽出2(grep)

single-endでforward側(data1.fq.gz)のみ、reverse側(data2.fq.gz)のみのRockhopper2を実行し、配列数を表示。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l
```

```
java -Xmx2000m Rockhopper data1.fq.gz > result_f.txt
```

```
java -Xmx2000m Rockhopper data2.fq.gz > result_r.txt
```

```
ls -l
```

```
grep "number of assembled transcripts" result*.txt
```

①

②

# 情報抽出2(arep)

- 情報抽出2(grep)  
single-endでforward  
実行し、配列数を表示
- cd ~/Documents
- pwd
- ls -l
- java -Xmx2000m Rockhopper
- java -Xmx2000m Rockhopper
- ls -l
- grep "number of assembled transcripts"

```

File Edit View Terminal Help
iu@bielinux[20160804] pwd [ 3:05午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l [ 3:06午後]
total 136328
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
-rw-rw-r-- 1 iu iu      580 6月 23 14:17 result.txt
drwxrwxr-x 4 iu iu     4096 6月 23 14:17 Rockhopper_Results
iu@bielinux[20160804] java -Xmx2000m Rockhopper data1.fq.gz > result_f.txt
iu@bielinux[20160804] java -Xmx2000m Rockhopper data2.fq.gz > result_r.txt [ 3:07午後]
iu@bielinux[20160804] ls -l
total 136336
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
-rw-rw-r-- 1 iu iu      568 6月 23 15:07 result_f.txt
-rw-rw-r-- 1 iu iu      577 6月 23 15:07 result_r.txt
-rw-rw-r-- 1 iu iu      580 6月 23 14:17 result.txt
drwxrwxr-x 4 iu iu     4096 6月 23 14:17 Rockhopper_Results
iu@bielinux[20160804] grep "number of assembled transcripts" result*.txt
result_f.txt:Total number of assembled transcripts: 1
result_r.txt:Total number of assembled transcripts: 423
result.txt:Total number of assembled transcripts: 0
iu@bielinux[20160804] [ 3:09午後]

```

①

②

# 情報抽出2(arep)

## 情報抽出2(grep)

single-endでforward側(data1.fq.gz)のみ、reverse側(data2.fq.gz)のみ実行し、配列数を表示。

```
cd ~/Documents/srp017156/20160804
```

```
pwd  
ls -l  
java -Xmx2000m Rockhopper data1.fq.gz > result_f.txt  
java -Xmx2000m Rockhopper data2.fq.gz > result_r.txt  
ls -l  
grep "number of assembled transcripts" result*.txt
```

:50 data1.fq.gz  
:51 data2.fq.gz  
:17 result.txt  
:17 Rockhopper\_Results  
ockhopper data1.fq.gz > result\_f.tx

iu@bielinux[20160804] java -Xmx2000m Rockhopper data2.fq.gz > result\_r.txt [ 3:07午後 ]  
iu@bielinux[20160804] ls -l  
total 136336  
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz  
-rw-rw-r-- 1 iu iu 568 6月 23 15:07 result\_f.txt  
-rw-rw-r-- 1 iu iu 577 6月 23 15:07 result\_r.txt  
-rw-rw-r-- 1 iu iu 580 6月 23 14:17 result.txt  
drwxrwxr-x 4 iu iu 4096 6月 23 14:17 Rockhopper\_Results  
iu@bielinux[20160804] grep "number of assembled transcripts" result\*.txt  
result\_f.txt:Total number of assembled transcripts: 1  
result\_r.txt:Total number of assembled transcripts: 423  
result.txt:Total number of assembled transcripts: 0  
iu@bielinux[20160804] [ 3:09午後 ]

# これまでのまとめ

## ■ de novoトランスクリプトームアセンブリ

- バクテリア用のアセンブリRockhopper2を実行して変な結果に遭遇した
  - paired-end (**data1.fq.gz**と**data2.fq.gz**) : 0 transcript
  - single-end (forward側のみ; **data1.fq.gz**) : 1 transcript
  - single-end (reverse側のみ; **data2.fq.gz**) : 423 transcripts

## ■ FastQC

- forward側(**data1.fq.gz**)は、99–101塩基目あたりにk-merのピークが集中していた
- reverse側(**data2.fq.gz**)は、リードの両端付近にはk-merのピークはなかった

**目的:** *de novo* transcriptome assemblyでそれなりのアセンブリ結果を得る。**前提条件:** ゲノム配列は未知。**手段:** 末端塩基をトリムしてアセンブリ、を繰り返し行う。**評価基準:** 配列数

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 当面の目標

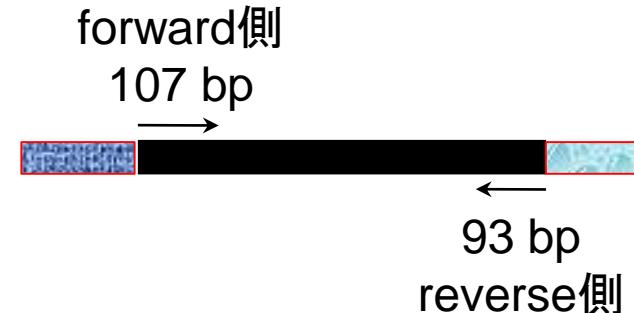
## ■ *de novo*トランスクript

- バクテリア用のアセンブラRockhopper2を実行して変な結果に遭遇した
  - paired-end (data1.fq.gzとdata2.fq.gz) : 0 transcript
  - single-end (forward側のみ; data1.fq.gz) : 1 transcript
  - single-end (reverse側のみ; data2.fq.gz) : 423 transcripts



## ■ FastQC

- forward側(data1.fq.gz)は、99–101塩基目あたりにk-merのピークが集中していた
- reverse側(data2.fq.gz)は、リードの両端付近にはk-merのピークがなかった



トリミングは、fastx\_trimmerを利用する。とりあえず第5回W16-2と同じようなコードで動作確認をしておさらい。①がfastx-trimmer実行部分。赤枠部分をコピペ

# トリミング

## ・ トリミング (fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) 99塩基目まで残すという指定で実行し、trim1.fq.gzというファイル名で保存。

```
cd ~/Documents/srp017156/20160804
```

```
pwd  
ls -l data1*  
gunzip -c data1.fq.gz | head -n 4  
gunzip -c data1.fq.gz | fastx_trimmer -l 99 - | gzip > trim1.fq.gz  
gunzip -c trim1.fq.gz | head -n 4
```

```
fastx_trimmer -h
```

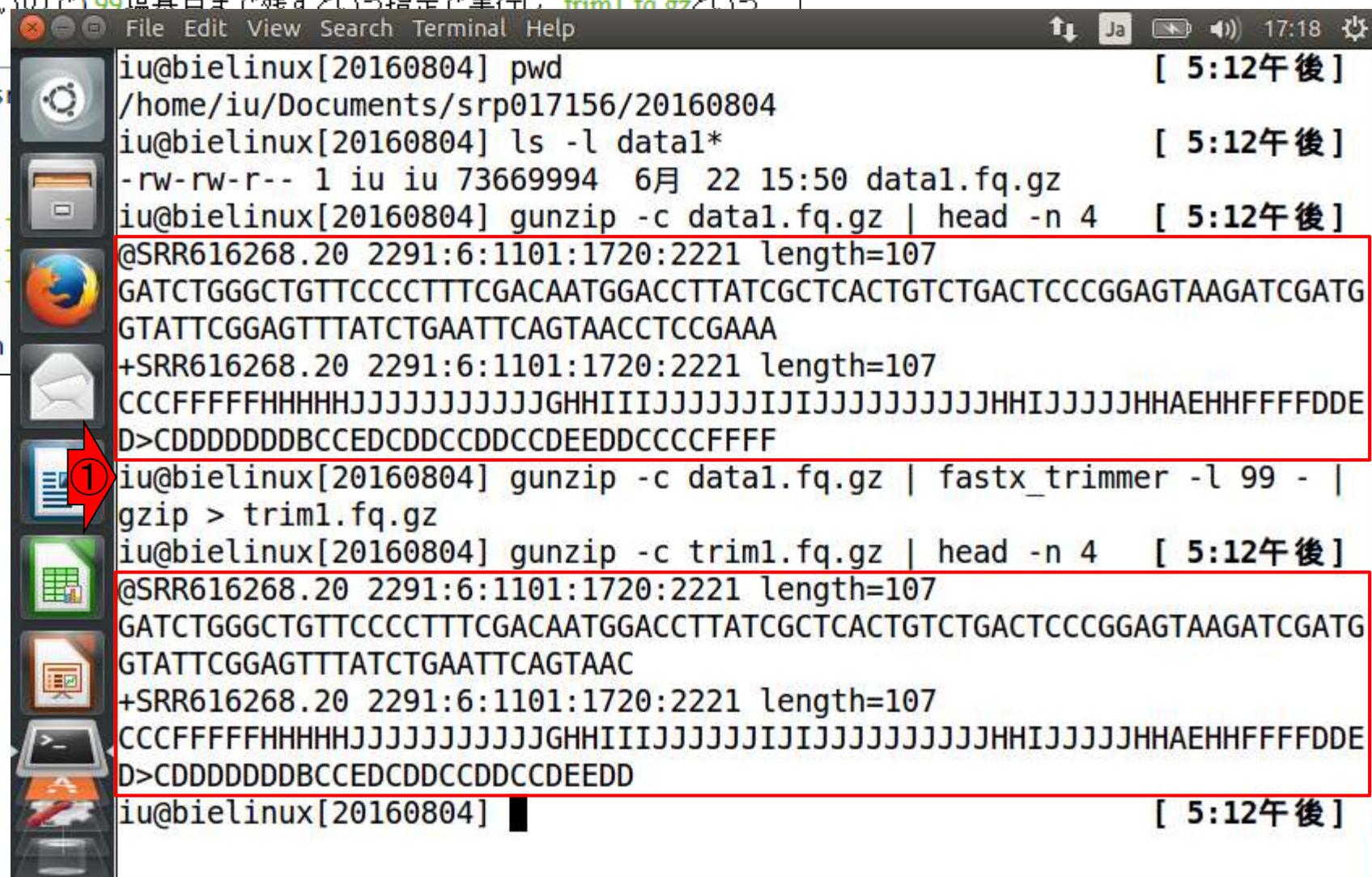
①

# トリミング

- ・トリミング(fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) 99塩基目まで残すという指定を実行し、trim1.fq.gzというファイル名で保存。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data1*
gunzip -c data1.fq.gz
gunzip -c data1.fq.gz | head -n 4
gunzip -c trim1.fq.gz
fastx_trimmer -h
```



```
iu@bielinux[20160804] pwd [ 5:12 午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1* [ 5:12 午後]
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
iu@bielinux[20160804] gunzip -c data1.fq.gz | head -n 4 [ 5:12 午後]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGCTGTTCCCCTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCGGAGTAAGATCGATG
GTATTGGAGTTATCTGAATTCACTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJJJHJIJJJJJJHHAEEHHFFFFDDE
D>CDDDDDDDDDBCCEDCDDCCDDCCDEEDCCCCFFFFF
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -l 99 - |
gzip > trim1.fq.gz
iu@bielinux[20160804] gunzip -c trim1.fq.gz | head -n 4 [ 5:12 午後]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGCTGTTCCCCTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCGGAGTAAGATCGATG
GTATTGGAGTTATCTGAATTCACTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJJJHJIJJJJJJHHAEEHHFFFFDDE
D>CDDDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 5:12 午後]
```

# トリミング

- トリミング (fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2で取得した100個のFASTQデータをトリムしたいので) `gg` 基本まで残すという指定で実行。`trim1.fq.gz`というファイル名で保存。

トリミング①実行前と②実行後の最初の4行分(つまり1リード分の情報)を比較。10塩基ごとに赤の縦棒を入れている。確かに99塩基目まで残すという(-l 99)オプション通りの結果になっていることがわかる

```
cd ~/Documents/srp017156/20160804  
pwd  
ls -l data1*  
gunzip -c data1.fq.gz | head -n 4 [ 5:12午後]  
gunzip -c data1.fq.gz | head -n 4 [ 5:12午後]  
gunzip -c trim1.fq.gz | head -n 4 [ 5:12午後]  
fastx_trimmer -h  
iu@bielinu[20160804] gunzip -c data1.fq.gz | fastx_trimmer -l 99 - |  
gzip > trim1.fq.gz  
iu@bielinu[20160804] gunzip -c trim1.fq.gz | head -n 4 [ 5:12午後]  
iu@bielinu[20160804] gunzip -c trim1.fq.gz | head -n 4 [ 5:12午後]
```

ちなみに①の部分はただの文字列なので、トリミング前後で不变(第5回W16-4)

# トリミング

- トリミング (fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) go 執行日まで残すという指定で実行し、`trim1.fa.gz`というファイル名で保存。

```
cd ~/Documents/si  
pwd  
ls -l data1*  
gunzip -c data1.  
gunzip -c data1.  
gunzip -c trim1.  
  
fastx_trimmer -h
```

```
iu@bielinux[20160804] pwd [ 5:12午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1*
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
iu@bielinux[20160804] gunzip -c data1.fq.gz | head -n 4 [ 5:12午後]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGT ACTCCCGAGTAAGATCGATG
GTATTGGAGTTATCTGAATTCACTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJJJJJHJIJJJJJJHHAEEHHFFFFDDE
D>CDDDDDDDBCCEDCDDCCDEEDDCCCCFFFF
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -l 99 - |
gzip > trim1.fq.gz
iu@bielinux[20160804] gunzip -c trim1.fq.gz | head -n 4 [ 5:12午後]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGT ACTCCCGAGTAAGATCGATG
GTATTGGAGTTATCTGAATTCACTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJJJJJHJIJJJJJJHHAEEHHFFFFDDE
D>CDDDDDDDBCCEDCDDCCDEEDD
iu@bielinux[20160804] [ 5:12午後]
```

# fastx-trimmer

トリミングの主なターゲットは、①リードの3'末端なので、②I(エル)オプションは必須。しかし、リードの5'末端もトリムして様々な組み合わせを試したいので…

- トリミング (fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) 99塩基目まで残すという指定で実行し、trim1.fq.gzというファイル名で保存。

```
cd ~/Documents/si  
pwd  
ls -l data1*  
gunzip -c data1.  
gunzip -c data1.  
gunzip -c trim1.  
  
fastx trimmer -h
```

①「fastx-trimmer -h」でオプションの利用法を眺める。②-fが5'末端側のトリムに相当すると判断

# fastx-trimmer

## ・トリミング(fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) 基本まで残すという指定を実行(trim1.fq.gz)というファイル名で保存。

```
cd ~/Documents/someplace
pwd
ls -l data1*
gunzip -c data1.fq.gz > trim1.fq.gz
gunzip -c data1.fq.gz > trim1.fq
fastx_trimmer -h
```

```
iu@bielinux[20160804] fastx_trimmer -h [ 5:52午後]
usage: fastx_trimmer [-h] [-f N] [-l N] [-t N] [-m MINLEN] [-z] [-v] [-i INFIL
E] [-o OUTFILE]
Part of FASTX Toolkit 0.0.14 by A. Gordon (assafgordon@gmail.com)

[-h]      = This helpful help screen.
[-f N]    = First base to keep. Default is 1 (=first base).
[-l N]    = Last base to keep. Default is entire read.
[-t N]    = Trim N nucleotides from the end of the read.
           '-t' can not be used with '-l' and '-f'.
[-m MINLEN] = With [-t], discard reads shorter than MINLEN.
[-z]      = Compress output with GZIP.
[-i INFIL
E]    = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

iu@bielinux[20160804] [ 5:52午後]
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# fastx-trimmer -f -l

• fastx-trimmer -f -l

「-f 3」を追加したら、「-l 99」はどうなるのかを調査。つまり、「-f 3」は3塩基目から残すという意味なので、最初の2塩基分がトリムされる。この場合に「-l 97」としないといけないのかどうかを知りたい。

```
cd ~/Documents/srp017156/20160804  
  
pwd  
ls -l data1*  
gunzip -c data1.fq.gz | head -n 4  
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip > trim1.fq  
gunzip -c trim1.fq.gz | head -n 4
```



①「-f 3」は、3塩基目から残すという意味であり、最初の2塩基分をトリムすることに相当する。この場合、②の「-l 99」の指定が-fに影響されるのかどうかが気になる(結論としては独立)

コピペ実行結果。①「-f 3」で最初の2塩基分をトリムすることと、②の「-l 99」は無関係。オリジナルのリードのポジションで残す範囲の指定と考えればよい

# fastx-trimmer -f -l

## fastx-trimmer -f -l

「-f 3」を追加したら、「-l 99」はどうなるのかを調査。つまり、「-f 3」は3塩基目からという意味なので、最初の2塩基分がトリムされる。この場合に「-l 97」としたいといけないのかどうかを知りたい。

```
iu@bielinux[20160804] pwd [ 7:12 午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1*
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz [ 7:12 午後]
iu@bielinux[20160804] gunzip -c data1.fq.gz | head -n 4 [ 7:12 午後]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGGCTGTTCCCCTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCGGAGTAAGATCGATG
GTATTGGAGTTATCTGAATTCACTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJHIIJJJJJJHHAEEHHFFFFDDE
D>CDDDDDDDDDBCCEDCDDCCDDCCDEEDCCCCFFFFF
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99
- | gzip > trim1.fq.gz
iu@bielinux[20160804] gunzip -c trim1.fq.gz | head -n 4 [ 7:12 午後]
① ②
@SRR616268.20 2291:6:1101:1720:2221 length=107
TCTGGGCTGTTCCCCTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCGGAGTAAGATCGATGGT
ATTGGAGTTATCTGAATTCACTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CFFFFFHHHHHJJJJJJJJJJGHHIIIIJJJJJJJJJJJJHIIJJJJJJHHAEEHHFFFFDDED>
CDDDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 7:12 午後]
```

# fastx-trimmer –Iのみ

- トリミング (fastx-trimmer)

第5回W16-2で利用したfastx-trimmerを利用して、(とりあえず第5回W16-2と同じく100塩基目以降をトリムしたいので) `99` 塩基目まで残すという指定で実行し、`trim1.fq.gz`というファイル名で保存。

```
cd ~/Documents/si  
pwd  
ls -l data1*  
gunzip -c data1.  
gunzip -c data1.  
gunzip -c trim1.  
  
fastx_trimmer -h
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 様々なトリム条件1

## 様々なトリム条件1

2種類のトリム条件で作成したtrim1.fq.gzを入力としてRockhopper2を実行し、配列数を得る基本形。この段階で出力ファイル名の形式も意識する。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data1*
```

#-f 1 : 残す最初(first)の塩基が1塩基目という意味。  
#-l 107 : 残す最後(last)の塩基が107塩基目という意味。  
#つまり、トリムしないという意味。

```
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 107 - | gzip > trim1.fq.gz  
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_107.txt  
grep "number of assembled transcripts" result_01_f_107.txt
```

#-f 1 : 残す最初(first)の塩基が1塩基目という意味。  
#-l 104 : 残す最後(last)の塩基が104目という意味。  
#例えば元が104塩基より長いリードは3'末端側を104 bp分までトリムするが、  
#FaQCs実行で元々104 bp以下のリードはトリムしない。

```
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 104 - | gzip > trim1.fq.gz  
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_104.txt  
grep "number of assembled transcripts" result_01_f_104.txt
```

2種類のトリム条件で作成したtrim1.fq.gzを入力としてRockhopper2を実行し、配列数を得る基本形。①forward側の3'末端をトリムしない場合と、②104 bp以上のリードを104 bpになるまで3'末端をトリムする場合。赤枠を丸々コピペすると③のquoteが原因で変なことになる…のを経験してもよい。「quote>」となってもCTRL + Cで脱出可

①

②

③



# 様々なトリム条件1

コピペ実行結果。②104 bp以上のリードを104 bpになるまで3'末端をトリムすることで、③配列数が11個に増えたことがわかる

## 様々なトリム条件1

2種類のトリム条件で作成した `trim1.fq.gz` を入力として Rockhopper2 を実行し、配列数を得る基本形。この段階で出力ファイル名の形式も意識する。

```
iu@bielinux[~/Documents/srp017156/20160804]
cd ~/Documents/srp017156/20160804
pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1*
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 107 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_107.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_01_f_107.txt
Total number of assembled transcripts: 1
iu@bielinux[20160804]
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 104 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_104.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_01_f_104.txt
Total number of assembled transcripts: 11
iu@bielinux[20160804]
```

トリム後に残す範囲を計8通り分作成して、コピペ実行。約8分。例えば①は、オリジナルのリードポジションで、3-104番目の塩基以外をトリムするオプション

## 様々なトリム条件2

### 様々なトリム条件2

多くのトリム条件を一気に計算する。**3'末端のみ**3塩基刻みで95 bpまでと、**5'末端を3塩基目からスタート**(2塩基トリム)して**3'末端を3塩基刻みで**107-95 bpまでの長さを残す、計8通り分を作成し、一気にコピペ実行

```
### -f 1 and -l 101 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 101 - | gzip > trim1
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_101.txt
grep "number of assembled transcripts" result_01_f_101.txt
### -f 1 and -l 98 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 98 - | gzip > trim1.
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_098.txt
grep "number of assembled transcripts" result_01_f_098.txt
### -f 1 and -l 95 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 95 - | gzip > trim1.
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_095.txt
grep "number of assembled transcripts" result_01_f_095.txt
### -f 3 and -l 107 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 107 - | gzip > trim1
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_107.txt
grep "number of assembled transcripts" result_03_f_107.txt
### -f 3 and -l 104 ###

gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 104 - | gzip > trim1
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_104.txt
grep "number of assembled transcripts" result_03_f_104.txt
### -f 3 and -l 101 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 101 - | gzip > trim1
```

# 様々なトリム条件2

計算終了後の状態。計算中は①配列数に相当する数値を(それほど気合いは入れずに)眺めて結果の全体像や②条件あたりの所要時間を大まかに把握しておく。また、エラーメッセージが出ていないかも見る

## 様々なトリム条件2

多くのトリム条件を一気に計算する。  
3'末端のみ 3'塩基刻みで95 bpまでと、5'末端を3'塩基からスタート (2'塩基トリムなし) 3'末端を3'塩基刻みで107 bpまでの長さを残す 計  
8通り分を作成し、一氣

```
iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_101.txt
Total number of assembled transcripts: 372 ① ② [10:02午後]

iu@bielinux[20160804] ### -f 3 and -l 98 ####
zsh: command not found: ####

iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 98 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_098.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_098.txt
Total number of assembled transcripts: 705 ① ② [10:02午後]

iu@bielinux[20160804] ### -f 3 and -l 95 ####
zsh: command not found: ####

iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 95 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_095.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_095.txt
Total number of assembled transcripts: 704 ① ② [10:03午後]
```

# 結果の全体像を把握

File Edit View Search Terminal Help

```

iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls
data1.fq.gz          result_01_f_107.txt    result_f.txt
data2.fq.gz          result_03_f_095.txt    result_r.txt
result_01_f_095.txt  result_03_f_098.txt    result.txt
result_01_f_098.txt  result_03_f_101.txt    Rockhopper_Results
result_01_f_101.txt  result_03_f_104.txt    trim1.fq.gz
result_01_f_104.txt  result_03_f_107.txt

iu@bielinux[20160804] grep "number of assembled transcripts" result_0*
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_101.txt:Total number of assembled transcripts: 344
result_01_f_104.txt:Total number of assembled transcripts: 11
result_01_f_107.txt:Total number of assembled transcripts: 1
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_101.txt:Total number of assembled transcripts: 372
result_03_f_104.txt:Total number of assembled transcripts: 12
result_03_f_107.txt:Total number of assembled transcripts: 1
iu@bielinux[20160804]

```

[11:33午前]

①lsでファイル名をざっくり眺めて、②様々な条件でトリムした結果ファイル群のみをうまく表現できる「ワイルドカード(③ここでは\*)」を考える。もちろんファイル名を眺めるだけでどのような条件でトリムしたかが一目でわかるように意味を持たせるのも重要であり、作る段階で考えておくのが普通

結果の解釈。①がトリム条件で、②がアセンブリで得られた配列数(コンティグ数; 転写物数)

# 結果の全体像を把握

```

File Edit View Search Terminal Help [11:33午前]
iu@bielinux[20160804] pwd [11:33午前]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls [11:33午前]
data1.fq.gz      result_01_f_107.txt  result_f.txt
data2.fq.gz      result_03_f_095.txt  result_r.txt
result_01_f_095.txt  result_03_f_098.txt  result.txt
result_01_f_098.txt  result_03_f_101.txt  Rockhopper_Results
result_01_f_101.txt  result_03_f_104.txt  trim1.fq.gz
result_01_f_104.txt  result_03_f_107.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_0* [11:33午前]
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_101.txt:Total number of assembled transcripts: 344
result_01_f_104.txt:Total number of assembled transcripts: 11
result_01_f_107.txt:Total number of assembled transcripts: 1
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_101.txt:Total number of assembled transcripts: 372
result_03_f_104.txt:Total number of assembled transcripts: 12
result_03_f_107.txt:Total number of assembled transcripts: 1
iu@bielinux[20160804] [11:33午前]

```

# 結果の全体像を把握

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls
data1.fq.gz          result_01_f_107.txt
data2.fq.gz          result_03_f_095.txt
result_01_f_095.txt  result_03_f_098.txt
result_01_f_098.txt  result_03_f_101.txt
result_01_f_101.txt  result_03_f_104.txt
result_01_f_104.txt  result_03_f_107.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_0*
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_101.txt:Total number of assembled transcripts: 344
result_01_f_104.txt:Total number of assembled transcripts: 11
result_01_f_107.txt:Total number of assembled transcripts: 1
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_101.txt:Total number of assembled transcripts: 372
result_03_f_104.txt:Total number of assembled transcripts: 12
result_03_f_107.txt:Total number of assembled transcripts: 1
iu@bielinux[20160804]

```

forward側  
 107 bp →  
  
 ② ③ ← 93 bp  
 reverse側

[11:33午前]

今解析しているのは、①forward側リード。②と③で指定した範囲外のトリムを、計10条件(動作確認用の2条件と一緒にやった8条件)で行ったときに、どれだけアセンブリがうまくいったかを調べようとしている。数が多くれば多いほどいいというものでもないが、全くアセンブルされないのはおかしいという考え方のほうがこの場合は優先

result\_r.txt  
 result.txt  
 Rockhopper\_Results  
 trim1.fq.gz

# 結果の全体像を把握

①と②が似た傾向であることから、リードの  
③5'側のトリミングがアセンブリ結果に与  
える寄与度は、④3'側に比べて低いと判断

```

File Edit View Search Terminal Help [11:33午前]
iu@bielinux[20160804] pwd /home/iu/Documents/srp017156/20160804 [11:33午前]
iu@bielinux[20160804] ls
data1.fq.gz      result_01_f_107.txt  result_f.txt
data2.fq.gz      result_03_f_095.txt  result_r.txt
result_01_f_095.txt  result_03_f_098.txt  result.txt
result_01_f_098.txt  result_03_f_101.txt  Rockhopper_Results
result_01_f_101.txt  result_03_f_104.txt  trim1.fq.gz
result_01_f_104.txt  result_03_f_107.txt

iu@bielinux[20160804] grep "number of assembled transcripts" result_0*
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_101.txt:Total number of assembled transcripts: 344
result_01_f_104.txt:Total number of assembled transcripts: 11
result_01_f_107.txt:Total number of assembled transcripts: 1
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_101.txt:Total number of assembled transcripts: 372
result_03_f_104.txt:Total number of assembled transcripts: 12
result_03_f_107.txt:Total number of assembled transcripts: 1

forward側
107 bp
→
③ ④
93 bp
reverse側

```

727
723
344
11
1
704
705
372
12
1

[11:33午前]

# 結果の全体像を把握

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls
data1.fq.gz          result_01_f_107.t
data2.fq.gz          result_03_f_095.txt  result_r.txt
result_01_f_095.txt  result_03_f_098.txt  result.txt
result_01_f_098.txt  result_03_f_101.txt  Rockhopper_Results
result_01_f_101.txt  result_03_f_104.txt  trim1.fq.gz
result_01_f_104.txt  result_03_f_107.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_0*
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_101.txt:Total number of assembled transcripts: 344
result_01_f_104.txt:Total number of assembled transcripts: 11
result_01_f_107.txt:Total number of assembled transcripts: 1
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_101.txt:Total number of assembled transcripts: 372
result_03_f_104.txt:Total number of assembled transcripts: 12
result_03_f_107.txt:Total number of assembled transcripts: 1
iu@bielinux[20160804]

```

[11:33午前]

5' 側の①1塩基目を残す(全くトリムしない)ほうが、②3塩基目から始める(最初の2塩基をトリムする)よりも配列数が増える、という判断を①と②あたりを眺めて下す。これは③95と98塩基まで残す2条件であるが、コンティグ数の大小関係が①(727 > 723)と②(704 < 705)で逆転しているので、一応99塩基目まで残す条件あたりもやったほうがいいかも…などと考える

①

②

③

①97と99塩基まで残す条件の追加、および②2塩基目から始める(最初の1塩基をトリムする)条件も行う。これはコピペせずにスライドを眺めるだけ。約6分

# 様々なトリム条件3

## 様々なトリム条件3

気になったトリム条件を実行。

```
### -f 1 and -l 99 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_099.txt
grep "number of assembled transcripts" result_01_f_099.txt

### -f 1 and -l 97 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 97 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_01_f_097.txt
grep "number of assembled transcripts" result_01_f_097.txt

### -f 3 and -l 99 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_099.txt
grep "number of assembled transcripts" result_03_f_099.txt

### -f 3 and -l 97 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 97 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_097.txt
grep "number of assembled transcripts" result_03_f_097.txt
```

①

```
### -f 2 and -l 99 ###
gunzip -c data1.fq.gz | fastx_trimmer -f 2 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_02_f_099.txt
grep "number of assembled transcripts" result_02_f_099.txt

### -f 2 and -l 98 ###

```

②

# 様々なトリム条件3

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -lt result 0*
-rw-rw-r-- 1 iu iu 577 6月 25 13:07 result_02_f_097.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:06 result_02_f_098.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:05 result_02_f_099.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:05 result_03_f_097.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:04 result_03_f_099.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:03 result_01_f_097.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:02 result_01_f_099.txt
-rw-rw-r-- 1 iu iu 577 6月 23 22:03 result_03_f_095.txt
④
③
-rw-rw-r-- 1 iu iu 577 6月 23 22:02 result_03_f_098.txt
-rw-rw-r-- 1 iu iu 576 6月 23 22:02 result_03_f_101.txt
-rw-rw-r-- 1 iu iu 571 6月 23 22:01 result_03_f_104.txt
-rw-rw-r-- 1 iu iu 568 6月 23 22:00 result_03_f_107.txt
-rw-rw-r-- 1 iu iu 577 6月 23 21:59 result_01_f_095.txt
-rw-rw-r-- 1 iu iu 577 6月 23 21:58 result_01_f_098.txt
-rw-rw-r-- 1 iu iu 576 6月 23 21:57 result_01_f_101.txt
-rw-rw-r-- 1 iu iu 571 6月 23 21:34 result_01_f_104.txt
-rw-rw-r-- 1 iu iu 568 6月 23 21:33 result_01_f_107.txt
iu@bielinux[20160804]

```

コピペ実行後に①lsした結果。-ltオプションをつけたおかげで作成時間順にソートされる。こうすることで、②さきほどコピペ実行したものと、そうでないものを分けることができる。ここでも③作成時間や④ファイルサイズを一応眺めてチェック。赤枠分が見えないだけだとわかつていれば①をやってもよい

[ 1:45午後 ]

# 結果の全体像を把握2

## 結果の全体像を把握2

「ワイルドカード(ここでは\*)」をうまく利用して、着目したい結果ファイルのみよりわかりやすく把握できるような小細工もする。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls result_0*
```

```
grep "number of assembled transcripts" result_0*9[0-9].txt
```



①



③



②

①で表示させるものを「result\_0\*」に限定してls。grep時に、さらに②の赤下線で90-99塩基まで残す結果に限定。赤下線一番左の③9は、90.txtから99.txtというファイル名のものに限定するという意味。[0-9]は、91.txtでも97.txtでも0-9の範囲内の数値1つであればなんでもよいという意味(第6回W11-10; W13-4)

# 結果の全体像を把握2

## 結果の全体像を把握2

「ワイルドカード(ここでは\*)」をうまく利用して、着目したい結果ファイルのみよりわかりやすく把握できるような小細工もする。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls result_0*
```

```
grep "number of assembled transcripts" result_0*9[0-9].txt
```

①で表示させるものを「result\_0\*」に限定してls。grep時に、さらに②の赤下線で90-99塩基まで残す結果に限定。赤下線一番左の③9は、90.txtから99.txtというファイル名のものに限定するという意味。[0-9]は、91.txtでも97.txtでも0-9の範囲内の数値1つであればなんでもよいという意味(第6回W11-10; W13-4)。実行結果と見比べるとわかりやすいでしょう



```
result_01_f_098.txt  result_02_f_099.txt  result_03_f_101.txt  
result_01_f_099.txt  result_02_f_099.txt  result_03_f_104.txt  
result_01_f_101.txt   result_03_f_095.txt  result_03_f_107.txt  
result_01_f_104.txt   result_03_f_097.txt  
iu@bielinux[20160804] grep "number of assembled transcripts" result_0*9[0-9].txt  
result_01_f_095.txt:Total number of assembled transcripts: 727  
result_01_f_097.txt:Total number of assembled transcripts: 728  
result_01_f_098.txt:Total number of assembled transcripts: 723  
result_01_f_099.txt:Total number of assembled transcripts: 731  
result_02_f_097.txt:Total number of assembled transcripts: 708  
result_02_f_098.txt:Total number of assembled transcripts: 711  
result_02_f_099.txt:Total number of assembled transcripts: 713  
result_03_f_095.txt:Total number of assembled transcripts: 704  
result_03_f_097.txt:Total number of assembled transcripts: 711  
result_03_f_098.txt:Total number of assembled transcripts: 705  
result_03_f_099.txt:Total number of assembled transcripts: 705
```

[ 2:52午後 ]

# 結果の全体像を把握2

① 2塩基目以降を残す(1塩基目をトリム)場合は、1塩基目と3塩基目以降の結果の中間であることを確認できた。また、②リードの3'末端側のみ99塩基目まで残した場合に、最も配列数が多くなる(731個)ことも分かった

```
iu@bielinux[20160804] pwd  
/home/iu/Documents/srp017156/20160804  
iu@bielinux[20160804] ls result_0*  
result_01_f_095.txt  result_01_f_107.txt  result_03_f_098.txt  
result_01_f_097.txt  result_02_f_097.txt  result_03_f_099.txt  
result_01_f_098.txt  result_02_f_098.txt  result_03_f_101.txt  
result_01_f_099.txt  result_02_f_099.txt  result_03_f_104.txt  
result_01_f_101.txt  result_03_f_095.txt  result_03_f_107.txt  
result_01_f_104.txt  result_03_f_097.txt  
iu@bielinux[20160804] grep "number of assembled transcripts" result_0*9[0-9].txt  
result_01_f_095.txt:Total number of assembled transcripts: 727  
result_01_f_097.txt:Total number of assembled transcripts: 728  
result_01_f_098.txt:Total number of assembled transcripts: 723  
result_01_f_099.txt:Total number of assembled transcripts: 731  
result_02_f_097.txt:Total number of assembled transcripts: 708  
result_02_f_098.txt:Total number of assembled transcripts: 711  
result_02_f_099.txt:Total number of assembled transcripts: 713  
result_03_f_095.txt:Total number of assembled transcripts: 704  
result_03_f_097.txt:Total number of assembled transcripts: 711  
result_03_f_098.txt:Total number of assembled transcripts: 705  
result_03_f_099.txt:Total number of assembled transcripts: 705  
iu@bielinux[20160804]
```

[ 2:49午後 ] [ 2:49午後 ] [ 2:52午後 ]

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 他の結果もチェック

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls result_0*
result_01_f_095.txt  result_01_f_107.txt  result_03_f_098.txt
result_01_f_097.txt  result_02_f_097.txt  result_03_f_099.txt
result_01_f_098.txt  result_02_f_098.txt  result_03_f_101.txt
result_01_f_099.txt  result_02_f_099.txt  result_03_f_104.txt
result_01_f_101.txt  result_03_f_095.txt  result_03_f_107.txt
result_01_f_104.txt  result_03_f_097.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_0*9[0-9].txt
result_01_f_095.txt:Total number of assembled transcripts: 727
result_01_f_097.txt:Total number of assembled transcripts: 728
result_01_f_098.txt:Total number of assembled transcripts: 723
result_01_f_099.txt:Total number of assembled transcripts: 731
result_02_f_097.txt:Total number of assembled transcripts: 708
result_02_f_098.txt:Total number of assembled transcripts: 711
result_02_f_099.txt:Total number of assembled transcripts: 713
result_03_f_095.txt:Total number of assembled transcripts: 704
result_03_f_097.txt:Total number of assembled transcripts: 711
result_03_f_098.txt:Total number of assembled transcripts: 705
result_03_f_099.txt:Total number of assembled transcripts: 705
iu@bielinux[20160804]

```

おさらい。①Total number of assembled transcriptsを含む行の結果では、②リードの3'末端側のみ99塩基目まで残した場合に、最も配列数が多かった(731個)。赤枠は5'末端のトリム位置の違いで区別しやすくしたいだけ

[ 2:49 午後 ]

②

[ 2:52 午後 ]

# 他の結果もチェック1

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls result_0*
result_01_f_095.txt  result_01_f_107.txt
result_01_f_097.txt  result_02_f_097.txt  result_03_f_099.txt
result_01_f_098.txt  result_02_f_098.txt  result_03_f_101.txt
result_01_f_099.txt  result_02_f_099.txt  result_03_f_104.txt
result_01_f_101.txt  result_03_f_095.txt  result_03_f_107.txt
result_01_f_104.txt  result_03_f_097.txt
iu@bielinux[20160804] grep "Perfectly aligned reads" result_0*9[0-9].txt
result_01_f_095.txt:  Perfectly aligned reads: 623660 64%
result_01_f_097.txt:  Perfectly aligned reads: 624139 64%
result_01_f_098.txt:  Perfectly aligned reads: 623898 64%
result_01_f_099.txt:  Perfectly aligned reads: 624169 64%
result_02_f_097.txt:  Perfectly aligned reads: 679686 70%
result_02_f_098.txt:  Perfectly aligned reads: 679713 70%
result_02_f_099.txt:  Perfectly aligned reads: 679436 70%
result_03_f_095.txt:  Perfectly aligned reads: 688831 71%
result_03_f_097.txt:  Perfectly aligned reads: 689383 71%
result_03_f_098.txt:  Perfectly aligned reads: 689419 71%
result_03_f_099.txt:  Perfectly aligned reads: 689402 71%
iu@bielinux[20160804]

```

①Perfectly aligned readsを含む行の結果は、マップされたリード数とその割合に関するもの。リードの5'末端側を②1塩基目から残す場合(64%)は、③それ以外の2塩基目(70%)および3塩基目(71%)から残す場合に比べて極端に減ることがわかった。配列数も重要だが、この数は発現解析時に効いてくるので無視できない

result_01_f_095.txt:	Perfectly aligned reads:	623660	64%
result_01_f_097.txt:	Perfectly aligned reads:	624139	64%
result_01_f_098.txt:	Perfectly aligned reads:	623898	64%
result_01_f_099.txt:	Perfectly aligned reads:	624169	64%
result_02_f_097.txt:	Perfectly aligned reads:	679686	70%
result_02_f_098.txt:	Perfectly aligned reads:	679713	70%
result_02_f_099.txt:	Perfectly aligned reads:	679436	70%
result_03_f_095.txt:	Perfectly aligned reads:	688831	71%
result_03_f_097.txt:	Perfectly aligned reads:	689383	71%
result_03_f_098.txt:	Perfectly aligned reads:	689419	71%
result_03_f_099.txt:	Perfectly aligned reads:	689402	71%

[ 2:55 午後 ]

# 他の結果もチェック2

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls result_0*
result_01_f_095.txt  result_01_f_107.txt  result_03_f_098.txt
result_01_f_097.txt  result_02_f_097.txt  result_03_f_099.txt
result_01_f_098.txt  result_02_f_098.txt  result_03_f_101.txt
result_01_f_099.txt  result_02_f_099.txt  result_03_f_104.txt
result_01_f_101.txt  result_03_f_095.txt  result_03_f_107.txt
result_01_f_104.txt  result_03_f_097.txt
iu@bielinux[20160804] grep "number of assembled bases" result_0*9[0-9].txt
result_01_f_095.txt:  Total number of assembled bases: 170283
result_01_f_097.txt:  Total number of assembled bases: 175219
result_01_f_098.txt:  Total number of assembled bases: 177046
result_01_f_099.txt:  Total number of assembled bases: 180116
result_02_f_097.txt:  Total number of assembled bases: 178878
result_02_f_098.txt:  Total number of assembled bases: 181484
result_02_f_099.txt:  Total number of assembled bases: 183106
result_03_f_095.txt:  Total number of assembled bases: 171974
result_03_f_097.txt:  Total number of assembled bases: 176714
result_03_f_098.txt:  Total number of assembled bases: 179437
result_03_f_099.txt:  Total number of assembled bases: 181491
iu@bielinux[20160804]

```

①Total number of assembled basesを含む行の結果は、総塩基数に関するもの（ゲノムの場合はゲノムサイズに相当）。②最も総塩基数が多かったのは、2-99塩基目を残した場合。総合的に見て、②が一番いいだろうと思いつつ、一応これまでの結果をファイルに保存してホストOS上で整形してまとめる

[ 3:22 午後 ]

# ファイル保存して整形

詳細は省くが、これまで行った3種類の文字列を含む行情報をhoge\_\*.txtという名前で共有フォルダに保存

- ホストOS上で整形すべく、共有フォルダ([~/Desktop/mac\\_share](#))上に結果をファイル保存

Linux上でざっと傾向を見ておいて、あとはエクセルで手作業で整形してまとめたりします。リアルは、[cut](#)コマンドなどを用いたりしてもう少し手作業の手間を省いたりします。コマンドを複数行にわたくって記述する場合は「バックスラッシュ」です([第4回W5-2](#)や[W9-8](#))。Macで「バックスラッシュ」を出したい場合は、「Alt + \」で出るらしい。

```
cd ~/Documents/srp017156/20160804

pwd
grep "number of assembled transcripts" result_0*9[0-9].txt \
    > ~/Desktop/mac_share/hoge_1.txt
grep "Perfectly aligned reads" result_0*9[0-9].txt \
    > ~/Desktop/mac_share/hoge_2.txt
grep "number of assembled bases" result_0*9[0-9].txt \
    > ~/Desktop/mac_share/hoge_3.txt
```

# ファイル保存して整形

ホストOS上で整形すべく、共有フォルダ(`~/Desktop/mac_share`)上に結果をLinux上でざっと傾向を見ておいて、あとはエクセルで手作業で整形します。`cut`コマンドなどを用いたりしてもう少し手作業の手間を省いたりしまして記述する場合は「バックスラッシュ」です(第4回W5-2やW9-8)。N/Aの場合は、「Alt + \」で出るらしい。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
grep "number of assembled transcripts" result_0*9[0-9].txt \
> ~/Desktop/mac_share/hoge_1.txt
grep "Perfectly aligned reads" result_0*9[0-9].txt \
> ~/Desktop/mac_share/hoge_2.txt
grep "number of assembled bases" result_0*9[0-9].txt \
> ~/Desktop/mac_share/hoge_3.txt
```

まとめた結果。**①2-99塩基目の部分を残すのがいいかな**と私は判断。実はこれって、第5回W18-7(2016.08.02のスライド85)で妄想した記憶が残っているのも多少はあるが、ゲノム配列を使わずとも同様な結論を論理的に導き出せる例として重要。実際の作業としては、①のメインデータのデータ解析進行速度を妨げない程度に、②や③などの他のいくつかの候補も含めて同時並行でその後の解析を行う

トリム条件	配列数	Perfectly aligned reads リード数	その割合	総塩基数
01_f_095	727	623660	64%	170283
01_f_097	728	624139	64%	175219
01_f_098	723	623898	64%	177046
② 01_f_099	731	624169	64%	180116
02_f_097	708	679686	70%	178878
02_f_098	711	679713	70%	181484
① 02_f_099	713	679436	70%	183106
03_f_095	704	688831	71%	171974
03_f_097	711	689383	71%	176714
03_f_098	705	689419	71%	179437
③ 03_f_099	705	689402	71%	181491

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# ベストな条件で..

- ベストな条件でpaired-endアセンブリを実行

reverse側は何もトリムをしていないdata2.fq.gzを与えてpaired-endでRockhopperを実行。アセンブリ結果ファイル(Rockhopper\_Results/transcripts.txt)はどんどん上書きされるので、別名でコピーしている。

```

pwd
### -f 1 and -l 99 ####
①
gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz%data2.fq.gz > result_01_f_099_r_.txt
grep "number of assembled transcripts" result_01_f_099_r_.txt
cp Rockhopper_Results/transcripts.txt Rockhopper_Results/transcripts_01_f_09

### -f 2 and -l 99 ####
②
gunzip -c data1.fq.gz | fastx_trimmer -f 2 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz%data2.fq.gz > result_02_f_099_r_.txt
grep "number of assembled transcripts" result_02_f_099_r_.txt
cp Rockhopper_Results/transcripts.txt Rockhopper_Results/transcripts_02_f_09

### -f 3 and -l 99 ####
③
gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip > trim1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz%data2.fq.gz > result_03_f_099_r_.txt
grep "number of assembled transcripts" result_03_f_099_r_.txt
cp Rockhopper_Results/transcripts.txt Rockhopper_Results/transcripts_03_f_09

grep "number of assembled transcripts" *_r_.txt
grep "Perfectly aligned reads" *_r_.txt
grep "number of assembled bases" *_r_.txt

```

forward側の99塩基目まで残すのは固定で、①1塩基目、②2塩基目、③3塩基目から残すという計3つの条件でpaired-endアセンブリを実行。ここで、reverse側は何も行わないdata2.fq.gzを入力として与えている。コピペ実行。約6分

コピペ実行後の状態。①赤枠部分  
が結果の全体像を眺めているところ

# ベストな条件で...

- ベストな条件で paired-end アセンブリを実行

reverse側は何もトリムをしていない data2.fq.gz を与えて paired-end で Rockhopper2 を実行。アセンブリ結果ファイル result\_03\_f\_099\_r\_.txt

```

File Edit View Search Terminal Help
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip >
trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz%data2.fq.gz > result_0
3_f_099_r_.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_099_r_.tx
t
Total number of assembled transcripts: 660
iu@bielinux[20160804] cp Rockhopper_Results/transcripts.txt Rockhopper_Results/tran
scripts_03_f_099_r_.txt
iu@bielinux[20160804] grep "number of assembled transcripts" *_r_.txt [10:21午後]
result_01_f_099_r_.txt:Total number of assembled transcripts: 739
result_02_f_099_r_.txt:Total number of assembled transcripts: 662
result_03_f_099_r_.txt:Total number of assembled transcripts: 660
iu@bielinux[20160804] grep "Perfectly aligned reads" *_r_.txt [10:22午後]
result_01_f_099_r_.txt: Perfectly aligned reads: 600797 62%
result_02_f_099_r_.txt: Perfectly aligned reads: 653510 67%
result_03_f_099_r_.txt: Perfectly aligned reads: 663347 68%
iu@bielinux[20160804] grep "number of assembled bases" *_r_.txt [10:22午後]
result_01_f_099_r_.txt: Total number of assembled bases: 451183
result_02_f_099_r_.txt: Total number of assembled bases: 459768
result_03_f_099_r_.txt: Total number of assembled bases: 459662
iu@bielinux[20160804] [10:22午後]

```

①

# ベストな条件で...

- ベストな条件で paired-end アセンブリを実行

reverse側は何もトリムをしていない data2.fq.gz を与えて paired-end で  
結果ファイル result\_03\_f\_099\_r\_.txt が得られる

①マップされたリード数や②総塩基数は、paired-end でも確かに「02\_f\_099」がイメージ通りの相対的な関係だった。しかし、③配列数が「01\_f\_099」よりも 10%ほど少ない (739 個 vs. 662 個) のは、無視できない違いな気がする。悩ましいが一応まとめる

```

File Edit View Terminal Help
pwd
### gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip > trim1.fq.gz
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 3 -l 99 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz%data2.fq.gz > result_03_f_099_r_.txt
iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_099_r_.txt
Total number of assembled transcripts: 660
iu@bielinux[20160804] cp Rockhopper_Results/transcripts.txt Rockhopper_Results/transcripts_03_f_099_r_.txt
iu@bielinux[20160804] grep "number of assembled transcripts" *_r_.txt [10:21午後]
result_01_f_099_r_.txt: Total number of assembled transcripts: 739
result_02_f_099_r_.txt: Total number of assembled transcripts: 662 } ③
result_03_f_099_r_.txt: Total number of assembled transcripts: 660
iu@bielinux[20160804] grep "Perfectly aligned reads" *_r_.txt [10:22午後]
result_01_f_099_r_.txt: Perfectly aligned reads: 600797 62%
result_02_f_099_r_.txt: Perfectly aligned reads: 653510 67% } ①
result_03_f_099_r_.txt: Perfectly aligned reads: 663347 68%
iu@bielinux[20160804] grep "number of assembled bases" *_r_.txt [10:22午後]
result_01_f_099_r_.txt: Total number of assembled bases: 451183
result_02_f_099_r_.txt: Total number of assembled bases: 459768 } ②
result_03_f_099_r_.txt: Total number of assembled bases: 459662
iu@bielinux[20160804] [10:22午後]

```

# これまでのまとめ

トリム条件	配列数	Perfectly aligned reads		総塩基数
		リード数	その割合	
01_f_095	727	623660	64%	170283
01_f_097	728	624139	64%	175219
01_f_098	723	623898	64%	177046
01_f_099	731	624169	64%	180116
02_f_097	708	679686	70%	178878
02_f_098	711	679713	70%	181484
02_f_099	713	679436	70%	183106
03_f_095	704	688831	71%	171974
03_f_097	711	689383	71%	176714
03_f_098	705	689419	71%	179437
03_f_099	705	689402	71%	181491

トリム条件	配列数	Perfectly aligned reads		総塩基数
		リード数	その割合	
01_f_099_r_	739	600797	62%	451183
02_f_099_r_	662	653510	67%	459768
03_f_099_r_	660	663347	68%	459662

# これまでのまとめ

①reverse側を加えてpaired-endにすることで、①総塩基数が②forward側のみのsingle-endに比べて劇的に増加することがわかる。配列数は、③paired-endは④single-endに比べて若干全体的に減っていることから、総塩基数との関係を踏まえ、配列あたりの平均長が3倍程度伸びていると解釈できる。paired-endにする利点がよくわかる結果といえる

トリム条件	配列数	Perfectly aligned reads			総塩基数	トリム条件	配列数	Perfectly aligned reads			総塩基数
		リード数	その割合					リード数	その割合		
01_f_095	727	623660	64%		170283		01_f_099_r_	739	600797	62%	451183
01_f_097	728	624139	64%		175219		02_f_099_r_	662	653510	67%	459768
01_f_098	723	623898	64%		177046		03_f_099_r_	660	663347	68%	459662
01_f_099	731	624169	64%		180116						
02_f_097	708	679686	70%		178878						
02_f_098	711	679713	70%		181484						
02_f_099	713	679436	70%		183106						
03_f_095	704	688831	71%		171974						
03_f_097	711	689383	71%		176714						
03_f_098	705	689419	71%		179437						
03_f_099	705	689402	71%		181491						



# これまでのまとめ

どれがいいかはこの段階でも悩ましいですが、やはりマップされたリード数や総塩基数の観点から、①ですかねえ…

トリム条件	配列数	Perfectly aligned reads		総塩基数	トリム条件	配列数	Perfectly aligned reads		総塩基数
		リード数	その割合				リード数	その割合	
01_f_095	727	623660	64%	170283					
01_f_097	728	624139	64%	175219					
01_f_098	723	623898	64%	177046					
01_f_099	731	624169	64%	180116	01_f_099_r_	739	600797	62%	451183
02_f_097	708	679686	70%	178878					
02_f_098	711	679713	70%	181484					
02_f_099	713	679436	70%	18310①	02_f_099_r_	662	653510	67%	459768
03_f_095	704	688831	71%	171974					
03_f_097	711	689383	71%	176714					
03_f_098	705	689419	71%	179437					
03_f_099	705	689402	71%	181491	03_f_099_r_	660	663347	68%	459662

# アセンブリ評価系...

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриптーム、正規化、発現変動、統計、モラ  
(last modified 2016/06/03, since 2011)

- ・前処理 | フィルタリング | paired-end | 共通リード抽出 | [ShortRead\(Morgan 2009\)](#) (last modified 2015/06/26)
- ・アセンブル | について (last modified 2014/06/20)
- ・アセンブル | ゲノム用 (last modified 2016/03/2)
- ・アセンブル | トранскриптーム(転写物)用 (last modified 2016/06/03) ①
- ・マッピング | について (last modified 2016/04/0)
- ・マッピング | [basic aligner](#) (last modified 2014/08/08)
- ・マッピング | [splice-aware aligner](#) (last modified 2016/04/07)
- ・マッピング | [Bisulfite sequencing](#)用 (last modified 2014/07/09)
- ・マッピング | [\(ESTレベルの長さの\)contig](#) (last modified 2014/06/24)
- ・マッピング | 基礎 (last modified 2013/06/19)
- ・マッピング | single-end | ゲノム | basic aligner(基礎) | [QuasR\(Gaidatz\)](#)

スライドを見るだけ。様々なプログラムが他にもあります。②のIFRATは、得られた配列の確からしさを評価するプログラム群を提供していたような…

## アセンブル | トランскриптーム(転写物)用 NEW

Rパッケージはおそらくありません。

プログラム:

- ・ [Multiple-k](#): Surget-Groba and Montoya-Burgos, Genome Res., 2010
- ・ [Trans-ABYSS](#): Robertson et al., Nat Methods, 2010
- ・ [Rnnotator](#): Martin et al., BMC Genomics, 2010
- ・ [Trinity](#): Grabherr et al., Nat Biotechnol, 2011
- ・ [Oases](#): Schulz et al., Bioinformatics, 2012
- ・ [EBARDenovo](#): Chu et al., Bioinformatics, 2013
- ・ [BRANCH](#): Bao et al., Bioinformatics, 2013
- ・ [IDBA-tran](#): Peng et al., Bioinformatics, 2013
- ・ [SOAPdenovo-Trans](#): Xie et al., Bioinformatics, 2014
- ・ [VTBuilder](#): Archer et al., BMC Bioinformatics, 2014
- ・ [Rockhopper 2\(バクテリア用\)](#): Tjaden B, Genome Biol., 2015
- ・ [DETONATE\(RSEM-EVAL\)](#): Li et al., Genome Biol., 2014
- ・ [Bridger](#): Chang et al., Genome Biol., 2015
- ・ [IFRAT](#): Mbandi et al., BMC Bioinformatics, 2015 ②
- ・ [SCERNA\(主に植物\)](#): Honaas et al., PLoS One, 2010
- ・ [BinPacker](#): Liu et al., PLoS Comput Biol., 2016

Review、ガイドライン、バイオライン系:

- ・ Review: Martin and Wang, Nat Rev Genet., 2011
- ・ ガイドライン: Haznedaroglu et al., BMC Bioinformatics, 2012
- ・ Review: Góngora-Castillo, Nat Prod Rep., 2013

# この後の展開は...

## (Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、ranscripトーム、正規化、発現変動、統計  
(last modified 2016/06/03, since 2011)

- ・前処理 | フィルタリング | paired-end | 共通リード抽出 | [SL](#)
- ・アセンブル | [について](#) (last modified 2014/06/20)
- ・アセンブル | [ゲノム用](#) (last modified 2016/03/24)
- ・アセンブル | [ランスクリプトーム\(転写物\)用](#) (last modified 2016/06/03)
- ・マッピング | [について](#) (last modified 2016/04/07)
- ・マッピング | [basic aligner](#) (last modified 2014/08/08)
- ・マッピング | [splice-aware aligner](#) (last modified 2016/04/07)
- ・マッピング | [Bisulfite sequencing用](#) (last modified 2014/07/09)
- ・マッピング | [\(ESTレベルの長さの\)contig](#) (last modified 2014/06/24)
- ・マッピング | [基礎](#) (last modified 2013/06/19)
- ・マッピング | [single-end](#) | [ゲノム](#) | [basic aligner\(基礎\)](#) | [QuasR\(Gaidatzis et al., 2011\)](#)

①乳酸菌データについては、バクテリア専用のRockhopper2がいいに決まっているだろうが、念のため  
②有名なTrinityをインストールしてやってみたら、バクテリア用でもないのに配列数が相当増加したという衝撃  
の結果をお話。次に第5回でも紹介した③Bridgerを使おうと思ったがサンプルデータ実行段階でこけたという話

## アセンブル | ランスクリプトーム(転写物)用 NEW

Rパッケージはおそらくありません。

プログラム:

- ・ [Multiple-k: Surget-Groba and Montoya-Burgos, Genome Res., 2010](#)
- ・ [Trans-ABYSS: Robertson et al., Nat Methods, 2010](#)
- ・ [Rnnotator: Martin et al., BMC Genomics, 2010](#)
- ② [Trinity: Grabherr et al., Nat Biotechnol, 2011](#)
- ・ [Oases: Schulz et al., Bioinformatics, 2012](#)
- ・ [EBARDenovo: Chu et al., Bioinformatics, 2013](#)
- ・ [BRANCH: Bao et al., Bioinformatics, 2013](#)
- ・ [IDBA-tran: Peng et al., Bioinformatics, 2013](#)
- ・ [SOAPdenovo-Trans: Xie et al., Bioinformatics, 2014](#)
- ・ [VTBuilder: Archer et al., BMC Bioinformatics, 2014](#)
- ・ [Rockhopper 2\(バクテリア用\): Tjaden B, Genome Biol., 2015](#)
- ③ [DETONATE\(RSEM-EVAL\): Li et al., Genome Biol., 2014](#)
- ・ [Bridger: Chang et al., Genome Biol., 2015](#)
- ① [IFRAT: Mbandi et al., BMC Bioinformatics, 2015](#)
- ・ [SCERNA\(主に植物\): Honaas et al., PLoS One, 2016](#)
- ・ [BinPacker: Liu et al., PLoS Comput Biol., 2016](#)

Review、ガイドライン、バイオライン系:

- ・ Review: [Martin and Wang, Nat Rev Genet., 2011](#)
- ・ ガイドライン: [Haznedaroglu et al., BMC Bioinformatics, 2012](#)
- ・ Review: [Góngora-Castillo, Nat Prod Rep., 2013](#)

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



①ここからダウンロードできるよう  
なのでクリック。スライドを見るだけ

# Trinity

The screenshot shows the GitHub repository for Trinity. On the left, there's a sidebar with links like 'Home', 'Issues 46', and 'Pull requests 3'. Below it is a large logo for Trinity, featuring the word 'Trinity' in green with three circular icons: 'Inchworm' (green), 'Chrysalis' (blue), and 'Butterfly' (purple). The main content area displays the 'Quick Guide for the Impatient' page. This page contains instructions for Trinity assembly, command-line examples, and documentation links. A red circle with the number '1' points to the 'Download Trinity here.' link. The right sidebar lists various documentation sections such as 'Transcripts', 'Downstream Analyses', and 'Trinity Tidbits'.

Quick Guide for the Impatient

Trinity assembles transcript sequences from Illumina RNA-Seq data.

Download Trinity [here.](#) (1)

Build Trinity by typing 'make' in the base installation directory.

Assemble RNA-Seq data like so:

```
Trinity --seqType fq --left reads_1.fq --right reads_2.fq --CPU 6 --max_memory 20G
```

Find assembled transcripts as: 'trinity\_out\_dir/Trinity.fasta'

Use the documentation links in the right-sidebar to navigate this documentation, and contact our Google group for technical support.

## Intro to Trinity

Trinity, developed at the [Broad Institute](#) and the [Hebrew University of Jerusalem](#), represents a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data. Trinity combines three independent software modules: Inchworm, Chrysalis, and Butterfly, applied sequentially to process large volumes of RNA-seq reads. Trinity partitions the sequence data into many individual de Bruijn graphs, each representing the transcriptional complexity at a given gene or locus, and then processes each graph independently to extract full-length splicing

Clone this wiki locally  
<https://github.com/trinityrnaseq/trinityrnaseq>

Clone in Desktop

# Trinity

<https://github.com/trinitymaseq/trinitymaseq/releases>

①ページ下部に移動するとTrinityの昔のバージョンもダウンロード可能なようだが、②最新版のダウンロードが基本なので、とりあえず③tar.gzのやつをダウンロードすべくwindowsの場合は右クリックで「ショートカットのコピー」…した後からが次のスライド。約170MBもあるので~/Downloadsにダウンロード済み

The screenshot shows the GitHub repository page for 'trinityrnaseq / trinityrnaseq'. At the top, there are navigation links for Personal, Open source, Business, Explore, Pricing, Blog, Support, This repository, Search, Sign in, and Sign up. Below the header, there are buttons for Watch (48), Star (96), Fork (61), and Graphs. The 'Releases' tab is selected. A green button labeled 'Latest release' is visible. The main content area features a large title 'Trinity Release v2.2.0' with a red arrow pointing to it from above. Below the title, it says 'brianjohnhaas released this on Mar 17 · 3 commits to master since this release'. A list of changes includes: '-Butterfly update: bugfix related to polynucleotide runs.', '-util/SAM\_nameSorted\_to\_uniq\_count\_stats.pl: count fragments instead of reads.', '-util/abundance\_estimates\_to\_matrix.pl: will output a matrix even if only a single sample is', '-util/align\_and\_estimate\_abundance.pl: added support for salmon', and '-sample\_data/test\_align\_and\_estimate\_abundance/: added examples and tests for single-end and'. At the bottom, there is a section titled 'Downloads' with two options: 'Source code (zip)' and 'Source code (tar.gz)', with a red arrow pointing to the 'tar.gz' link.

wgetのところはコメントアウト(#)しています。次のスライドは①のコピペとほぼ同じ

# (ダウンロードと)解凍

- Trinity: Grabherr et al., Nat Biotechnol, 2011

- ダウンロードとインストール

講習会ではTrinity (ver. 2.2.0)のtar.gzファイルを~/Downloadsにダウンロード済み。

```
cd ~/Downloads
```

```
pwd
# wget -c https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.2.0.tar.gz
ls -l v2*
tar zxvf v2.2.0.tar.gz
cd trinityrnaseq-2.2.0
ls
more README

make
ls -lt
```

①

- パスを通す(Trinity)

~/binへのパスは第6回W12-3 (2016.08.03のスライド56)で通したので、ここにファイルを置くだけでよい。

```
cd ~/Downloads/trinityrnaseq-2.2.0
```

```
pwd
ls
where Trinity
cp Trinity ~/bin
where Trinity
```

# 解凍

①場所は~/Downloads。②Trinity ver. 2.2.0のtar.gzファイルが存在する状態からスタート。③解凍コマンドを実行。リターンキーを押す

The screenshot shows a terminal window with a dark theme. On the left is a vertical dock with icons for various applications: a terminal window, a file manager, a browser (Firefox), an email client, a file viewer, a spreadsheet, a presentation slide, and a file transfer tool. The terminal window has a black header bar with the menu: File, Edit, View, Search, Terminal, Help. The status bar at the top right shows system information: a network icon, Japanese input method (Ja), battery level, volume, the time 14:36, and a gear icon. The main area of the terminal displays the following command history:

```
iu@bielinux[Downloads] pwd [ 2:34午後]
/home/iu/Downloads
iu@bielinux[Downloads] ls -l v2*
-rw-rw-r-- 1 iu iu 174159736 6月 26 14:26 v2.2.0.tar.gz
iu@bielinux[Downloads] tar zxvf v2.2.0.tar.gz [ 2:34午後]
```

Three red arrows with numbers 1, 2, and 3 point to the first three lines of the terminal output, corresponding to the steps in the explanatory text above.

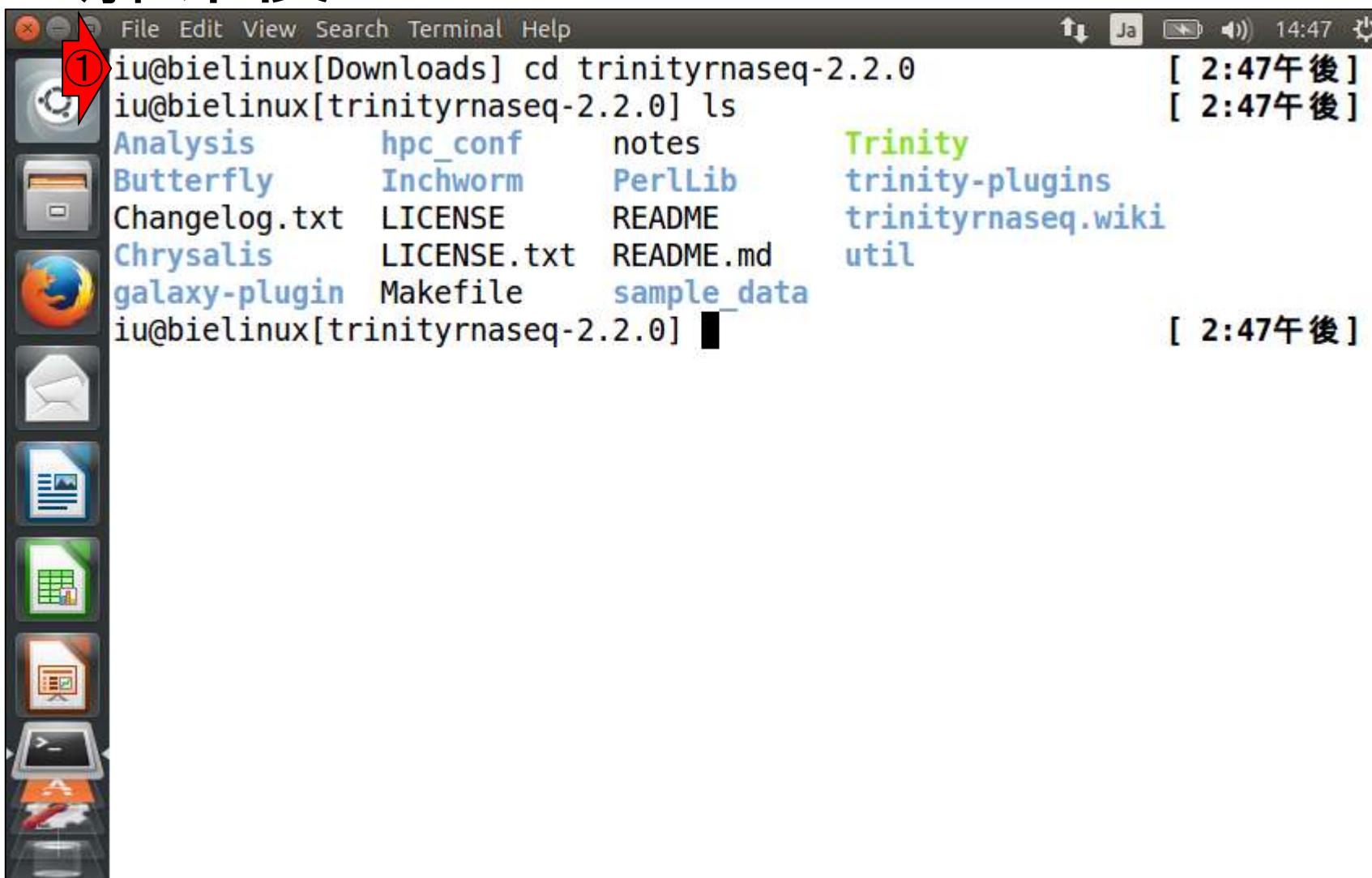
解凍終了後の状態。①赤下線部分の  
trinityrnaseq-2.2.0というディレクトリが作成さ  
れていると解釈する。エラーは出てないっぽい

# 解凍後

```
File Edit View Search Terminal Help
ome_assisted_assembly.pl
trinityrnaseq-2.2.0/util/support_scripts/print_butterfly_assemblies.pl
trinityrnaseq-2.2.0/util/support_scripts/process_GMAP_alignments_gff3_c
himeras_ok.pl
trinityrnaseq-2.2.0/util/support_scripts/revcomp_fasta.pl
trinityrnaseq-2.2.0/util/support_scripts/run_TMM_scale_matrix.pl
trinityrnaseq-2.2.0/util/support_scripts/run_UpperQuartileNormalization
_matrix.pl
trinityrnaseq-2.2.0/util/support_scripts/salmon_trans_to_gene_results.p
l
trinityrnaseq-2.2.0/util/support_scripts/scaffold_iworm_contigs.pl
trinityrnaseq-2.2.0/util/support_scripts/segment_GFF_partitions.pl
trinityrnaseq-2.2.0/util/support_scripts/tests/
trinityrnaseq-2.2.0/util/support_scripts/tests/sample_data_tests.py
trinityrnaseq-2.2.0/util/support_scripts/tests/test_prep.py
trinityrnaseq-2.2.0/util/support_scripts/tests/tests.py
trinityrnaseq-2.2.0/util/support_scripts/trinity_install_tests.sh
trinityrnaseq-2.2.0/util/support_scripts/wig_clip_to_bed.pl
trinityrnaseq-2.2.0/util/support_scripts/write_partitioned_trinity_cmds
.pl
iu@bielinux[Downloads] [ 2:36午後 ]
```

①trinityrnaseq-2.2.0ディレクトリに移動して、ls。タブ補完を有効利用

# 解凍後

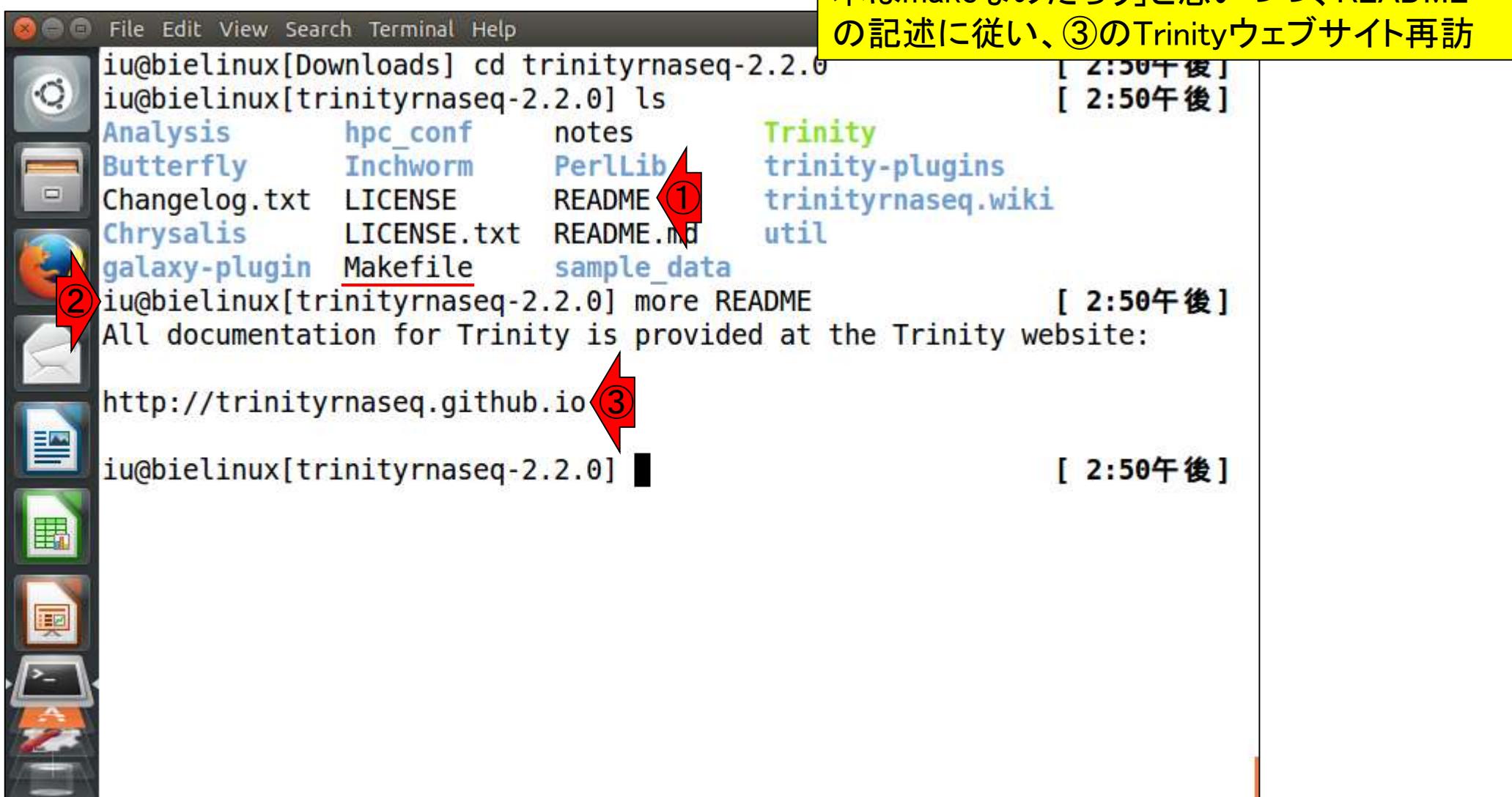


```
iu@bielinux[Downloads] cd trinityrnaseq-2.2.0 [ 2:47午後]
iu@bielinux[trinityrnaseq-2.2.0] ls [ 2:47午後]
Analysis      hpc_conf    notes          Trinity
Butterfly     Inchworm   PerlLib        trinity-plugins
Changelog.txt LICENSE    README         trinityrnaseq.wiki
Chrysalis     LICENSE.txt README.md    util
galaxy-plugin Makefile    sample_data
iu@bielinux[trinityrnaseq-2.2.0] [ 2:47午後]
```

# README

① READMEファイルがあったので、②moreで表示。赤下線のMakefileを見つけて、「基本はmakeなのだろう」と思いつつ、READMEの記述に従い、③のTrinityウェブサイト再訪

```
File Edit View Search Terminal Help
iu@bielinux[Downloads] cd trinityrnaseq-2.2.0 [ 2:50午後]
iu@bielinux[trinityrnaseq-2.2.0] ls [ 2:50午後]
Analysis      hpc_conf    notes          Trinity
Butterfly     Inchworm   PerlLib       trinity-plugins
Changelog.txt LICENSE    README        trinityrnaseq.wiki
Chrysalis     LICENSE.txt README.md    util
galaxy-plugin Makefile    sample_data
iu@bielinux[trinityrnaseq-2.2.0] more README [ 2:50午後]
All documentation for Trinity is provided at the Trinity website:
http://trinityrnaseq.github.io [ 2:50午後]
iu@bielinux[trinityrnaseq-2.2.0]
```



# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



よく見ると、①にInstalling Trinityと書いてあるので、とりあえずクリック

# Install

https://github.com/trinityrnaseq/trinityrnaseq/wiki GitHub, Inc. [US] Home · trinityrnaseq/tri... Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up Watch 48 Star 96 Fork 61

trinityrnaseq / trinityrnaseq

Code Issues 46 Pull requests 3 Wiki Pulse Graphs

## Home

Thisaru Guruge edited this page on Apr 4 · 30 revisions

# RNA-Seq De novo Assembly Using Trinity

Inchworm Chrysalis Butterfly

ATCG GATA TCG  
GATC ATCG TCG  
TGC ACG TCG

Pages 24

- Trinity Wiki Home
- **Installing Trinity** ①
  - Trinity Computing Requirements
  - Accessing Trinity on Publicly Available Compute Resources
- Running Trinity
  - Genome Guided Trinity Transcriptome Assembly
  - Genome Annotation
- Monitoring Progress of a Trinity Run
- Output of Trinity Assembly

①makeだけでよさそう。②はprerequisite (gcc ver. 4.3以上、Java-1.7以上)に関する記載。makeで失敗したら、prerequisiteの可能性も考えるというスタンスもありかも

# Install

The screenshot shows a GitHub repository page for 'trinityrnaseq / trinityrnaseq'. The 'Wiki' tab is selected. The main content is titled 'Installing Trinity'. It instructs users to type 'make' in the base installation directory to build Inchworm and Chrysalis, both written in C++. A note specifies that Java-1.7 or higher is required for Chrysalis. The page also mentions that Butterfly is written in Java and does not require special compilation. A sidebar on the right lists other pages in the Trinity Wiki.

After downloading the software to a Linux server, simply type

make

in the base installation directory. This should build Inchworm and Chrysalis, both written in C++. A version of gcc greater than 4.3 is required. Butterfly should not require any special compilation, as it's written in Java and already provided as portable precompiled software, but Java-1.7 (or higher) is required.

Afterwards, you may want to build the additional plugin components that provide support for downstream analyses in which case you would then type:

① makeだけでよさそう。②はprerequisite (gcc ver. 4.3以上、Java-1.7以上)に関する記載。makeで失敗したら、prerequisiteの可能性も考えるというスタンスもありかも

- Trinity Wiki Home
- Installing Trinity
  - Trinity Computing Requirements
  - Accessing Trinity on Publicly Available Compute Resources
- Running Trinity
  - Genome Guided Trinity Transcriptome Assembly
  - Genome Annotation
- Monitoring Progress of a Trinity Run
- Output of Trinity Assembly

# Install

```
File Edit View Search Terminal Help [ 15:07 ]  
iu@bielinux[Downloads] cd trinityrnaseq-2.2.0 [ 2:50午後 ]  
iu@bielinux[trinityrnaseq-2.2.0] ls [ 2:50午後 ]  
Analysis hpc_conf notes Trinity  
Butterfly Inchworm PerlLib trinity-plugins  
Changelog.txt LICENSE README trinityrnaseq.wiki  
Chrysalis LICENSE.txt README.md util  
galaxy-plugin Makefile sample_data  
iu@bielinux[trinityrnaseq-2.2.0] more README [ 2:50午後 ]  
All documentation for Trinity is provided at the Trinity website:  
http://trinityrnaseq.github.io  
①iu@bielinux[trinityrnaseq-2.2.0] make [ 2:50午後 ]
```

無事終了。Properlyというポジティブな副詞なのでインストール成功と解釈する

# Install



```

File Edit View Search Terminal Help
ity-plugins/samtools-0.1.19/misc'
make[2]: Leaving directory `/home/iu/Downloads/trinityrnaseq-2.2.0/trin
ity-plugins/samtools-0.1.19'
mv samtools-0.1.19/samtools ./BIN/.
make[1]: Leaving directory `/home/iu/Downloads/trinityrnaseq-2.2.0/trin
ity-plugins'
sh ./util/support_scripts/trinity_install_tests.sh
-----
Performing Unit Tests of Build
-----
JellyFish: has been Installed Properly
Inchworm: has been Installed Properly
Chrysalis: has been Installed Properly
QuantifyGraph: has been Installed Properly
GraphFromFasta: has been Installed Properly
ReadsToTranscripts: has been Installed Properly
fastool: has been Installed Properly
parafly: has been Installed Properly
samtools-0.1.19 has been Installed Properly
iu@bielinux[trinityrnaseq-2.2.0] [ 3:12 午後 ]

```

# Install後の確認

①ls -ltでソートして表示。②この3つが更新または新規作成されたようだ。他は何も変わっていないのが気になるが…

```

File Edit View Search Terminal Help
iu@bielinux[trinityrnaseq-2.2.0] ls -lt
total 236
drwxrwxr-x 13 iu iu 4096 6月 26 15:11 trinity-plugins
drwxrwxr-x 8 iu iu 4096 6月 26 15:10 Chrysalis
drwxrwxr-x 4 iu iu 4096 6月 26 15:09 Inchworm
drwxrwxr-x 5 iu iu 4096 3月 17 20:26 Analysis
drwxrwxr-x 3 iu iu 4096 3月 17 20:26 Butterfly
-rw-rw-r-- 1 iu iu 57355 3月 17 20:26 Changelog.txt
drwxrwxr-x 4 iu iu 4096 3月 17 20:26 galaxy-plugin
drwxrwxr-x 2 iu iu 4096 3月 17 20:26 hpc_conf
-rw-rw-r-- 1 iu iu 1490 3月 17 20:26 LICENSE
-rw-rw-r-- 1 iu iu 1492 3月 17 20:26 LICENSE.txt
-rw-rw-r-- 1 iu iu 1737 3月 17 20:26 Makefile
-rw-rw-r-- 1 iu iu 443 3月 17 20:26 notes
drwxrwxr-x 7 iu iu 4096 3月 17 20:26 PerlLib
-rw-rw-r-- 1 iu iu 99 3月 17 20:26 README
-rw-rw-r-- 1 iu iu 162 3月 17 20:26 README.md
drwxrwxr-x 13 iu iu 4096 3月 17 20:26 sample_data
-rwxrwxr-x 1 iu iu 109078 3月 17 20:26 Trinity
drwxrwxr-x 2 iu iu 4096 3月 17 20:26 trinityrnaseq.wiki
drwxrwxr-x 6 iu iu 4096 3月 17 20:26 util
iu@bielinux[trinityrnaseq-2.2.0]

```

[ 3:19 午後 ] [ 3:22 午後 ]

# Install

さきほどの①makeで行ったことは、②赤枠内の、③InchwormとChrysalisを作成するためのものだと解釈すれば納得できる

The screenshot shows a GitHub wiki page for the 'trinityrnaseq/trinityrnaseq' repository. The page title is 'Installing Trinity'. The main content discusses the installation process, mentioning the use of 'make' in the base installation directory to build 'Inchworm' and 'Chrysalis' (both written in C++) and that Java-1.7 (or higher) is required for 'Butterfly' (written in Java). A red box highlights this text, and a red arrow labeled '1' points to the word 'make'. Another red arrow labeled '2' points to the word 'Inchworm'. A red arrow labeled '3' points to the word 'special' in the sentence about 'Butterfly'.

Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

trinityrnaseq / trinityrnaseq Watch 48 Star 96 Fork 61

in the base installation directory. This should build Inchworm and Chrysalis, both written in C++. A version of gcc greater than 4.3 is required. Butterfly should not require any **special** compilation, as its written in Java and already provided as portable precompiled software, but Java-1.7 (or higher) is required.

## Installing Trinity

After downloading the software to a Linux server, simply type

make

in the base installation directory. This should build Inchworm and Chrysalis, both written in C++. A version of gcc greater than 4.3 is required. Butterfly should not require any special compilation, as its written in Java and already provided as portable precompiled software, but Java-1.7 (or higher) is required.

Afterwards, you may want to build the additional plugin components that provide support for downstream analyses in which case you would then type:

Pages 24

- Trinity Wiki Home
- Installing Trinity
  - Trinity Computing Requirements
  - Accessing Trinity on Publicly Available Compute Resources
  - Running Trinity
    - Genome Guided Trinity Transcriptome Assembly
    - Genome Annotation
- Monitoring Progress of a Trinity Run
- Output of Trinity Assembly

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 実行法を調べる

The screenshot shows a web browser displaying the GitHub repository for Trinity. The URL in the address bar is <https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing-Trinity>. The page title is "Installing Trinity". The GitHub header includes links for Personal, Open source, Business, Explore, Pricing, Blog, Support, This repository, Search, Sign in, and Sign up. Below the header, the repository name is "trinityrnaseq / trinityrnaseq" with metrics: Watch (48), Star (96), Fork (61). Navigation tabs include Code, Issues (46), Pull requests (3), Wiki (selected), Pulse, and Graphs. The main content area has a heading "Installing Trinity" and a note that Brian Haas edited it on Mar 22 · 7 revisions. To the right, there is a sidebar with a "Pages" section containing a list of wiki pages. A red arrow points to the "Running Trinity" link in this list.

After downloading the software to a Linux server, simply type

```
make
```

in the base installation directory. This should build Inchworm and Chrysalis, both written in C++. A version of gcc greater than 4.3 is required. Butterfly should not require any special compilation, as its written in Java and already provided as portable precompiled software, but Java-1.7 (or higher) is required.

Afterwards, you may want to build the additional plugin components that provide support for downstream analyses in which case you would then type:

- Trinity Wiki Home
- Installing Trinity
  - Trinity Computing Requirements
  - Accessing Trinity on Publicly Available Compute Resources
- Running Trinity
  - Genome Guided Trinity Transcriptome Assembly
  - Genome Annotation
- Monitoring Progress of a Trinity Run
- Output of Trinity Assembly

# 実行法を調べる

①下にスクロールしていくと、オプションやら基本的な利用法をざっと眺める

# 実行法を調べる

私がスクロールをやめて眺めるのは、①利用例のところ。②メモリとCPU数に気を付ければよい、③Trinityというコマンドが実行プログラムだと判断

Typical Trinity Command Line ①

A typical Trinity command for assembling non-strand-specific RNA-seq data would be like so, running the entire process on a single high-memory server (aim for ~1G RAM per ~1M ~76 base Illumina paired reads, but often *much* less memory is required):

Run Trinity like so:

```
③ Trinity --seqType fq --max_memory 50G \
    --left reads_1.fq.gz --right reads_2.fq.gz --CPU 6 ②
```

If you have multiple sets of fastq files, such as corresponding to multiple tissue types or conditions, etc., you can indicate them to Trinity like so:

```
Trinity --seqType fq --max_memory 50G \
    --left condA_1.fq.gz,condB_1.fq.gz,condC_1.fq.gz \
    --right condA_2.fq.gz,condB_2.fq.gz,condC_2.fq.gz \
    --CPU 6
```

Also note that fastq files can be gzip-compressed as shown above, in which case they should require a '.gz' extension.

## Options to Consider when Running Trinity

Trinity includes additional options to automate various aspects of RNA-Seq read processing that

# Install後の確認

```
File Edit View Search Terminal Help
iu@bielinux[trinityrnaseq-2.2.0] ls
total 236
drwxrwxr-x 13 iu iu 4096 6月 26 15:11 trinity-plugins
drwxrwxr-x 8 iu iu 4096 6月 26 15:11 Chrysalis
drwxrwxr-x 4 iu iu 4096 6月 26 15:11 Inchworm
drwxrwxr-x 5 iu iu 4096 3月 17 20:26 Analysis
drwxrwxr-x 3 iu iu 4096 3月 17 20:26 Butterfly
-rw-rw-r-- 1 iu iu 57355 3月 17 20:26 Changelog.txt
drwxrwxr-x 4 iu iu 4096 3月 17 20:26 galaxy-plugin
drwxrwxr-x 2 iu iu 4096 3月 17 20:26 hpc_conf
-rw-rw-r-- 1 iu iu 1490 3月 17 20:26 LICENSE
-rw-rw-r-- 1 iu iu 1492 3月 17 20:26 LICENSE.txt
-rw-rw-r-- 1 iu iu 1737 3月 17 20:26 Makefile
-rw-rw-r-- 1 iu iu 443 3月 17 20:26 notes
drwxrwxr-x 7 iu iu 4096 3月 17 20:26 PerlLib
-rw-rw-r-- 1 iu iu 99 3月 17 20:26 README
-rw-rw-r-- 1 iu iu 162 3月 17 20:26 README.md
drwxrwxr-x 13 iu iu 4096 3月 17 20:26 sample_data
-rwxrwxr-x 1 iu iu 109078 3月 17 20:26 Trinity
drwxrwxr-x 2 iu iu 4096 3月 17 20:26 trinityrnaseq.wiki
drwxrwxr-x 6 iu iu 4096 3月 17 20:26 util
iu@bielinux[trinityrnaseq-2.2.0] [ 3:22 午後 ]
```

①Trinityが確かにあった。②(緑色だからそれで判断してもよいが)実行権限(x; エックス)が自分にあることを一応確認。ウェブページ中の他の記述内容も合わせることでTrinityの実体が③Inchworm(シャクトリムシ), Chrysalis(サナギ), Butterfly(チョウ)なのだろうと想像する

# パスを通す

- [Trinity: Grabherr et al., Nat Biotechnol, 2011](#)

- ダウンロードとインストール (スライド 89)

講習会では Trinity (ver. 2.2.0) の tar.gz ファイルを ~/Downloads にダウンロード済み。

```
cd ~/Downloads

pwd
# wget -c https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.2.0.tar.gz
ls -l v2*
tar zxvf v2.2.0.tar.gz
cd trinityrnaseq-2.2.0
ls
more README

make
ls -lt
```

- パスを通す (スライド 106)

~/bin へのパスは 第6回W12-3 (2016.08.03のスライド 56) で通したので、ここにファイルを置くだけでよい。

```
cd ~/Downloads/trinityrnaseq-2.2.0
```

```
pwd
ls
where Trinity
cp Trinity ~/bin
where Trinity
```



# パスを通す

```

File Edit View Search Terminal Help
iu@bielinux[trinityrnaseq-2.2.0] pwd
/home/iu/Downloads/trinityrnaseq-2.2.0
iu@bielinux[trinityrnaseq-2.2.0] ls
Analysis      hpc_conf      notes
Butterfly     Inchworm     PerlLib
Changelog.txt LICENSE      README
Chrysalis     LICENSE.txt  README.md
galaxy-plugin Makefile      sample_data
iu@bielinux[trinityrnaseq-2.2.0] where Trinity
Trinity not found ②
iu@bielinux[trinityrnaseq-2.2.0] cp Trinity ~/bin
iu@bielinux[trinityrnaseq-2.2.0] where Trinity
/home/iu/bin/Trinity ③
/home/iu/bin/Trinity
iu@bielinux[trinityrnaseq-2.2.0]

```

The terminal window shows a user navigating through a 'trinityrnaseq-2.2.0' directory. They run 'where Trinity' and get an error message 'Trinity not found'. They then run 'cp Trinity ~/bin' to copy the executable to the user's bin directory. Finally, they run 'where Trinity' again and see it correctly finds the executable at '/home/iu/bin/Trinity'.

①～/binへのパスは第6回W12-3 (2016.08.03のスライド56)で通したので、ここにファイルを置くだけでよい。①のパスを通す作業の②前と③後で「where Trinity」実行結果の違いがわかる。  
①のやり方は間違います。コピーではなくシンボリックリンクを貼れば、スライド111までのエラーを回避できます(20160812修正)

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 色々試しながら実行1

## 色々試しながら実行1 (スライド 109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力してTrinityを実行。

```
cd ~/Documents/srp017156/20160804  
pwd  
ls -l data*  
### COMMON.pmというPerlモジュール部分でエラー ###  
Trinity --seqType fq --max_memory 2G --CPU 2 \  
--left data1.fq.gz --right data2.fq.gz } }  
  
### COMMON.pm問題は解決。が、Phase 1のJellyfishの所でフリーズ状態に... ###  
~/Downloads/trinityrnaseq-2.2.0/Trinity \  
--seqType fq --max_memory 2G --CPU 2 \  
--left data1.fq.gz --right data2.fq.gz  
  
### 「CTRL + C」で脱出し、計算途中だったTrinity関連のものを念のため削除 ###  
ls -lt | head  
rm -rf trinity_out_dir  
ls -l *readcount  
rm -f *readcount
```

赤枠部分をkopペ。①paired-endファイルがあるディレクトリに移動して、②Trinityの基本形を実行しようとしている。③赤字のコメント行部分でエラーが出るが気にしない

# 色々試しながら実行1

## 色々試しながら実行1 (スライド 109)

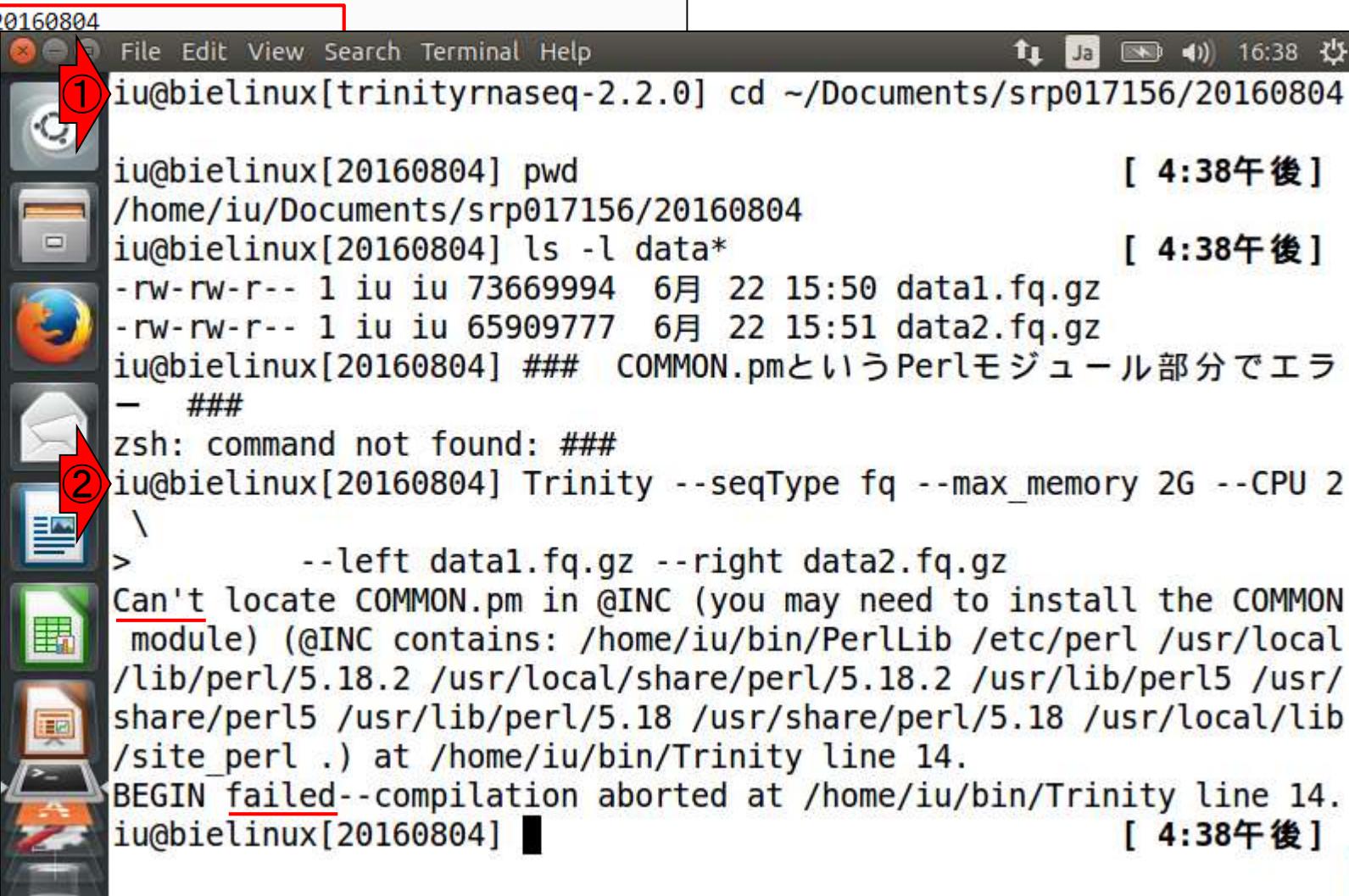
とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

①ディレクトリ変更して、②Trinityを実行すると、すぐにエラーメッセージが出て終了する。赤下線部分のネガティブな単語を見て、失敗しているのだと気づく

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### COMMON.pmというPerlモジュール部分でエラー
Trinity --seqType fq --max_memory 2G --CPU 2
--left data1.fq.gz
--right data2.fq.gz

### COMMON.pm問題は解決。
~/Downloads/trinityrnaseq-2.2.0
--seqType fq --max_memory 2G --CPU 2
--left data1.fq.gz
--right data2.fq.gz

### 「CTRL + C」で脱出し、
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount
```



```
iu@bielinux[trinityrnaseq-2.2.0] cd ~/Documents/srp017156/20160804
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data*
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
iu@bielinux[20160804] ### COMMON.pmというPerlモジュール部分でエラー
- #####
zsh: command not found: #####
iu@bielinux[20160804] Trinity --seqType fq --max_memory 2G --CPU 2
\
> --left data1.fq.gz --right data2.fq.gz
Can't locate COMMON.pm in @INC (you may need to install the COMMON
module) (@INC contains: /home/iu/bin/PerlLib /etc/perl /usr/local/
/lib/perl/5.18.2 /usr/local/share/perl/5.18.2 /usr/lib/perl5 /usr/
share/perl5 /usr/lib/perl/5.18 /usr/share/perl/5.18 /usr/local/lib
/site_perl .) at /home/iu/bin/Trinity line 14.
BEGIN failed--compilation aborted at /home/iu/bin/Trinity line 14.
iu@bielinux[20160804] [ 4:38午後]
```

# 色々試しながら実行1

## 色々試しながら実行1 (スライド 109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### COMMON.pmというPerlモジュール部分でエラー
Trinity --seqType fq --max_memory 2G --CPU 2
--left data1.fq.gz
--right data2.fq.gz

### COMMON.pm問題は解決。
~/Downloads/trinityrnaseq-2.2.0/ Trinity --seqType fq --max_
--left data1.fq.gz
--right data2.fq.gz

### 「CTRL + C」で脱出し、
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount
```



```
iu@bielinux[trinityrnaseq-2.2.0] cd ~/Documents/srp017156/20160804
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data*
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz
iu@bielinux[20160804] ### COMMON.pmというPerlモジュール部分でエラー
- #####
zsh: command not found: #####
iu@bielinux[20160804] Trinity --seqType fq --max_memory 2G --CPU 2
\
> --left data1.fq.gz --right data2.fq.gz
Can't locate COMMON.pm in @INC (you may need to install the COMMON
module) (@INC contains ① /home/iu/bin/PerlLib /etc/perl /usr/local/
/lib/perl/5.18.2 /usr/local/share/perl/5.18.2 /usr/lib/perl5 /usr/
share/perl5 /usr/lib/perl/5.18 /usr/share/perl/5.18 /usr/local/lib/
/site_perl .) at /home/iu/bin/Trinity line 14.
BEGIN failed--compilation aborted at /home/iu/bin/Trinity line 14.
iu@bielinux[20160804] [ 4:38午後]
```

①COMMON.pmというのが原因らしい。  
.pmは、perl module関連ファイル。  
…ふとこのようなメッセージを克服した過去の記憶が蘇る(第4回W15-6)

# 色々試しながら実行1

## 色々試しながら実行1 (スライド 109)

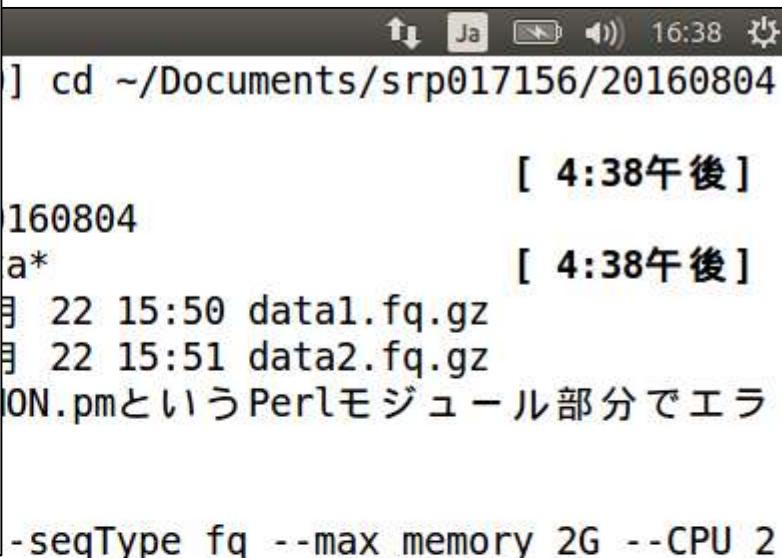
とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力してTrinityを実行。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### COMMON.pmというPerlモジュール部分でエラー ####
Trinity --seqType fq --max_memory 2G --CPU 2 \
    --left data1.fq.gz --right data2.fq.gz

### COMMON.pm問題は解決。が、Phase 1のJellyfishの所でフリーズ状態に... ####
~/Downloads/trinityrnaseq-2.2.0/Trinity \
    --seqType fq --max_memory 2G --CPU 2 \
    --left data1.fq.gz --right data2.fq.gz

### 「CTRL + C」で脱出し、計算途中だったTrinity関連のものを念のため削除 ####
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount
```

①当時は実行ファイル(この場合はTrinity)の相対パス指定(第4回W15-7)で問題が解決した。赤下線で示すように当時の成功体験を頼りにそれを踏襲して再度トライ



```
[ 4:38午後 ]
] cd ~/Documents/srp017156/20160804
160804
a*
月 22 15:50 data1.fq.gz
月 22 15:51 data2.fq.gz
ON.pmというPerlモジュール部分でエラ
-seqType fq --max_memory 2G --CPU 2
```



```
\ 
> --left data1.fq.gz --right data2.fq.gz
Can't locate COMMON.pm in @INC (you may need to install the COMMON
module) (@INC contains: /home/iu/bin/PerlLib /etc/perl /usr/local/
/lib/perl/5.18.2 /usr/local/share/perl/5.18.2 /usr/lib/perl5 /usr/
share/perl5 /usr/lib/perl/5.18 /usr/share/perl/5.18 /usr/local/lib
/site_perl .) at /home/iu/bin/Trinity line 14.
BEGIN failed--compilation aborted at /home/iu/bin/Trinity line 14.
iu@bielinux[20160804] [ 4:38午後 ]
```

# 色々試しながら実行1



## • 色々試しながら実行1 (スライド 109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### COMMON.pmというPerlモジュールが見つかりません。
```

```
Trinity --seqType fq --max_bam_size 1G  
--left data1.fq.gz
```

```
### COMMON.pm問題は解決しました。
```

```
~/Downloads/trinityrnaseq-2.2.0/Trinity  
--seqType fq --max_bam_size 1G  
--left data1.fq.gz
```

```
### 「CTRL + C」で脱出し、
```

```
ls -lt | head
```

```
rm -rf trinity_out_dir
```

```
ls -l *readcount
```

```
rm -f *readcount
```

```
iu@bielinux[20160804] ~/Downloads/trinityrnaseq-2.2.0/Trinity \  
> Trinity --seqType fq --max_memory 2G --CPU 2 \  
> --left data1.fq.gz --right data2.fq.gz  
Trinity version: v2.2.0
```

# 色々試しながら実行1

約1分後の状態。①Phase 1というところになったようだ。順調



## 色々試しながら実行1 (スライド 109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### COMMON.pmというPerlモジュール
```

```
Trinity --seqType fq --maxNC1 100  
--left data1.fq.gz
```

```
### COMMON.pm問題は解決。
```

```
~/Downloads/trinityrnaseq-2.2.0  
--seqType fq --maxNC1 100  
--left data1.fq.gz
```

```
### 「CTRL + C」で脱出し、
```

```
ls -lt | head
```

```
rm -rf trinity_out_dir
```

```
ls -l *readcount
```

```
rm -f *readcount
```

```
rp017156/20160804/trinity_out_dir/chrysalis  
-----  
Trinity Phase 1: Clustering of RNA-Seq Reads  
-----  
①  
Converting input files. (in parallel)  
Monday, June 27, 2016: 17:12:  
10      CMD: gunzip -c /home/iu/Documents/srp017156/20160804/data1.fq.gz | /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/fa  
stool/fastool --append /1 --to-fasta >> left.fa 2> /home/iu/Docume  
nts/srp017156/20160804/data1.fq.gz.readcount  
Monday, June 27, 2016: 17:12:10 CMD: gunzip -c /home/iu/Documents/  
srp017156/20160804/data2.fq.gz | /home/iu/Downloads/trinityrnaseq-  
2.2.0/trinity-plugins/fastool/fastool --append /2 --to-fasta >> ri  
ght.fa 2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readco  
unt
```

# 色々試しながら実行1

#### • 色々試しながら実行1(スライド109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-end Trinityを実行。

cd ~/Documents/srp017156/20160804

11

`lε =1 data*`

```
is -i data  
### COMMON.pmというPerlモ  
Trinity --seqType fq --max  
--left data1.fq.gz
```

### COMMON rm問題は解決。力

```
~/Downloads/trinityrnaseq-  
    --seqType fq --max  
    --left data1.fq.gz
```

### 「CTRL + C」で脱出

`ls -lt | head`

pm pf trinity cut dip

l6\_l \*readcount

Is -1 \*readcount  
nm f \*readcount

```
20160804 環境の変更と4GB以上の動画  
File Edit View Search Terminal Help 17:13  
nts/srp017156/20160804/data1.fq.gz.readcount  
Monday, June 27, 2016: 17:12:10 CMD: gunzip -c /home/iu/Documents/  
srp017156/20160804/data2.fq.gz | /home/iu/Downloads/trinityrnaseq-  
2.2.0/trinity-plugins/fastool/fastool --append /2 --to-fasta >> ri  
ght.fa 2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readco  
unt  
-conversion of 977202 from FQ to FA format succeeded.  
-conversion of 977202 from FQ to FA format succeeded.  
Monday, June 27, 2016: 17:12:30 CMD: touch left.fa.ok right.fa.ok  
Monday, June 27, 2016: 17:12:30 CMD: cat left.fa right.fa > both.f  
a  
Monday, June 27, 2016: 17:12:44 CMD: touch both.fa.ok  
-----  
① ----- Jellyfish -----  
-- (building a k-mer catalog from reads) --  
-----  
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug  
ins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 273227412 --canon  
ical both.fa  
②
```

さらに約30秒後の状態。①Jellyfishというプログラムが動いているようだ。②この状態のままでフリーズ(何も変化がなくなる)しているので、ここまで見届けたら「CTRL + C」で強制終了し、この状態から脱出する。ちなみに仮想環境のメモリを4GBにしていると動きます

# 色々試しながら実行1

「CTRL + C」で脱出した後の状態。①の1番左側の^Cが「CTRL + C」に相当する部分。重い計算をしているためか、反応が鈍い。気長に待つべし

## 色々試しながら実行1 (スライド 109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### COMMON.pmというPerlモジュール
```

```
Trinity --seqType fq --max_bam_size 1000000000  
--left data1.fq.gz
```

```
### COMMON.pm問題は解決。
```

```
~/Downloads/trinityrnaseq-2.2.0/trinity-plugins/fastool/fastool --append /2 --to-fasta > right.fa  
2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readcount
```

```
--seqType fq --max_bam_size 1000000000  
--left data1.fq.gz
```

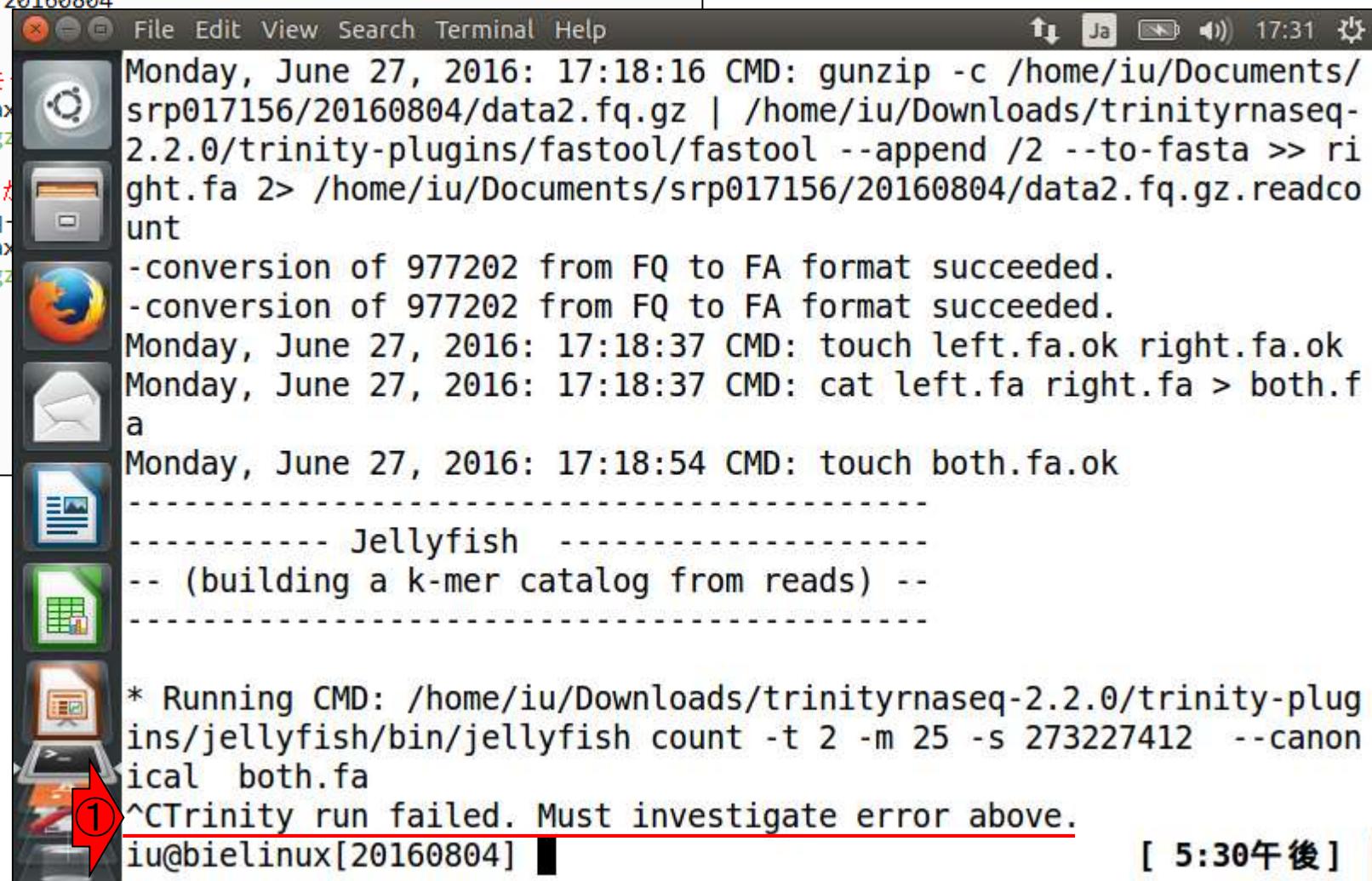
```
### 「CTRL + C」で脱出し、
```

```
ls -1t | head
```

```
rm -rf trinity_out_dir
```

```
ls -l *readcount
```

```
rm -f *readcount
```



```
Monday, June 27, 2016: 17:18:16 CMD: gunzip -c /home/iu/Documents/srp017156/20160804/data2.fq.gz | /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/fastool/fastool --append /2 --to-fasta > right.fa 2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readcount
-conversion of 977202 from FQ to FA format succeeded.
-conversion of 977202 from FQ to FA format succeeded.
Monday, June 27, 2016: 17:18:37 CMD: touch left.fa.ok right.fa.ok
Monday, June 27, 2016: 17:18:37 CMD: cat left.fa right.fa > both.fa
Monday, June 27, 2016: 17:18:54 CMD: touch both.fa.ok
-----
----- Jellyfish -----
-- (building a k-mer catalog from reads) --
-----
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 273227412 --canonical both.fa
^CTrinity run failed. Must investigate error above.
```

iu@bielinux[20160804] [ 5:30午後 ]

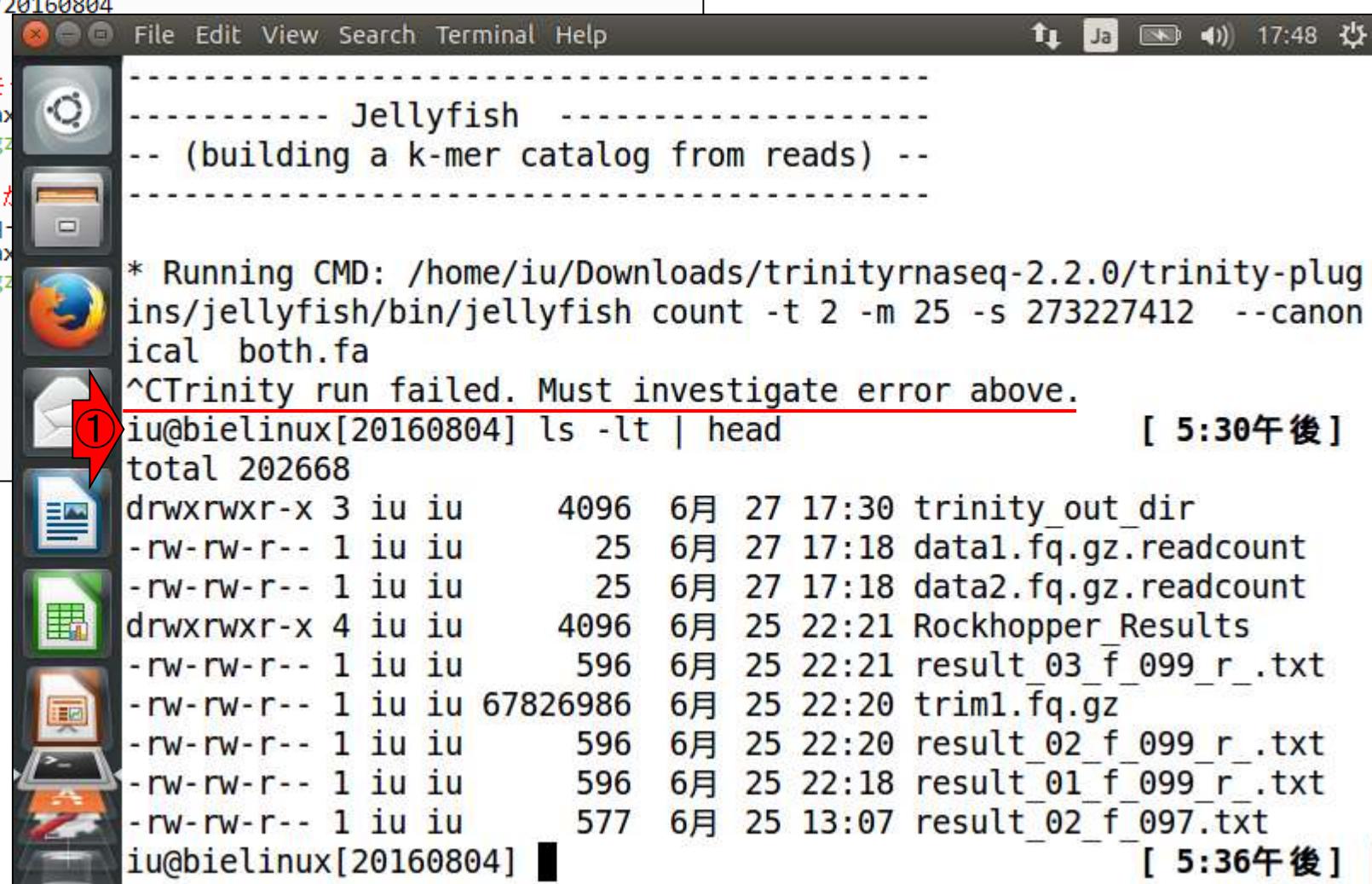
## 強制終了時の注意

#### • 色々試しながら実行1(スライド109)

とりあえずFaQC実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力Trinityを実行。

①のような感じでTrinity実行時に作成されたものを削除しておくべし！理由：それらが残っているがために、その後うまくいくコマンドを打っても失敗する場合がある（私はこれで何度かハマった経験があります）

```
cd ~/Documents/srp017156/20160804  
pwd  
ls -l data*  
### COMMON.pmというPerlモジュールが見つかりました。  
Trinity --seqType fq --max_seqlen 100  
--left data1.fq.gz  
  
### COMMON.pm問題は解決。たった1行の修正で問題が解消されました。  
~/Downloads/trinityrnaseq-2.0.0beta2 --seqType fq --max_seqlen 100  
--left data1.fq.gz  
  
### 「CTRL + C」で脱出し、ターミナルを閉じます。  
ls -lt | head  
rm -rf trinity_out_dir  
ls -l *readcount  
rm -f *readcount
```



①私はこんな感じで削除しましたが、最終的に目的を達成できればなんでもいいです

# 削除

- 色々試しながら実行1 (スライド109)

とりあえずFaQCs実行直後の977,202リードからなるdata1.fq.gzとdata2.fq.gzのpaired-endを入力としてTrinityを実行。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### COMMON.pmというPerlモジュール
```

```
Trinity --seqType fq --max_bam_size 1000000000 --left data1.fq.gz
```

```
### COMMON.pm問題は解決。た
```

```
~/Downloads/trinityrnaseq-2.2.0/trinityrnaseq-2.2.0.pl --seqType fq --max_bam_size 1000000000 --left data1.fq.gz
```

```
### 「CTRL + C」で脱出し、
```

```
ls -lt | head
```

```
rm -rf trinity_out_dir
```

```
ls -l *readcount
```

```
rm -f *readcount
```

The terminal window shows the following sequence of commands and their outputs:

```
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug-ins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 273227412 --canonical both.fa
^CTrinity run failed. Must investigate error above.
iu@bielinux[20160804] ls -lt | head [ 5:30午後]
total 202668
drwxrwxr-x 3 iu iu 4096 6月 27 17:30 trinity_out_dir
-rw-rw-r-- 1 iu iu 25 6月 27 17:18 data1.fq.gz.readcount
-rw-rw-r-- 1 iu iu 25 6月 27 17:18 data2.fq.gz.readcount
drwxrwxr-x 4 iu iu 4096 6月 25 22:21 Rockhopper_Results
-rw-rw-r-- 1 iu iu 596 6月 25 22:21 result_03_f_099_r_.txt
-rw-rw-r-- 1 iu iu 67826986 6月 25 22:20 trim1.fq.gz
-rw-rw-r-- 1 iu iu 596 6月 25 22:20 result_02_f_099_r_.txt
-rw-rw-r-- 1 iu iu 596 6月 25 22:18 result_01_f_099_r_.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:07 result_02_f_097.txt
iu@bielinux[20160804] rm -rf trinity_out_dir [ 5:36午後]
iu@bielinux[20160804] ls -l *readcount [ 6:03午後]
-rw-rw-r-- 1 iu iu 25 6月 27 17:18 data1.fq.gz.readcount
-rw-rw-r-- 1 iu iu 25 6月 27 17:18 data2.fq.gz.readcount
iu@bielinux[20160804] rm -f *readcount [ 6:03午後]
iu@bielinux[20160804]
```

Red boxes highlight the removal of the Trinity output directory and the readcount files. Red numbers '1' are placed near the 'rm' command in each highlighted section.

# 色々試しながら実行2

## 色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804  
pwd  
ls -l data*  
### なんとなくメモリ1Gにして実行したらうまくいった  
~/Downloads/trinityrnaseq-2.2.0/Trinity \  
    --seqType fq --max_memory 1G --CPU 2 \  
    --left data1.fq.gz --right data2.fq.gz  
  
### 確認 ###  
ls -lt | head  
cd trinity_out_dir  
pwd  
ls  
grep -c ">" Trinity.fasta  
grep -v ">" Trinity.fasta | wc  
cp Trinity.fasta ../Trinity1.fasta
```



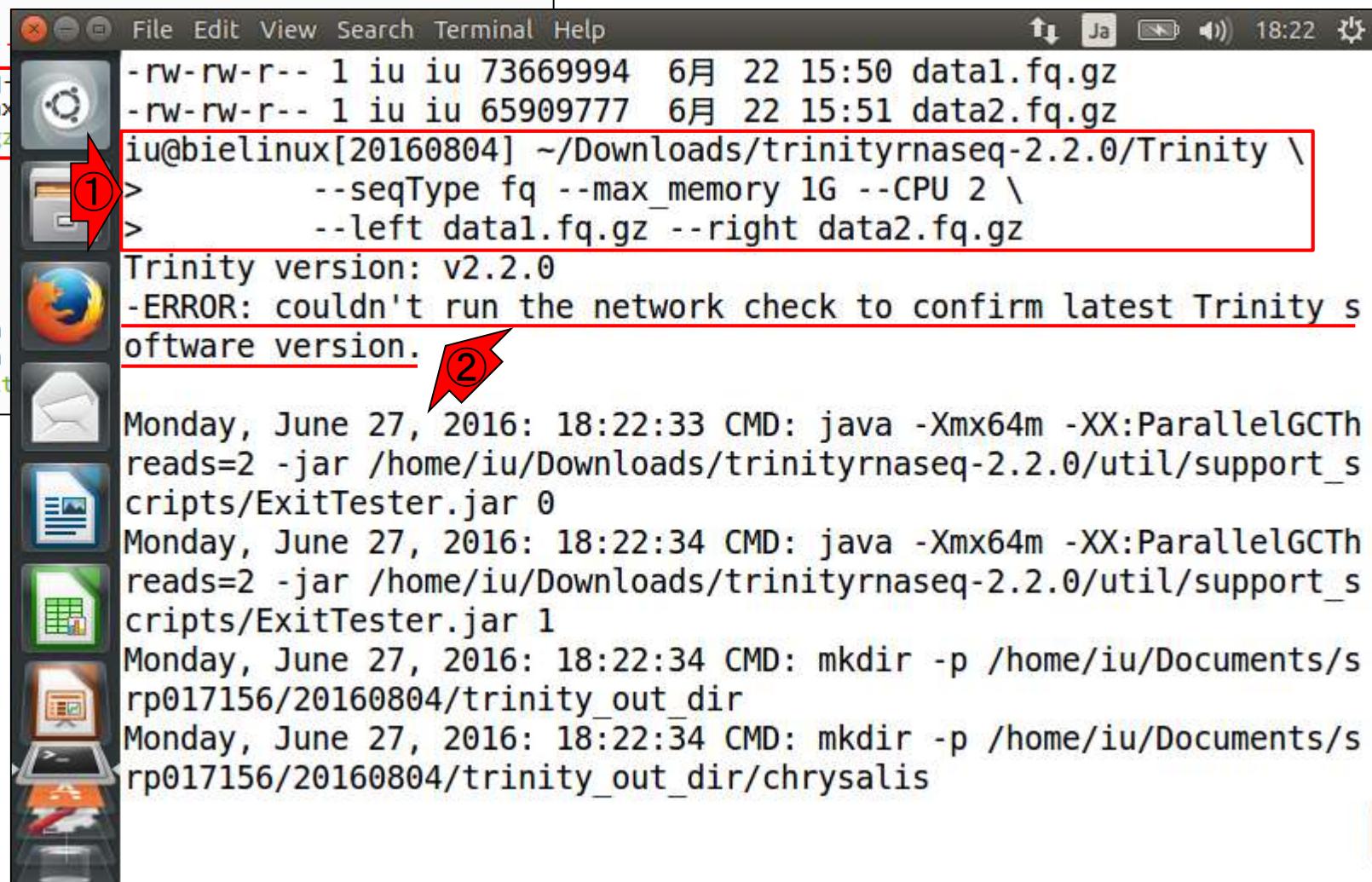
全く非論理的な展開だが、①のコメント部分に書いてあることが全て。まるで説明のつかない意味不明なことも起こりうるという例です。②をコピペ実行。**約85分**

# 途中経過1

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804  
pwd  
ls -l data*  
### なんなくメモリ1Gにし  
~/Downloads/trinityrnaseq-2.2.0/Trinity \\  
--seqType fq --max  
--left data1.fq.gz  
  
### 確認 ###  
ls -lt | head  
cd trinity_out_dir  
pwd  
ls  
grep -c ">" Trinity.fasta  
grep -v ">" Trinity.fasta  
cp Trinity.fasta ..//Trinity1.fasta
```



```
-rw-rw-r-- 1 iu iu 73669994 6月 22 15:50 data1.fq.gz  
-rw-rw-r-- 1 iu iu 65909777 6月 22 15:51 data2.fq.gz  
iu@bielinux[20160804] ~/Downloads/trinityrnaseq-2.2.0/Trinity \  
> --seqType fq --max_memory 1G --CPU 2 \  
> --left data1.fq.gz --right data2.fq.gz  
Trinity version: v2.2.0  
-ERROR: couldn't run the network check to confirm latest Trinity s  
oftware version.  
  
Monday, June 27, 2016: 18:22:33 CMD: java -Xmx64m -XX:ParallelGCThreads=2 -jar /home/iu/Downloads/trinityrnaseq-2.2.0/util/support_scripts/ExitTester.jar 0  
Monday, June 27, 2016: 18:22:34 CMD: java -Xmx64m -XX:ParallelGCThreads=2 -jar /home/iu/Downloads/trinityrnaseq-2.2.0/util/support_scripts/ExitTester.jar 1  
Monday, June 27, 2016: 18:22:34 CMD: mkdir -p /home/iu/Documents/srp017156/20160804/trinity_out_dir  
Monday, June 27, 2016: 18:22:34 CMD: mkdir -p /home/iu/Documents/srp017156/20160804/trinity_out_dir/chrysalis
```

①コピペ実行から約1分後の状態。②  
こういうエラーメッセージが出ることもあるようですが、気にしなくていいです

# 途中経過2

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### なんなくメモリ1Giにしてみる
~/Downloads/trinityrnaseq-2.2.0/trinity-plugins/fastool --append /2 --to-fasta >> right.fa
2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readcount
--seqType fq --max 1000000000
--left data1.fq.gz
```

```
### 確認 ###
```

```
ls -lt | head
cd trinity_out_dir
pwd
```

```
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ..
```

```
File Edit View Search Terminal Help
2.2.0/trinity-plugins/fastool --append /2 --to-fasta >> right.fa 2> /home/iu/Documents/srp017156/20160804/data2.fq.gz.readcount
-conversion of 977202 from FQ to FA format succeeded.
-conversion of 977202 from FQ to FA format succeeded.
Monday, June 27, 2016: 18:22:55 CMD: touch left.fa.ok right.fa.ok
Monday, June 27, 2016: 18:22:55 CMD: cat left.fa right.fa > both.fa
Monday, June 27, 2016: 18:23:18 CMD: touch both.fa.ok
-----
① ----- Jellyfish -----
-- (building a k-mer catalog from reads) --
-----
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 119835723 --canonical both.fa
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/jellyfish/bin/jellyfish dump -L 1 mer_counts.jf > jellyfish.kmers.fa
```

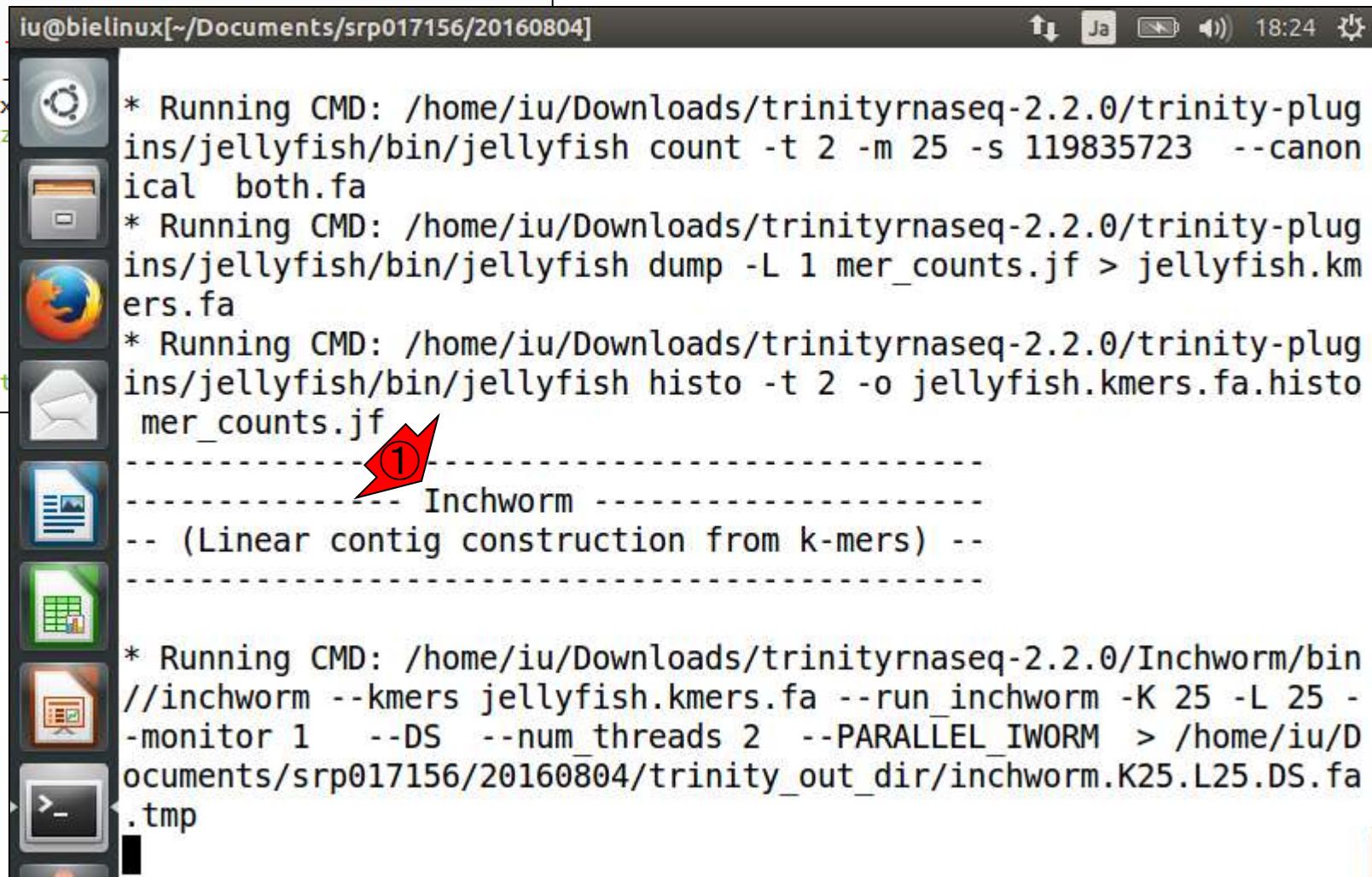
①コピペ実行から約2分後の状態。②前回はいつまで経っても出なかった赤枠部分が出て驚くが、これまでと違うhopefulな状況に対して素直に喜ぶ

# 途中経過3

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### なんとなくメモリ1Giにしてみる
~/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 119835723 --canonical both.fa
### 確認 ####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ..
```



iu@bielinux[~/Documents/srp017156/20160804] 18:24

```
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 119835723 --canonical both.fa
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish dump -L 1 mer_counts.jf > jellyfish.kmers.fa
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish histo -t 2 -o jellyfish.kmers.fa.histo
mer_counts.jf
-----
①----- Inchworm -----
-- (Linear contig construction from k-mers) --
-----
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/Inchworm/bin
//inchworm --kmers jellyfish.kmers.fa --run_inchworm -K 25 -L 25 -
-monitors 1 --DS --num_threads 2 --PARALLEL_IWORM > /home/iu/D
ocuments/srp017156/20160804/trinity_out_dir/inchworm.K25.L25.DS.fa
.tmp
```

確かにコピペ実行から約3分後の状態。  
①Inchwormという見たことのある単語のフェーズに入ったようだ。順調



# 途中経過4

色々試しながら実行2

Trinity 実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### なんなくメモリ1Gにしてみる
~/Downloads/trinityrnaseq-2.2.0 --seqType fq --max_bam_size 1G --left data1.fq.gz
### 確認 ####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ../Trinit
```

```
File Edit View Search Terminal Help
- reads_list_file partitioned_reads.files.list --CPU 1 --max_memory 1G --seqType fa --trinity_complete --full_cleanup > recursive_trinity.cmds
Monday, June 27, 2016: 18:36:09 CMD: touch recursive_trinity.cmds.
ok
Monday, June 27, 2016: 18:36:09 CMD: touch recursive_trinity.cmds.
ok
-----
----- Trinity Phase 2: Assembling Clusters of Reads -----
-----
-----
Monday, June 27, 2016: 18:36:09 CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plugins/parafly/bin/ParaFly -c recursive_trinity.cmds -CPU 2 -v
Number of Commands: 2307
succeeded(21) 0.910273% completed.
```

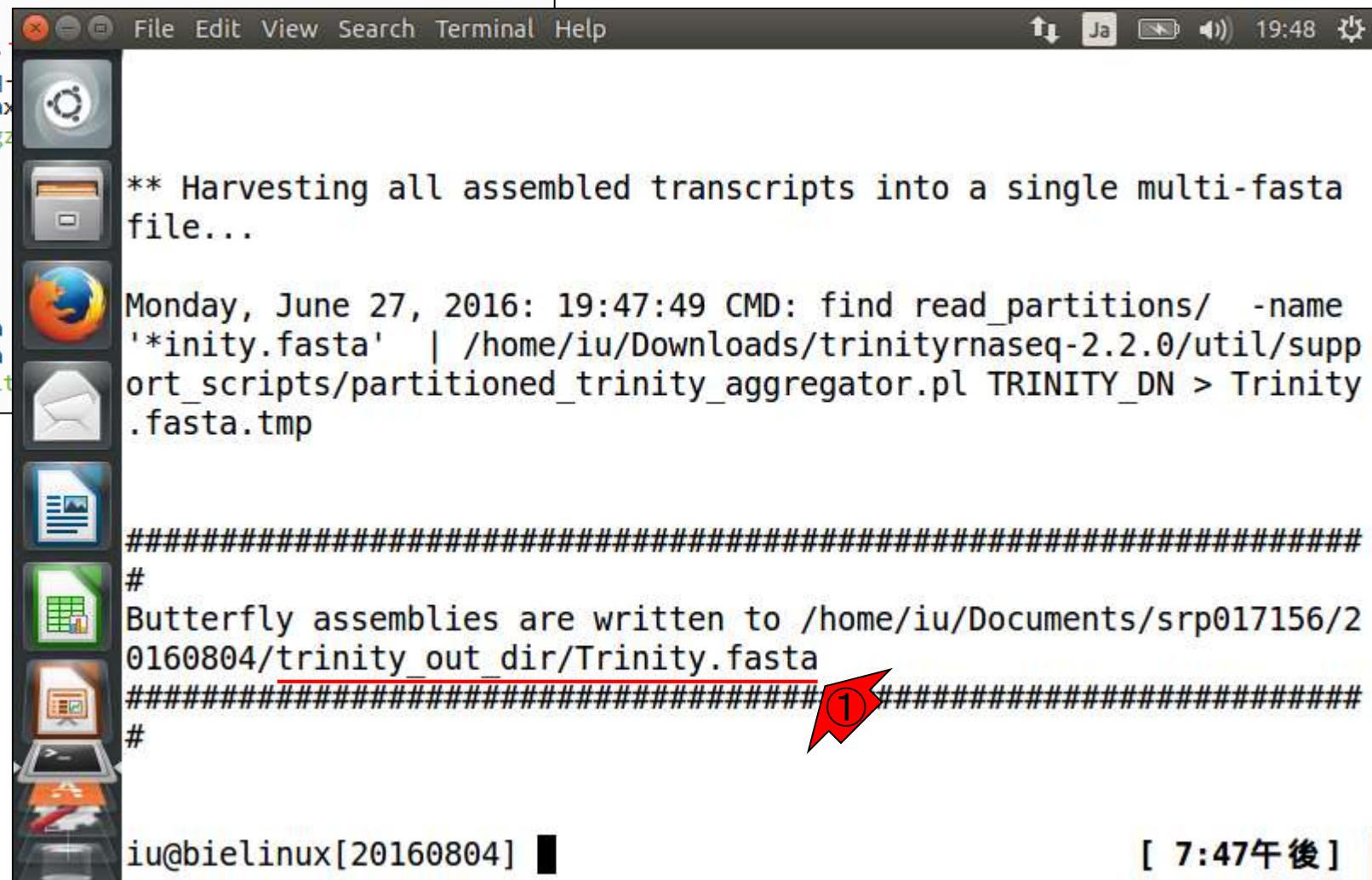
# 終了時の状態

(18:22頃スタートで)19:47に終了したので、このときは約85分。①Trinityのアセンブリ結果ファイルは、trinity\_out\_dirディレクトリ内にあるTrinity.fasta

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### なんなくメモリ1Giにしてしまった
~/Downloads/trinityrnaseq-2.2.0/util/support_scripts/partitioned_trinity_aggregator.pl --seqType fq --max_bam_size 1000000000 --left data1.fq.gz
### 確認 #####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ..//Trinity.fasta
```



\*\* Harvesting all assembled transcripts into a single multi-fasta file...

Monday, June 27, 2016: 19:47:49 CMD: find read\_partitions/ -name '\*inity.fasta' | /home/iu/Downloads/trinityrnaseq-2.2.0/util/support\_scripts/partitioned\_trinity\_aggregator.pl TRINITY\_DN > Trinity.fasta.tmp

#####
#  
# Butterfly assemblies are written to /home/iu/Documents/srp017156/20160804/trinity\_out\_dir/Trinity.fasta  
#####  
#

iu@bielinux[20160804] [ 7:47 午後 ]

# 確認

- 色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603個)に衝撃を受ける。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### なんなくメモリ1Giにして実行したらうまくいった #####
~/Downloads/trinityrnaseq-2.2.0/Trinity \
    --seqType fq --max_memory 1G --CPU 2 \
    --left data1.fq.gz --right data2.fq.gz

### 確認 #####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta | wc
cp Trinity.fasta ../Trinity1.fasta
```

```
Help
19:48
#####
e written to /home/iu/Documents/srp017156/20160804/Trinity.fasta
#####
```

```
iu@bielinux[20160804] ls -lt | head
total 202668
drwxrwxr-x 4 iu iu 4096 6月 27 19:47 trinity_out_dir
-rw-rw-r-- 1 iu iu 25 6月 27 18:22 data1.fq.gz.readcount
-rw-rw-r-- 1 iu iu 25 6月 27 18:22 data2.fq.gz.readcount
drwxrwxr-x 4 iu iu 4096 6月 25 22:21 Rockhopper_Results
-rw-rw-r-- 1 iu iu 596 6月 25 22:21 result_03_f_099_r_.txt
-rw-rw-r-- 1 iu iu 67826986 6月 25 22:20 trim1.fq.gz
-rw-rw-r-- 1 iu iu 596 6月 25 22:20 result_02_f_099_r_.txt
-rw-rw-r-- 1 iu iu 596 6月 25 22:18 result_01_f_099_r_.txt
-rw-rw-r-- 1 iu iu 577 6月 25 13:07 result_02_f_097.txt
iu@bielinux[20160804]
```

[ 7:47午後]

# 確認

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### なんとなくメモリ1Giにし
~/Downloads/trinityrnaseq-
--seqType fq --max
--left data1.fq.gz
### 確認 ###
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ../Trinit
```



赤枠内のコードを一気にコピペした結果。**①得られた配列数(転写物数; 2,603個)の多さに衝撃**を受ける。**②総塩基数は** $2,724,160 - 45,921 = 2,678,239$  bp**と一般的な乳酸菌を含むバクテリアのゲノム**サイズに近い値。これは転写物の総塩基数なので多過ぎ、というのが率直な感想。しかしこの中から、様々なフィルタリングによって落とされていくので、初期値としてはこれくらいでもいいのかもしれない。**アセンブリ結果もヒトによって異なる**ようだ。例えば2,300個とか…

```
iu@bielinux[20160804] cd trinity_out_dir
iu@bielinux[trinity_out_dir] pwd
/home/iu/Documents/srp017156/20160804/trinity_out_dir
iu@bielinux[trinity_out_dir] ls
both.fa
both.fa.ok
both.fa.read_count
chrysalis
inchworm.K25.L25.DS.fa
inchworm.K25.L25.DS.fa.finished
inchworm.kmer_count
jellyfish.kmers.fa
jellyfish.kmers.fa.histo
left.fa.ok
iu@bielinux[trinity_out_dir] grep -c ">" Trinity.fasta
2603
iu@bielinux[trinity_out_dir] grep -v ">" Trinity.fasta | wc
45921 45921 2724160
iu@bielinux[trinity_out_dir] cp Trinity.fasta ../Trinity1.fasta
iu@bielinux[trinity_out_dir]
```

partitioned\_reads.files.list  
partitioned\_reads.files.list.ok  
**read\_partitions**  
recursive\_trinity.cmds  
recursive\_trinity.cmds.completed  
recursive\_trinity.cmds.ok  
right.fa.ok  
Trinity.fasta  
Trinity.timing

[ 9:06 午後 ]

# 確認

色々試しながら実行2

Trinity実行結果ファイル([Trinity1.fasta](#); 約3MB)の転写物数(2,603)

```
cd ~/Documents/srp017156/20160804
```

```
pwd
```

```
ls -l data*
```

```
### なんなくメモリ1Giにしてみる
~/Downloads/trinityrnaseq
--seqType fq --max_bam_size 1000000000
--left data1.fq.gz
```

```
### 確認 ###
```

```
ls -lt | head
```

```
cd trinity_out_dir
```

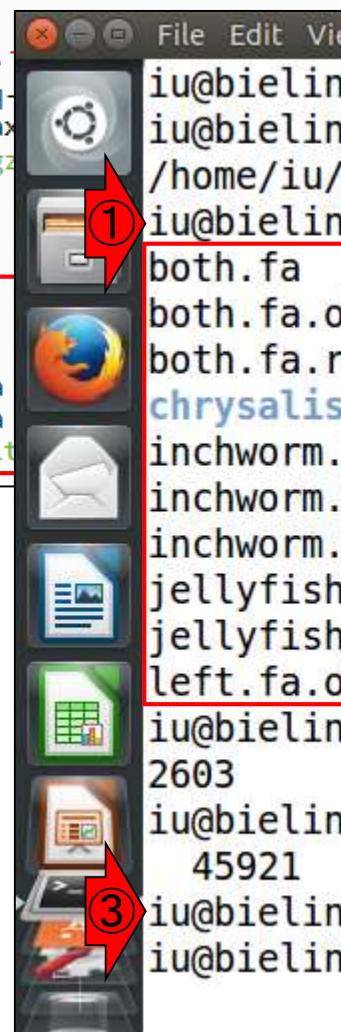
```
pwd
```

```
ls
```

```
grep -c ">" Trinity.fasta
```

```
grep -v ">" Trinity.fasta
```

```
cp Trinity.fasta ../Trinity1.fasta
```



ちなみに①ls実行結果の赤枠内には、実行ログファイルはなさそう。②.timingというファイルを見つけ、「この中に計算時間情報が含まれているのだろう」と思い、実際にそうであることを確認したりする(おそらくこれが事実上のログファイル)。③Rockhopper2と同じく、常にTrinity.fastaという同じファイル名になるので、次のアセンブリ結果で上書きされないように、1つ上のディレクトリ上にTrinity1.fastaでコピー。④が同じものです(アセンブリ失敗したヒト用; ~/Desktop/backupにもあり)

```
iu@bielinux[trinity_out_dir] ls [ 9:06午後]
both.fa
both.fa.ok
both.fa.read_count
chrysalis
inchworm.K25.L25.DS.fa
inchworm.K25.L25.DS.fa.finished
inchworm.kmer_count
jellyfish.kmers.fa
jellyfish.kmers.fa.histo
left.fa.ok
partitioned_reads.files.list
partitioned_reads.files.list.ok
read_partitions
recursive_trinity.cmds
recursive_trinity.cmds.completed
recursive_trinity.cmds.ok
right.fa.ok
Trinity.fasta
Trinity.timing [ 9:06午後]

iu@bielinux[trinity_out_dir] grep -c ">" Trinity.fasta
2603
iu@bielinux[trinity_out_dir] grep -v ">" Trinity.fasta | wc
45921 45921 2724160
iu@bielinux[trinity_out_dir] cp Trinity.fasta ../Trinity1.fasta
iu@bielinux[trinity_out_dir]
```

# 色々試しながら実行3

## 色々試しながら実行3

Trinity実行結果ファイル([Trinity2.fasta](#); 約3MB)の転写物数(3,090個; 別のPCでやると3,087個)から、strand情報を考慮すると配列数が増える(たまたまかも)と学習する。

```
cd ~/Documents/srp017156/20160804
### 計算途中じゃなくともTrinity関連のものを予め削除しないといけないよう #####
### これをやらずに実行すると、すぐに計算が終わるのでやらねばと学習する #####
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *count
①

### SRR616268の実験情報SRX204226の記述(Illumina Stranded Std PE)から #####
### orientation情報も入れたほうがよいと判断して--SS_lib_type FRを追加 #####
~/Downloads/trinityrnaseq-2.2.0/Trinity \
    --seqType fq --max_memory 1G --CPU 2 --SS_lib_type FR
    --left data1.fq.gz --right data2.fq.gz
③

### 確認 #####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta | wc
cp Trinity.fasta ../Trinity2.fasta
```

今回取り扱っている①SRR616268を含む最近のRNA-seqは、②strand情報もわかるような実験プロトコルで行われるようです。このようなデータの場合は、③--SS\_lib\_type FRオプションをTrinity実行時に追加するようです。ここはコピペ実行しないで！

②

# 色々試しながら実行3

## 色々試しながら実行3

Trinity実行結果ファイル([Trinity2.fasta](#); 約3MB)の転写物数(3,090個; 別のPCでやると3,089個した)から、strand情報を考慮すると配列数が増える(たまたまかも)と学習する。

```
cd ~/Documents/srp017156/20160804
### 計算途中じゃなくともTrinity関連のものを予め削除しないといけないようだ #####
### これをやらずに実行すると、すぐに計算が終わるのでやらねばと学習する #####
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount

### SRR616268の実験情報SRX204226の記述(Illumina Stranded Std PE)から #####
### orientation情報も入れたほうがよいと判断して--SS lib_type FRを追加 #####
~/Downloads/trinityrnaseq-2.2.0/Trinity \
    --seqType fq --max_memory 1G --CPU 2 --SS lib_type FR \
    --left data1.fq.gz --right data2.fq.gz

### 確認 #####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta | wc
cp Trinity.fasta ../Trinity2.fasta
```

ここも眺めるだけ。結論としては、色々試すのは重要ですね、ということ。先に①をコピペ実行し、1分ほどで計算終了しておかしいと思う。そして、②直前のアセンブリ結果と同じ数値情報が得られたら、先に③をやらないといけないと学習します

③

①

②

# 色々試しながら実行3

参考  
コピペ実行結果。①配列数は3,090、②総塩基数は $2,825,449 - 47,826 = 2,777,623$  bp。ここもおそらくヒトによって異なるだろう

## 色々試しながら実行3

Trinity実行結果ファイル([Trinity2.fasta](#); 約3MB)の転写物数(3,090個; 別のPCでやると3,089個でした)から、strand情報を考慮すると配列数が増える(たまたまかも)と学習する。

```
cd ~/Documents/srp017156/20160804
### 計算途中じゃなくともTrinityを実行する。
### これをやらずに実行するとreadcountが増える。
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount

### SRR616268の実験情報SRX0000000をorientation情報も入れる。
~/Downloads/trinityrnaseq-0.1.14/bin/trinityrnaseq \
    --seqType fq --max_bam_size 1000000000 \
    --left data1.fq.gz

### 確認 ####
ls -lt | head
cd trinity_out_dir
pwd
ls
grep -c ">" Trinity.fasta
grep -v ">" Trinity.fasta
cp Trinity.fasta ../Trinity2.fasta
```

```
File Edit View Search Terminal Help
- rw-rw-r-- 1 iu iu      596  6月 25 22:18 result_01_f_099_r_.txt
iu@bielinux[20160804] cd trinity_out_dir          [ 1:05午後]
iu@bielinux[trinity_out_dir] pwd                  [ 1:05午後]
/home/iu/Documents/srp017156/20160804/trinity_out_dir
iu@bielinux[trinity_out_dir] ls                  [ 1:05午後]
both.fa
both.fa.ok
both.fa.read_count
chrysalis
inchworm.K25.L25.fa
inchworm.K25.L25.fa.finished
inchworm.kmer_count
jellyfish.kmers.fa
jellyfish.kmers.fa.histo
left.fa.ok
iu@bielinux[trinity_out_dir] grep -c ">" Trinity.fasta
3090
iu@bielinux[trinity_out_dir] grep -v ">" Trinity.fasta | wc
47826 47826 2825449
iu@bielinux[trinity_out_dir] cp Trinity.fasta ../Trinity2.fasta
iu@bielinux[trinity_out_dir] [ 1:05午後]
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# apt-getでインストール

参考

## apt-getでインストール

apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd
```

```
pwd
```

```
where Trinity
```

```
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity
```

```
sudo apt-get install trinityrnaseq
```

```
where Trinity
```

```
Trinity --version
```

```
/usr/bin/Trinity --version
```

The screenshot shows a terminal window with a dark theme. On the left is a vertical dock with icons for a terminal, file manager, browser (Firefox), email, and file viewer. The terminal window has a title bar with the user's name and session details. The main area contains the following command history:

```
iu@bielinux[trinity_out_dir] cd  
iu@bielinux[iu] pwd  
/home/iu  
iu@bielinux[iu] where Trinity  
/home/iu/bin/Trinity  
/home/iu/bin/Trinity  
iu@bielinux[iu] ~/Downloads/trinityrnaseq-2.2.0/Trinity --version  
Trinity version: v2.2.0  
-currently using the latest production release of Trinity.
```

Red arrows point to specific parts of the terminal output:

- ① Points to the command "where Trinity" which lists multiple "Trinity" entries in the user's home directory.
- ② Points to the output of "Trinity --version" which shows the version as "v2.2.0".

①apt-getでインストールする前の状態をおさらい。②ver. 2.2.0がインストールされている。スライド120で出たようなERRORメッセージが出ることもあるが気にしない。スライドを見るだけ

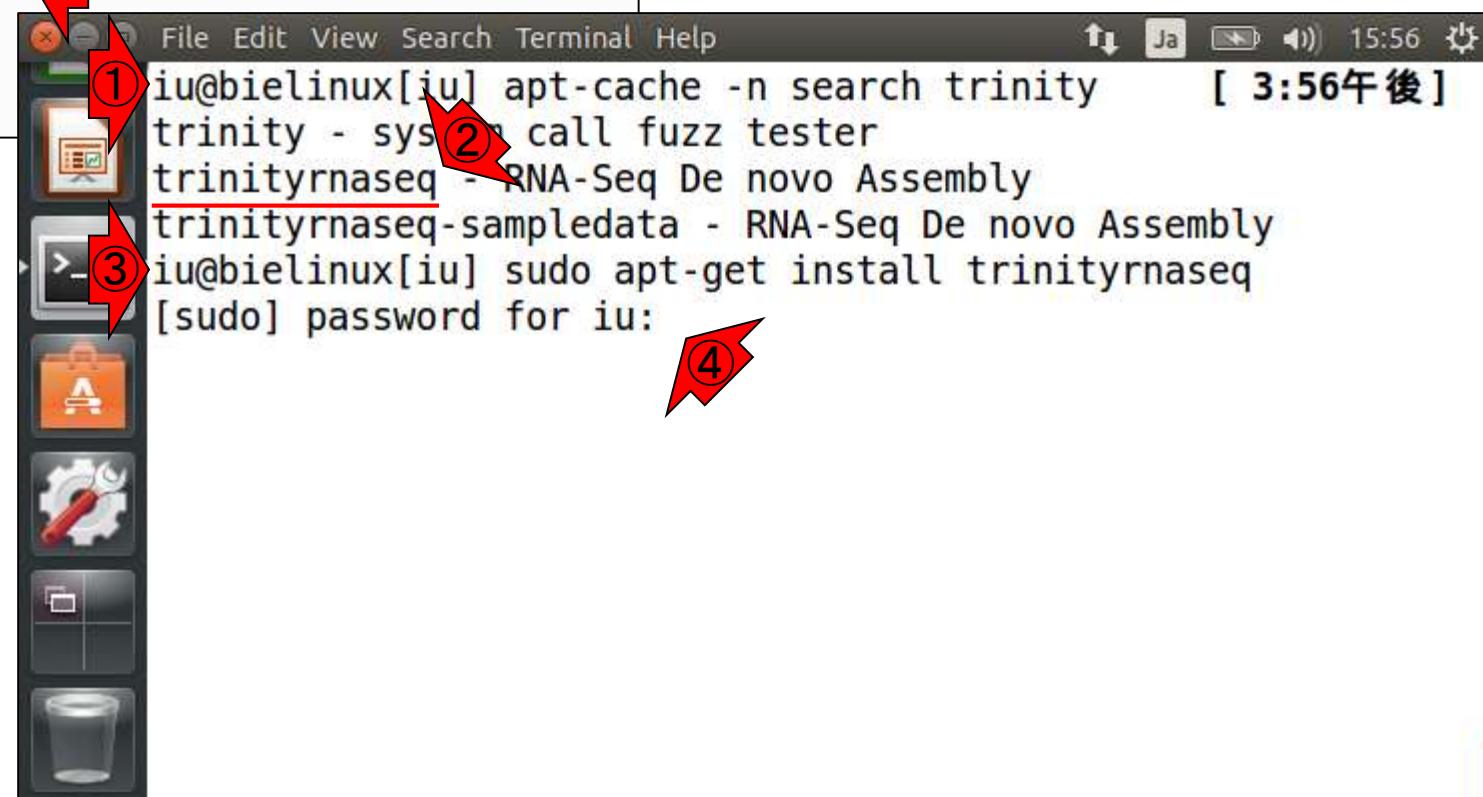
# apt-getでインストール

①trinityというキーワードを含むソフトウェア名をリストアップ。②正式名称がtrinityrnaseqであることを確認し、③apt-getでインストールを実行。④rootのパスワードはpass1409。[見るだけ](#)

## apt-getでインストール

apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-1.0/Trinity --version  
  
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq  
  
where Trinity  
Trinity --version  
/usr/bin/Trinity --version
```



# apt-getでインストール

①何か聞かれているが、基本的に思考停止してyでよい

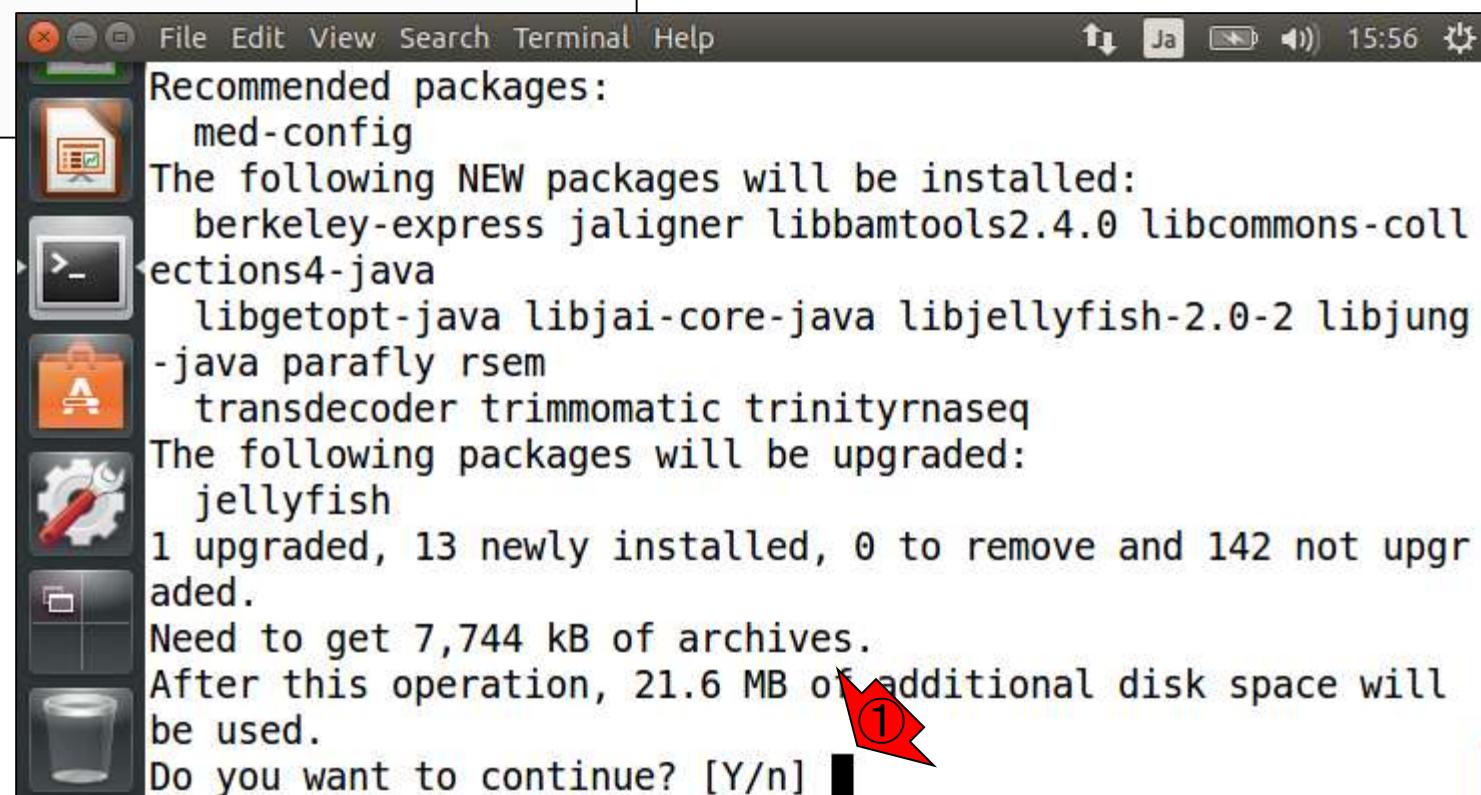
## apt-getでインストール

apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq
```

```
where Trinity  
Trinity --version  
/usr/bin/Trinity --version
```



# apt-getでインストール

参考

## apt-getでインストール

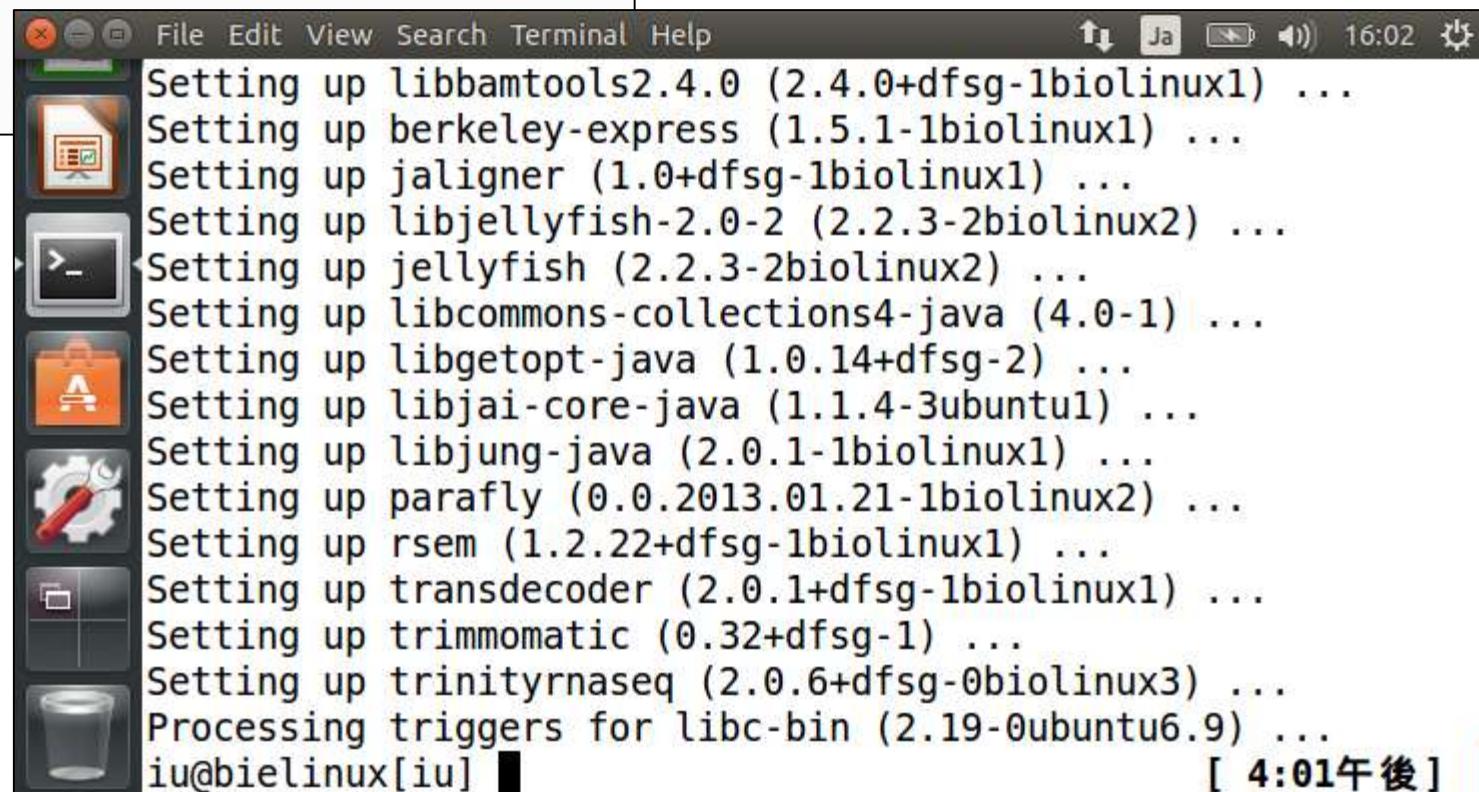
apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq
```

```
where Trinity  
Trinity --version  
/usr/bin/Trinity --version
```

(東大有線LAN環境だからかどうかは不明だが)数分でインストール完了。「E: Failed to fetch http:…」とか「E: Unable to fetch some …」などと出ることもある。この場合はホストOSがOKでも、ゲストOSがネットワークにつながっていないこともあるのでそれが理由。こういうときは、大抵ゲストOSのFirefoxもつながらないので納得できる。対策: 時間を空けるとか、一旦ゲストOSの再起動を行うとか…



# apt-getでインストール

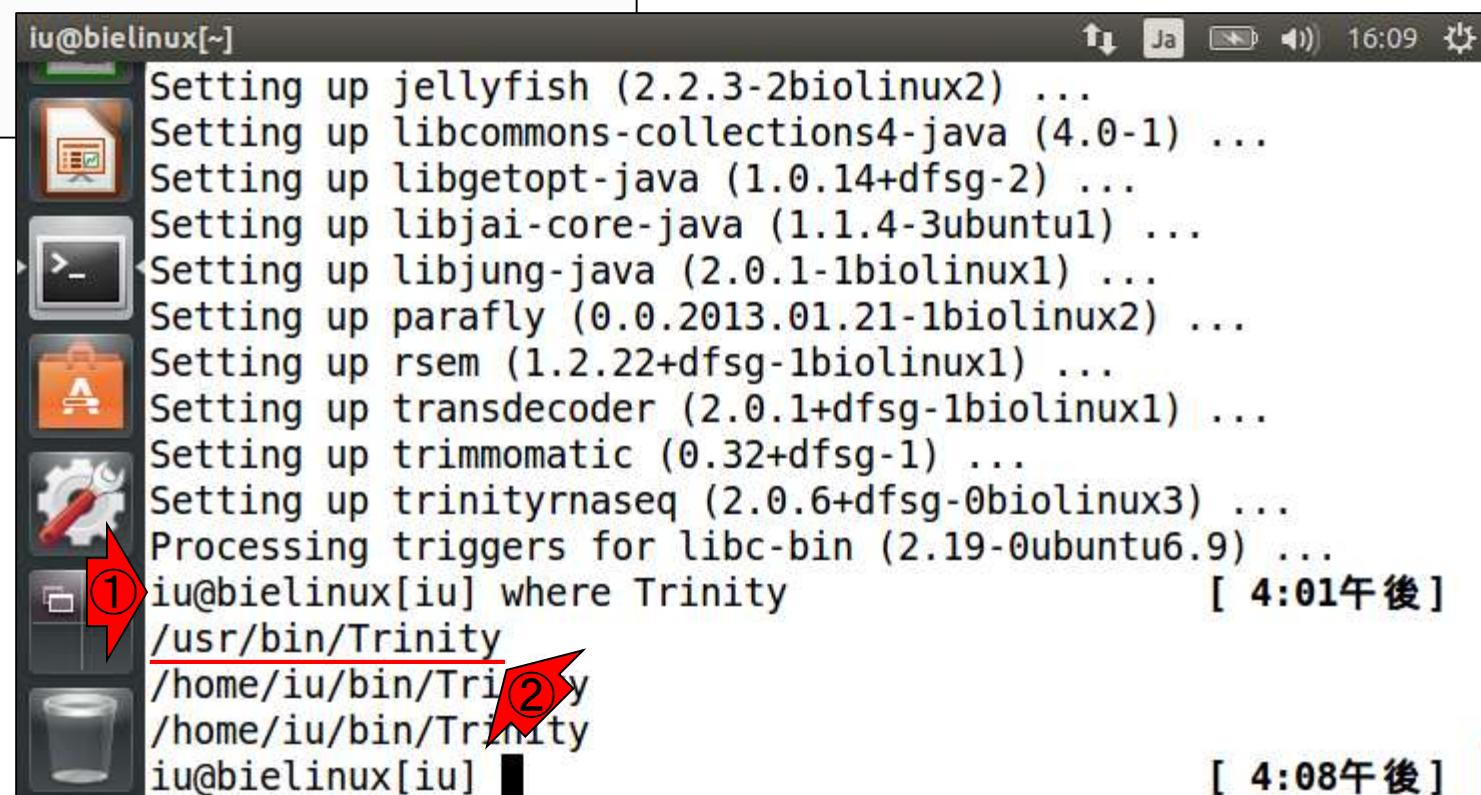
参考

## apt-getでインストール

apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq  
  
where Trinity ①  
Trinity --version  
/usr/bin/Trinity --version
```



```
iu@bielinux[~] Setting up jellyfish (2.2.3-2biolinux2) ...  
Setting up libcommons-collections4-java (4.0-1) ...  
Setting up libgetopt-java (1.0.14+dfsg-2) ...  
Setting up libjai-core-java (1.1.4-3ubuntul) ...  
Setting up libjung-java (2.0.1-1biolinux1) ...  
Setting up parafly (0.0.2013.01.21-1biolinux2) ...  
Setting up rsem (1.2.22+dfsg-1biolinux1) ...  
Setting up transdecoder (2.0.1+dfsg-1biolinux1) ...  
Setting up trimmomatic (0.32+dfsg-1) ...  
Setting up trinityrnaseq (2.0.6+dfsg-0biolinux3) ...  
Processing triggers for libc-bin (2.19-0ubuntu6.9) ...  
iu@bielinux[iu] where Trinity [ 4:01午後]  
/usr/bin/Trinity ②  
/home/iu/bin/Trinity  
/home/iu/bin/Trinity  
iu@bielinux[iu] [ 4:08午後]
```

①パスが通っているかを確認。② /usr/bin/Trinityが追加されたようだ

# apt-getでインストール

参考

## apt-getでインストール

apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqであることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq
```

```
where Trinity  
Trinity --version  
/usr/bin/Trinity --version
```

```
File Edit View Search Terminal Help 16:09  
Setting up trinityrnaseq (2.0.6+dfsg-0biolinux3) ...  
Processing triggers for libc-bin (2.19-0ubuntu6.9) ...  
iu@bielinux[iu] where Trinity [ 4:01 午後]  
/usr/bin/Trinity  
/home/iu/bin/Trinity  
/home/iu/bin/Trinity  
iu@bielinux[iu] Trinity --version [ 4:08 午後]  
Can't locate COMMON.pm in @INC (you may need to install the  
COMMON module) (@INC contains: /home/iu/bin/PerlLib /etc/perl  
/usr/local/lib/perl/5.18.2 /usr/local/share/perl/5.18.2 /u  
sr/lib/perl5 /usr/share/perl5 /usr/lib/perl/5.18 /usr/share/  
perl/5.18 /usr/local/lib/site_perl .) at /home/iu/bin/Trinit  
y line 14.  
BEGIN failed--compilation aborted at /home/iu/bin/Trinity li  
ne 14.  
iu@bielinux[iu] [ 4:09 午後]
```

①依然として「Trinity」のみではバージョン情報は表示されないが、②のフルパスを利用すればいいと学習

# apt-getでインストール

参考

## apt-getでインストール

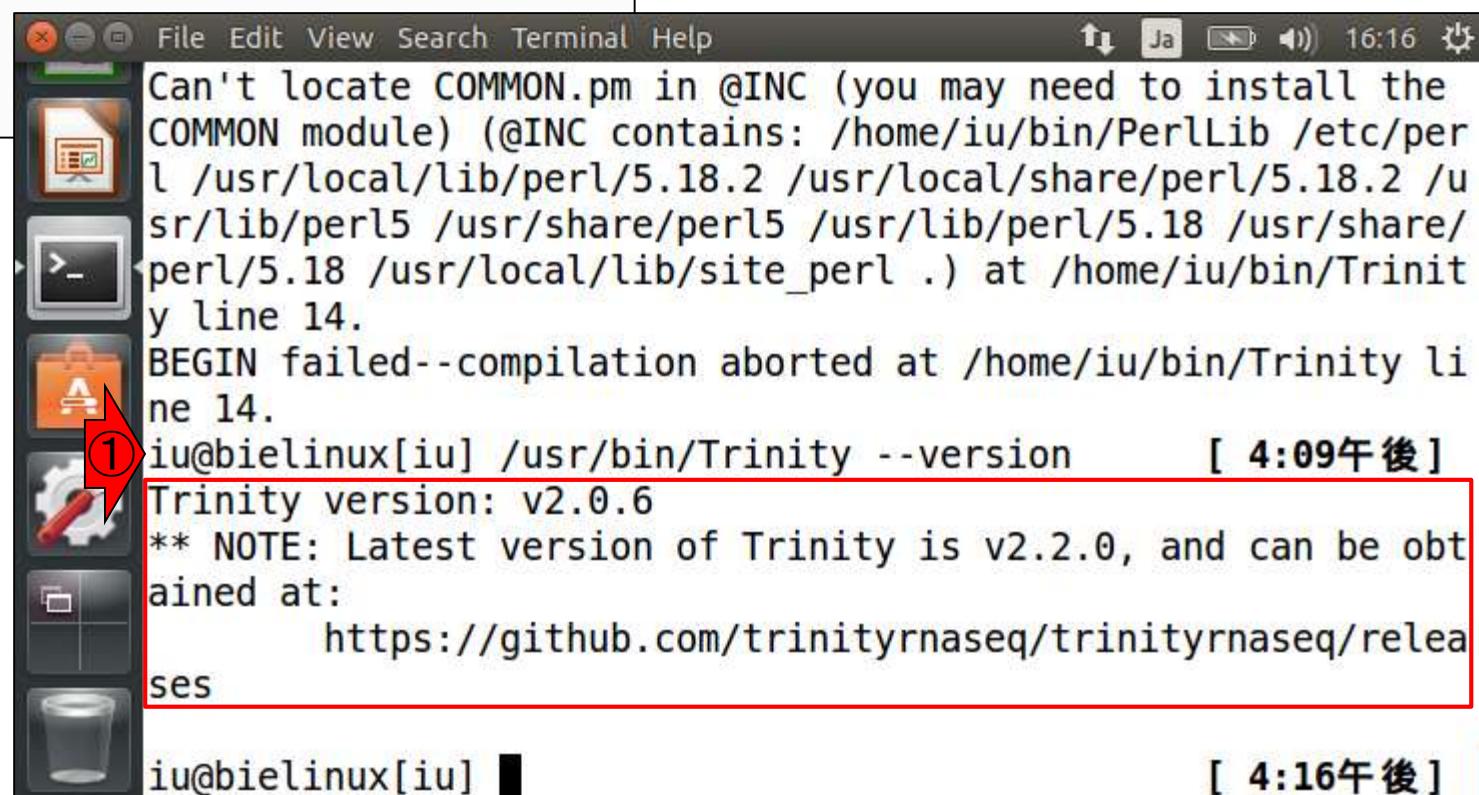
apt-cacheでtrinityのキーワードを含むソフトウェア名をリストアップ。正式名称がtrinityrnaseqることを確認し、sudo apt-get installを実行。

```
cd  
pwd  
where Trinity  
~/Downloads/trinityrnaseq-2.2.0/Trinity --version
```

```
apt-cache -n search trinity  
sudo apt-get install trinityrnaseq
```

```
where Trinity  
Trinity --version  
/usr/bin/Trinity --version
```

①apt-get経由でインストールしたTrinityはver. 2.0.6である。最新版(ver. 2.2.0)がインストールされない現象は、sra-toolkitのときと同じ(第7回W4-3; 2016.08.03スライド153)。一般論としては、最新版だとうまく動かないこともある(2016.08.12追加)



```
File Edit View Search Terminal Help  
Can't locate COMMON.pm in @INC (you may need to install the  
COMMON module) (@INC contains: /home/iu/bin/PerlLib /etc/per  
l /usr/local/lib/perl/5.18.2 /usr/local/share/perl/5.18.2 /u  
sr/lib/perl5 /usr/share/perl5 /usr/lib/perl/5.18 /usr/share/  
perl/5.18 /usr/local/lib/site_perl .) at /home/iu/bin/Trinit  
y line 14.  
BEGIN failed--compilation aborted at /home/iu/bin/Trinity li  
ne 14.  
iu@bielinuix[iu] /usr/bin/Trinity --version [ 4:09 午後]  
Trinity version: v2.0.6  
** NOTE: Latest version of Trinity is v2.2.0, and can be obt  
ained at:  
https://github.com/trinityrnaseq/trinityrnaseq/relea  
ses  
iu@bielinuix[iu] [ 4:16 午後]
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# この後の展開は

## (Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、ranscriptom、正規化、発現量  
(last modified 2016/06/03, since 2011)

- ・前処理 | フィルタリング | paired-end | 共通リード
- ・アセンブル | について (last modified 2014/06/20)
- ・アセンブル | ゲノム用 (last modified 2016/03/24)
- ・アセンブル | ランスクリプトーム(転写物)用 (la...  
・マッピング | について (last modified 2016/04/07)
- ・マッピング | basic aligner (last modified 2014/08/11)
- ・マッピング | splice-aware aligner (last modified 2014/07/09)
- ・マッピング | Bisulfite sequencing用 (last modified 2014/07/09)
- ・マッピング | (ESTレベルの長さの)contig (last modified 2014/06/24)
- ・マッピング | 基礎 (last modified 2013/06/19)
- ・マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Gaidatzis et al., 2011)

①乳酸菌データについては、バクテリア専用のRockhopper2がいいに決まっているだろうが、念のため②有名なTrinityをインストールしてやってみたら、バクテリア用でもないのに配列数が相当増加したという衝撃の結果をお話。次に第5回でも紹介した③Bridgerを使おうと思ったがサンプルデータ実行段階でこけたという話が当日だったが、Bridgerが必要とするboostパッケージ(が古いバージョンになったおかげで)をapt-get経由でインストールしたらうまくいくという情報が受講生から寄せられました。よって、Bridgerの後継である④BinPackerもこのやり方でうまくいくはずです(2016.08.12追加)

- ・Multiple-k: Surget-Groba and Montoya-Burgos, Genome Res., 2010
- ・Trans-ABvSS: Robertson et al., Nat Methods, 2010
- ・Rnnotator: Martin et al., BMC Genomics, 2010
- ・Trinity: Grabherr et al., Nat Biotechnol, 2011
- ・Oases: Schulz et al., Bioinformatics, 2012
- ・EBARDenovo: Chu et al., Bioinformatics, 2013
- ・BRANCH: Bao et al., Bioinformatics, 2013
- ・IDBA-tran: Peng et al., Bioinformatics, 2013
- ・SOAPdenovo-Trans: Xie et al., Bioinformatics, 2014
- ・VTBuilder: Archer et al., BMC Bioinformatics, 2014
- ・Rockhopper 2(バクテリア用): Tjaden B, Genome Biol., 2015
- ・DETONATE(RSEM-EVAL): Li et al., Genome Biol., 2014
- ・Bridger: Chang et al., Genome Biol., 2015
- ・IFRAT: Mbandi et al., BMC Bioinformatics, 2015
- ・SCERNA(主に植物): Honaas et al., PLoS One, 2016
- ・BinPacker: Liu et al., PLoS Comput Biol., 2016

Review、ガイドライン、バイオライン系:

- ・Review: Martin and Wang, Nat Rev Genet., 2011
- ・ガイドライン: Haznedaroglu et al., BMC Bioinformatics, 2012
- ・Review: Góngora-Castillo, Nat Prod Rep., 2013

# Bridger

• Bridger: Chang et al., Genome Biol., 2015

講義では Bridger\_r2014-12-01.tar.gz (約11MB)を ~/Downloadsにダウンロード済み

① ダウンロードのURLは「[https://sourceforge.net/projects/rnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/rnaseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/rnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz](https://sourceforge.net/projects/rnaseqassembly/files/Bridger_r2014-12-01.tar.gz)」に変更した。

```
cd ~/Downloads  
#wget -c https://sourceforge.net/projects/rnaseqassembly/files/Bridger_r2014-12-01.tar.gz  
pwd  
ls -l Bri*  
tar zxvf Bridger_r2014-12-01.tar.gz  
cd Bridger_r2014-12-01  
ls  
more INSTALL  
less README
```

スライドを見るだけ。①Bridgerのサイトに行き、②最新版(r2014-12-01)をダウンロード。Windowsの場合は右クリックで「ショートカットのコピー」、Macの場合は「リンクをコピー」でURL情報を取得

The screenshot shows a web browser window with the SourceForge website for the RNA-Seq Assembly project. The URL in the address bar is <https://sourceforge.net/projects/rnaseqassembly/files/>. The page displays a 'Get Google Chrome' advertisement and the project's summary information. Below the summary, there is a table of files available for download. The table includes columns for Name, Modified, Size, and Downloads / Week. The latest version, 'Bridger\_r2014-12-01.tar.gz', is highlighted with a red arrow labeled ②. The table shows the following data:

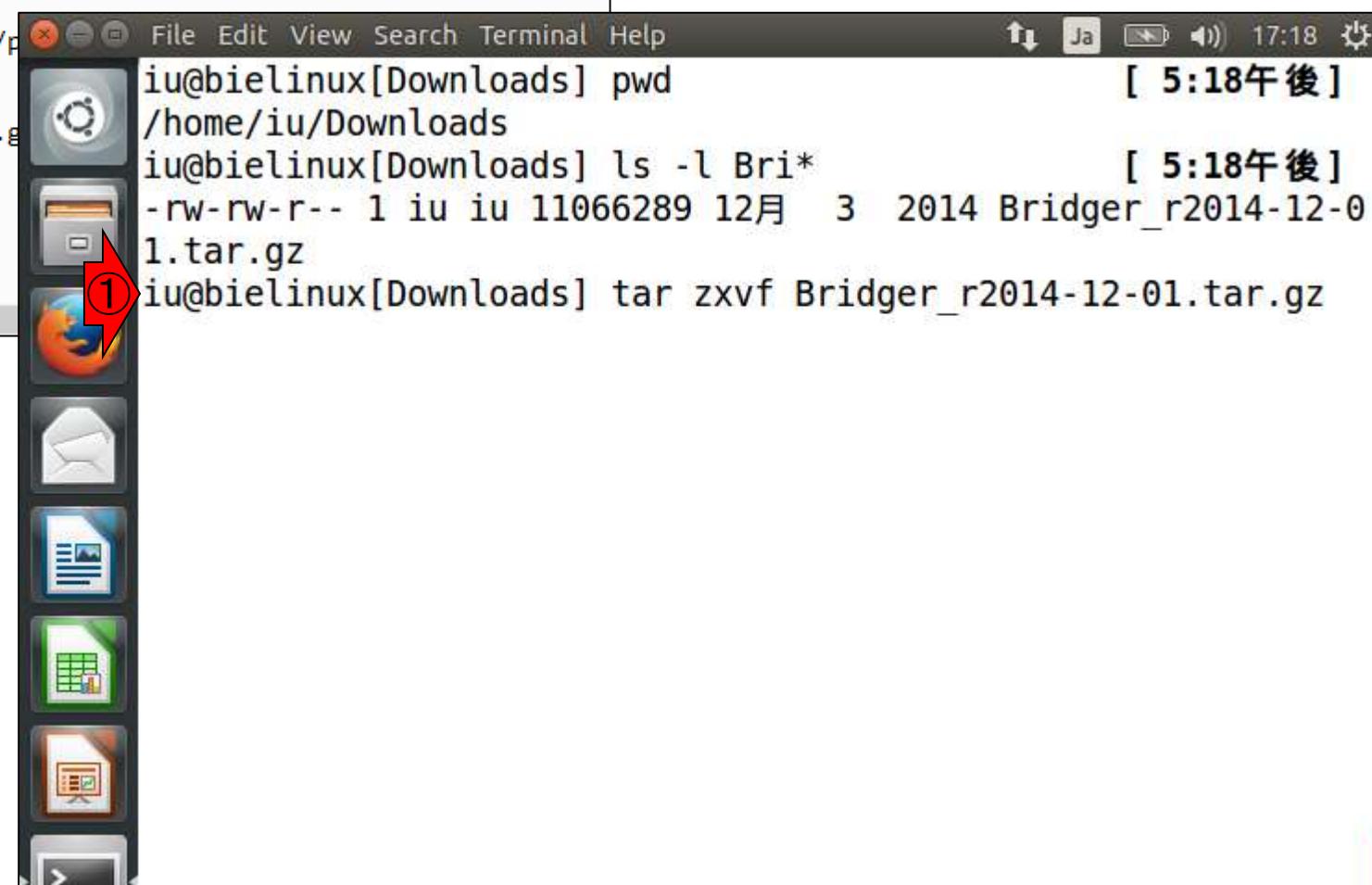
Name	Modified	Size	Downloads / Week
Bridger_r2014-12-01.tar.gz	2014-12-03	11.1 MB	11
Bridger_r2014-11-05.tar.gz	2014-11-09	11.1 MB	1
Bridger_r2013-06-02.tar.gz	2013-06-26	11.1 MB	1
Bridger_r2013-03-21.tar.gz	2013-04-02	2.3 MB	1
Totals: 4 Items		35.5 MB	14

# Bridger

## • Bridger: Chang et al., Genome Biol., 2015

講習会ではBridger\_r2014-12-01.tar.gz(約11MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/mnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/mnaseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/mnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz](https://sourceforge.net/projects/mnaseqassembly/files/Bridger_r2014-12-01.tar.gz)」に変更した。

```
cd ~/Downloads  
#wget -c https://sourceforge.net/p/  
pwd  
ls -l Bri*  
tar zxvf Bridger_r2014-12-01.tar.g  
cd Bridger_r2014-12-01  
ls  
more INSTALL  
less README
```



# Bridger

• Bridger: Chang et al., Genome Biol., 2015

講習会ではBridger\_r2014-12-01.tar.gz(約11MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/rnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/rnaseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/rnaseqassembly/files/Bridger\\_r2014-12-01.tar.gz](https://sourceforge.net/projects/rnaseqassembly/files/Bridger_r2014-12-01.tar.gz)」に変更した。

```
cd ~/Downloads
# wget -c https://sourceforge.net/p/rnaseqassembly/ci/master/tree/Bridger_r2014-12-01
pwd
ls -l Bri*
tar zxvf Bridger_r2014-12-01.tar.gz
cd Bridger_r2014-12-01
ls
more INSTALL
less README
```

```

iu@bielinux[Downloads] cd Bridger_r2014-12-01 [ 5:36午後]
iu@bielinux[Bridger_r2014-12-01] ls [ 5:36午後]
aclocal.m4           ChangeLog          COPYING
AUTHORS             config.h           INSTALL
autom4te.cache       config.h.in        libtool
ax_boost_base.m4    config.log         LICENCE
boost.m4             config.status      m4
Bridger.pl          configure          Makefile
build-aux           configure.ac      Makefile.am
stamp-h1             Makefile          stamp-h1
sample_test          README            src
src                 README            stamp-h1
iu@bielinux[Bridger_r2014-12-01] more INSTALL [ 5:36午後]
```

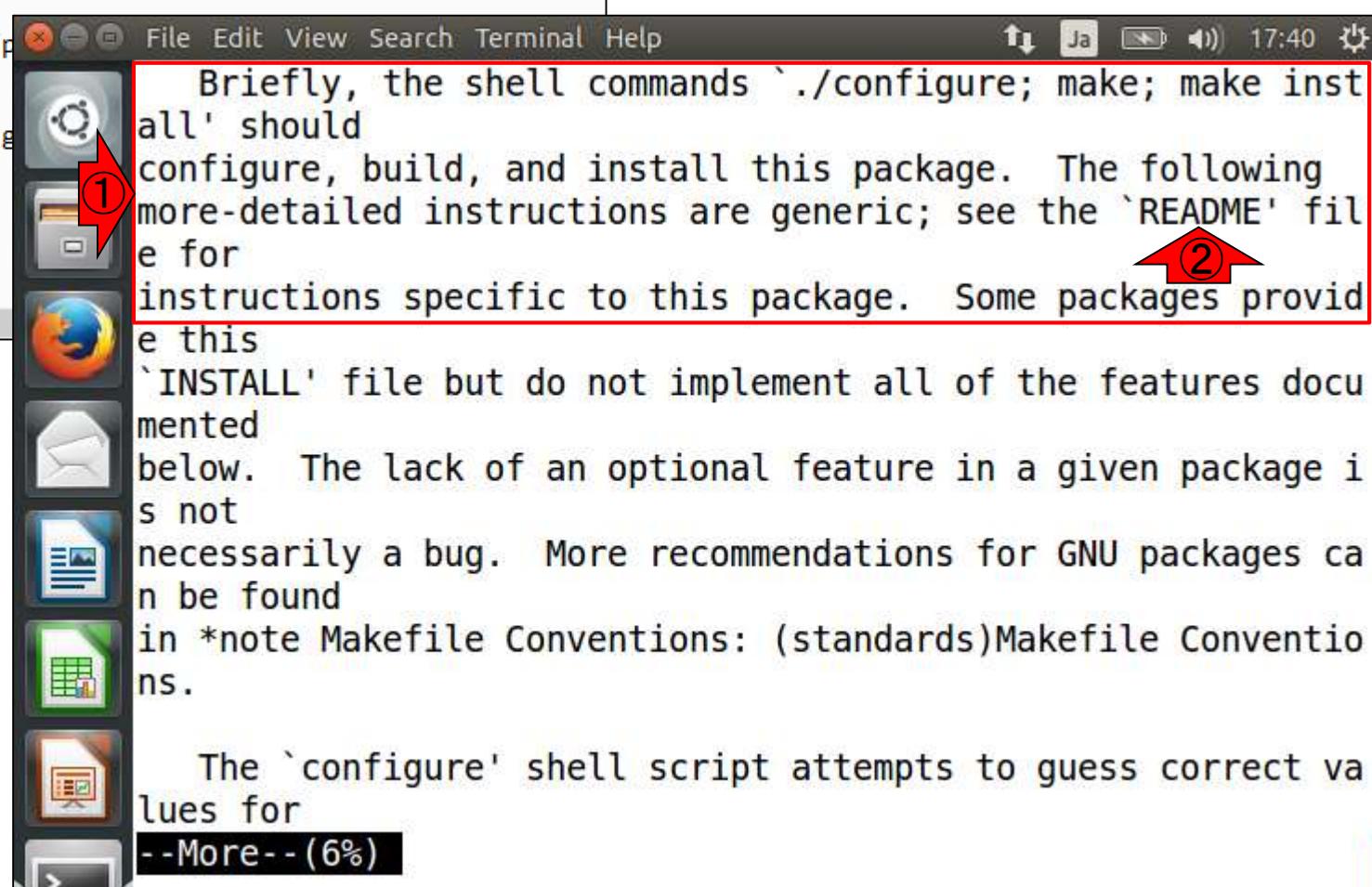
①ディレクトリ変更し、ls。②Makefileがあるので、makeでインストールするのが基本なのだろうと妄想。③INSTALLというファイルがあるので④moreで確認

# more INSTALL

• Bridger: Chang et al., Genome Biol., 2015

講習会ではBridger\_r2014-12-01.tar.gz(約11MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/maseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_r2014-12-01.tar.gz](https://sourceforge.net/projects/maseqassembly/files/Bridger_r2014-12-01.tar.gz)」に変更した。

```
cd ~/Downloads  
#wget -c https://sourceforge.net/p/  
pwd  
ls -l Bri*  
tar zxvf Bridger_r2014-12-01.tar.g  
cd Bridger_r2014-12-01  
ls  
more INSTALL  
less README
```



①あたりを眺めて、インストールの基本は、「./configure; make; make install」という3つの呪文を唱えればいいと学習。②詳細についてはREADMEだと書かれているので、READMEを見にいく。qで抜ける

# less README

## • Bridger: Chang et al., Genome Biol., 2015

講習会ではBridger\_r2014-12-01.tar.gz(約11MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/maseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_r2014-12-01.tar.gz](https://sourceforge.net/projects/maseqassembly/files/Bridger_r2014-12-01.tar.gz)」に変更した。

The screenshot shows a terminal window on a Linux desktop environment. The terminal history is as follows:

```
cd ~/Downloads  
#wget -c https://sourceforge.net/p/maseqassembly/ci/master/tree/Bridger_r2014-12-01  
pwd  
ls -l Bri*  
tar zxvf Bridger_r2014-12-01.tar.gz  
cd Bridger_r2014-12-01  
ls  
more INSTALL  
less README
```

A red arrow points to the terminal window, and another red arrow points to the desktop dock area where the terminal icon is highlighted with a red circle containing the number 1.

The desktop dock icons include:

- Terminal (highlighted with a red circle containing 1)
- File Manager
- File Explorer
- File Viewer
- File Editor
- File Converter
- File Compressor
- File Archiver
- File Reader

Bridgerの「less README」中。①Installationのあたりまでどうにか辿り着く。②Boostというものをインストールしないといけないらしいので、まずはこれに集中しよう

# less README

File Edit View Search Terminal Help

↑ ↓ Ja 19:56

## Installation

①

### 1. Installing Boost

②

a) download latest boost and unpack it.

```
$ tar zxvf boost_1_47_0.tar.gz
```

b) change to the boost directory and run ./bootstrap.sh.

```
$ cd boost_1_47_0  
$ ./bootstrap.sh
```

c) Run ./b2 and set the install path.

```
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
```

# Boost

①「linux boost」で検索した結果。②の日本語の記述内容から概要をつかみ、③が本家サイトなのだろうと予想し、③をクリック

linux boost

すべて ニュース ショッピング 動画 画像 もっと見る 検索ツール

**Boostライブラリのビルト方法 - boostjp**  
[boostjp.github.io/howtobuild.html](https://boostjp.github.io/howtobuild.html)

2015/04/09 - ここでは Boostライブラリのビルト方法について説明します。Windowsとそれ以外に分けて説明します。また、Linuxでは Boostライブラリがディストリビューションによって提供されていることがあります、ここではビルト方法のみを扱います。

**Boost - Wikipedia**  
<https://ja.wikipedia.org/wiki/Boost>

Boost（ブースト）とは、C++の先駆的な開発者のコミュニティ、およびそのコミュニティによって公開されているオープンソースライブラリのことを指す。コミュニティとしての Boost は C++標準化委員会の委員により設立されており、現在でもその多くが構成員として ...  
内容：線型代数・乱数生成・テキストのパース

**Boost C++ Libraries**  
[www.boost.org/](http://www.boost.org/) このページを訳す

Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the Getting Started Guide to download and install Boost. Popular Linux and Unix distributions such as Fedora, Debian, and NetBSD include ...

# Boost

The screenshot shows the homepage of boost.org. At the top, there's a navigation bar with back, forward, and search buttons, followed by the URL http://www.boost.org/ and the title "Boost C++ Libraries". Below the title is a quote: "...one of the most highly regarded and expertly designed C++ library projects in the world." — Herb Sutter and Andrei Alexandrescu, C++ Coding Standards. The main content area features the Boost logo (three blue hexagons) and the word "boost" in lowercase. A large red button labeled "GET BOOST" is prominently displayed. To the right of the button is a sidebar with a "Search" field containing "Google™ Custom Search" and a "Search" button, followed by a "Donate" button and payment method icons (MasterCard, Visa, American Express, Discover, Bank). The sidebar also lists several links under "WELCOME": Getting Started, Download, Libraries, Mailing Lists, Reporting and Fixing Bugs, and Wiki. Below these are sections for "INTRODUCTION", "COMMUNITY", "DEVELOPMENT", and "DOCUMENTATION". A red arrow points from the number ① in the top right corner to the "Download" link in the sidebar.

WELCOME TO BOOST.ORG!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are included in the C++ Standards Committee's Library Technical Report (TR1) and in the new C++11 Standard. C++11 also includes several more Boost libraries in addition to those from TR1. More Boost libraries are proposed for standardization in C++17.

Since 2006 an intimate week long annual conference related to Boost called C++ Now has been held in Aspen, Colorado each May. Boost has been a participant in the annual Google Summer of Code since 2007.

GETTING STARTED

Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the Getting Started Guide to download and install Boost. Popular Linux and Unix distributions such as Fedora, Debian, and NetBSD include pre-built Boost packages. Boost may also already be available on your organization's internal web server.

BACKGROUND

Read on with the introductory material to help you understand what Boost is about and to help in educating your organization about Boost.

COMMUNITY

Boost welcomes and thrives on participation from a variety of individuals and organizations. Many avenues for participation are available in the Boost Community.

DOWNLOADS

# Boost

The screenshot shows the official Boost C++ Libraries download page. At the top, there's a banner with the Boost logo and a quote from Herb Sutter and Andrei Alexandrescu. Below the banner, the 'BOOST DOWNLOADS' section features links for 'Current Release', 'Old Boost Releases', and 'Git Repositories'. A large red button labeled 'GET BOOST' is prominently displayed. To the right of the button is a sidebar with a 'Google Custom Search' bar, a 'Search' button, and a 'Donate' button with payment method icons. The sidebar also contains a navigation menu with sections like WELCOME, INTRODUCTION, DOWNLOAD, NEWS, VERSION HISTORY, LICENSE, PEOPLE, BIBLIOGRAPHY, WHO'S USING BOOST, FAQ, HOW BOOST STARTED, and INDEX. Other sections visible include 'CURRENT RELEASE' (Version 1.61.0), 'OTHER DOWNLOADS' (with a red arrow pointing to it), 'OLD BOOST RELEASES', and 'GIT REPOSITORIES'.

①

②

# Boost

①講習会では~/Downloadsに  
ダウンロード済み。②約85MB

Get Google Chrome  
One browser for all your devices. Fast, free & installs in seconds!

Report a problem with ad content

Home / Browse / Development / Software Development / Boost C++ Libraries / Files

## Boost C++ Libraries

Free peer-reviewed portable C++ source libraries  
Brought to you by: beman\_dawes, danieljames, eric\_niebler, grafik, and 2 others

Summary | Files | Reviews | Support | Wiki | Mailing Lists | News | Discussion | Code

Looking for the latest version? [Download boost\\_1\\_61\\_0.7z \(74.8 MB\)](#)

Home / boost / 1.61.0

Name	Modified	Size	Downloads / Week
boost_1_61_0.tar.bz2	2016-05-13	85.2 MB	3,609
boost_1_61_0.zip	2016-05-13	155.9 MB	3,062
boost_1_61_0.tar.gz	2016-05-13	104.9 MB	2,177
boost_1_61_0.7z	2016-05-13	74.8 MB	3,113

Parent folder

BEYONDV  
Analyze  
from th  
Report a problem with ad content

①ダウンロードから解凍までの一連の手順。  
②はtar.bz2解凍のお約束の呪文。コピペ

# Boost

## • Boost (スライド 147)

講習会ではboost\_1\_61\_0.tar.bz2(約85MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/boost/files/boost/1.61.0/boost\\_1\\_61\\_0.tar.bz2/download](https://sourceforge.net/projects/boost/files/boost/1.61.0/boost_1_61_0.tar.bz2/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/boost/files/boost/1.61.0/boost\\_1\\_61\\_0.tar.bz2](https://sourceforge.net/projects/boost/files/boost/1.61.0/boost_1_61_0.tar.bz2)」に変更した。

```
cd ~/Downloads
# wget -c https://sourceforge.net/projects/boost/files/boost/1.61.0/boost_1_
pwd
ls -l boo*
bzip2 -dc boost_1_61_0.tar.bz2 | tar xvf -
cd boost_1_61_0
ls
./bootstrap.sh
```

```
pwd
ls -ld b*
cd boost
pwd
ls -ld include lib

### ./b2 install ###
cd ..
pwd
./b2 install --prefix=/home/iu/Downloads/boost_1_61_0/boost

### includeとlibの確認 ###
< >
```

# Boost

## • Boost (スライド147)

講習会ではboost\_1\_61\_0.tar.bz2(約85MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/boost/files/boost/1.61.0/boost\\_1\\_61\\_0.tar.bz2/download](https://sourceforge.net/projects/boost/files/boost/1.61.0/boost_1_61_0.tar.bz2/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/boost/files/boost/1.61.0/boost\\_1\\_61\\_0.tar.bz2](https://sourceforge.net/projects/boost/files/boost/1.61.0/boost_1_61_0.tar.bz2)」に変更した。

```
cd ~/Downloads
# wget -c https://sourceforge.net/p/boost/boost/1.61.0/boost_1_61_0.tar.bz2
pwd
ls -l boo*
bzip2 -dc boost_1_61_0.tar.bz2 | tar -x
cd boost_1_61_0
ls
./bootstrap.sh

pwd
ls -ld b*

cd boost
pwd
ls -ld include lib

### ./b2 install ####
cd ..
pwd
./b2 install --prefix=/home/iu/Downloads

### includeとlibの確認 ####
```

The terminal window shows the following session:

```
boost_1_61_0/libs/iostreams/test/finite_state_filter_test.cpp
boost_1_61_0/libs/iostreams/test/seek_test.hpp
boost_1_61_0/libs/iostreams/test/read_input_seq_test.hpp
boost_1_61_0/libs/iostreams/test/stream_state_test.cpp
boost_1_61_0/libs/iostreams/test/mapped_file_test.cpp
boost_1_61_0/libs/iostreams/test/filtering_stream_test.cpp
boost_1_61_0/libs/iostreams/test/write_bidir_test.hpp
boost_1_61_0/libs/iostreams/test/read_input_filter_test.hpp
boost_1_61_0/libs/iostreams/test/write_output_seq_test.hpp
boost_1_61_0/libs/iostreams/test/seekable_file_test.cpp
```

File listing in the terminal:

```
iu@bielinux[Downloads] cd boost_1_61_0 [ 8:32 午後]
iu@bielinux[boost_1_61_0] ls [ 8:35 午後]
boost bootstrap.bat INSTALL rst.css
boost-build.jam bootstrap.sh Jamroot status
boostcpp.jam doc libs tools
boost.css index.htm LICENSE_1_0.txt
boost.png index.html more
iu@bielinux[boost_1_61_0] [ 8:35 午後]
```

Red boxes highlight the Boost source files in the first list and the build directory contents in the second list. Red circles with the number '1' point to the 'boost' file in both lists.

# less README

Bridgerの「less README」。①今はこのあたりに対応。②次は「./bootstrap.sh」と打てばいいようだ。スライドを眺めるだけ

```
File Edit View Search Terminal Help Ja 19:56  
-----  
Installation  
-----  
1. Installing Boost  
  
a) download latest boost and unpack it.  
  
$ tar zxvf boost_1_47_0.tar.gz  
  
b) change to the boost directory and run ./bootstrap.sh.  
$ cd boost_1_47_0  
$ ./bootstrap.sh ① ②  
  
c) Run ./bz and set the install path.  
  
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
```

# ./bootstrap.sh

①確かにbootstrap.shは存在する。②言われるがままにコマンドを実行。約20秒

```

=====
Installation
=====

1. Installing Boost
   a) download latest
      $ tar zxvf boost_1_61_0.tar.gz
   b) change to the directory
      $ cd boost_1_61_0
      $ ./bootstrap.sh
   c) Run ./bz and start building
      $ ./b2 install

```

p  
boost\_1\_61\_0/libs/iostreams/test/seek\_test.hpp  
boost\_1\_61\_0/libs/iostreams/test/read\_input\_seq.hpp  
boost\_1\_61\_0/libs/iostreams/test/stream\_state.hpp  
boost\_1\_61\_0/libs/iostreams/test/mapped\_file.hpp  
boost\_1\_61\_0/libs/iostreams/test/filtering\_stream.hpp  
boost\_1\_61\_0/libs/iostreams/test/write\_bidir.hpp  
boost\_1\_61\_0/libs/iostreams/test/read\_input\_filter.hpp  
boost\_1\_61\_0/libs/iostreams/test/write\_output\_seq.hpp  
boost\_1\_61\_0/libs/iostreams/test/seekable\_file.hpp  
iu@bielinux[Downloads] cd boost\_1\_61\_0 [ 8:32午後]  
iu@bielinux[boost\_1\_61\_0] ls [ 8:35午後]  
boost bootstrap.bat INSTALL  
boost-build.jam bootstrap.sh Jamroot  
boostcpp.jam doc  
boost.css index.htm lib LICENSE\_1\_0.txt  
boost.png index.html more  
iu@bielinux[boost\_1\_61\_0] ./bootstrap.sh [ 8:35午後]

# ./bootstrap.sh 実行後

./bootstrap.sh 終了後の状態。赤枠内、特に①に着目。②Bridgerの「less README」の./b2と同じなので安心する

```

=====
Installation
=====

1. Installing Boost
   a) download latest
      $ tar zxvf boost_1_61_0.tar.gz
      $ cd boost_1_61_0
      $ ./bootstrap.sh
→ c) Run ./b2 and start building
      $ ./b2 install
      $ ./b2 --status

Bootstrapping is done. To build, run:
  ./b2 ①

To adjust configuration, edit 'project-config.jam'.
Further information:
  - Command line help:
    ./b2 --help

  - Getting started guide:
    http://www.boost.org/more/getting_started/unix-variants.html

  - Boost.Build documentation:
    http://www.boost.org/build/doc/html/index.html

```

iu@bielinux[boost\_1\_61\_0] [ 8:55 午後 ]

# 統：less README

① Bridgerの「less README」の./b2部分を眺める。②の赤下線部分を自分で指定しないといけないらしい。例えば③のように指定するらしい

```
File Edit View Search Terminal Help
→ c) Run ./b2 and set the install path.

$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
 ①
> For example, if you want install boost in /home/czheng
/local/boost, the command is :
$ ./b2 install --prefix=/home/czheng/local/boost
 ②
If the boost is installed successfully, you would find two sub-directories in /home/czheng/local/boost/:
/home/czheng/local/boost/include/
/home/czheng/local/boost/lib/
 ③

Note: The default Boost installation directory is /usr/local. Take note of the boost installation directory, because you need to tell the Bridger installer where to find boost later on.
:
```

# 自分に置き換える

①czhengは、②ユーザ名iuに対応する。③のboostディレクトリが、④に対応するのかはよくわからないので、もうちょっとマニュアルを見る

→ c) Run ./b2 and set the install path.

```
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
```

For example, if you want install boost in /home/czheng  
/local/boost, the command is :

```
$ ./b2 install --prefix=/home/czheng/local/boost
```

If the boost is installed in two sub-directories:  
/home/czheng/local/boost  
/home/czheng/local/boost

Note: The default directory is /home/czheng/local. Take note of the directory, because it's easier where to find boost.

The terminal window shows the following session:

```
iu@bielinux[boost_1_61_0] pwd  
/home/iu/Downloads/boost_1_61_0  
iu@bielinux[boost_1_61_0] ls -ld b*  
-rwxrwxr-x 1 iu iu 271416 6月 28 20:55 b2  
-rwxrwxr-x 1 iu iu 271416 6月 28 20:55 bjam  
drwxrwxr-x 110 iu iu 12288 5月 6 06:48 boost  
-rw-rw-r-- 1 iu iu 850 5月 6 06:10 boost-build.jam  
-rw-rw-r-- 1 iu iu 21920 5月 6 06:10 boostcpp.jam  
-rw-rw-r-- 1 iu iu 989 5月 6 06:10 boost.css  
-rw-rw-r-- 1 iu iu 6308 5月 6 06:10 boost.png  
-rw-rw-r-- 1 iu iu 2477 5月 6 06:10 bootstrap.bat  
-rw-rw-r-- 1 iu iu 918 6月 28 20:55 bootstrap.log  
-rwxrwxr-x 1 iu iu 10631 5月 6 06:10 bootstrap.sh
```

Annotations with red arrows:

- ① Points to the word "boost" in the command line \$ ./b2 install --prefix=/home/czheng/local/boost.
- ② Points to the user icon in the terminal window title bar.
- ③ Points to the word "boost" in the output line drwxrwxr-x 110 iu iu 12288 5月 6 06:48 boost.
- ④ Points to the word "bootstrap.sh" in the output line -rwxrwxr-x 1 iu iu 10631 5月 6 06:10 bootstrap.sh.

# 自分に置き換える

File Edit View Search Terminal Help

→ c) Run ./b2 and set the install path.

```
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
```

For example, if you want install boost in /home/czheng  
/local/boost, the command is :

```
$ ./b2 install --prefix=/home/czheng/local/boost
```

If the boost is installed successfully, you would find two sub-directories in /home/czheng/local/boost/:

/home/czheng/local/boost/include/  
/home/czheng/local/boost/lib/

Note: The default Boost installation directory is /usr  
/local. Take note of the boost installation  
directory, because you need to tell the Bridger installer  
where to find boost later on.

①find?!. どっちにしろ②の「./b2 install …」が成功したらincludeとlibというサブディレクトリができるよ、と書いてるので、③のboostディレクトリ内にincludeとlibディレクトリがなければ、ここを指定してもいいだろうと判断した

[ 1:36 午後 ]

[ 1:36 午後 ]

[ 1:36 午後 ]

[ 1:36 午後 ]

8 20:55 b2  
8 20:55 bjam  
6 06:48 boost  
6 06:10 boost-build.jam  
6 06:10 boostcpp.jam  
6 06:10 boost.css  
6 06:10 boost.png  
6 06:10 bootstrap.bat  
6 20:55 bootstrap.log  
6 06:10 bootstrap.sh

[ 1:37 午後 ]



```
-rw-rw-r-- 1 iu iu 2477 5月 28  
-rw-rw-r-- 1 iu iu 918 6月 28  
-rwxrwxr-x 1 iu iu 10631 5月 28  
iu@bielinux[boost_1_61_0]
```

①boostディレクトリに移動したので、②今はココ。③includeとlibというファイルもディレクトリもないことを確認したので、②のフルパスを④のところに書くことに決めた

# 自分に置き換える

File Edit View Search Terminal Help

→ c) Run ./b2 and set the install path.

```
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
```

For example, if you want install boost in /home/czheng  
/local/boost, the command is :

```
$ ./b2 install --prefix=/home/czheng/local/boost
```

If the boost is installed in two sub-directories:  
/home/czheng/local/boost  
/home/czheng/local/boost/include  
/home/czheng/local/boost/lib

Note: The default prefix is /home/czheng/local. Take note of the directory, because it's easier where to find boost headers and libraries.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has a title bar with the user's name and session details. Red numbered arrows point to specific elements:

- ① Points to the icon in the dock.
- ② Points to the terminal window title bar.
- ③ Points to the docked application menu icon.
- ④ Points to the terminal window title bar, specifically the part where the user has typed the command.

```
iu@bielinux[boost_1_61_0] cd boost
iu@bielinux[boost] pwd
/home/iu/Downloads/boost_1_61_0/boost
iu@bielinux[boost] ls -ld include lib
ls: cannot access include: No such file or directory
ls: cannot access lib: No such file or directory
iu@bielinux[boost]
```

# ./b2 install

①1つ上の階層に移動しているのは、②「./b2 install …」を③の場所で行うから。②を実行。約15分

```
File Edit View Search Terminal Help ↑ Ja 09:50  
x - ⊞ File Edit View Search Terminal Help ↑ Ja 09:50  
Ubuntu Dash Terminal  
→ c) Run ./b2 and set the install path.  
  
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>  
  
For example, if you want install boost in /home/czheng  
/local/boost, the commnd is :  
$ ./b2 install --prefix=/home/czheng/local/boost
```

```
File Edit View Search Terminal Help ↑ Ja 14:08  
d two sub-directories  
/home/czheng/l  
/home/czheng/l  
  
If the boost i  
iu@bielinux[boost] cd ..  
iu@bielinux[boost_1_61_0] pwd  
/home/iu/Downloads/boost_1_61_0  
iu@bielinux[boost_1_61_0] ./b2 install --prefix=/home/iu/Dow  
nloads/boost_1_61_0/boost  
[ 2:07午後 ]  
[ 2:08午後 ]
```

Note: The default  
/local. Take note of  
directory, because  
aller where to find b  
:

(時間はかかるが)特にエラーメッセージが出ることもなく終了

# ./b2 install 実行後

```
File Edit View Search Terminal Help
→ c) Run ./b2 and set the install path.
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>
For example, if you want install boost in /home/czheng
/local/boost, the command is :
$ ./b2 install --prefix=/home/czheng/local/boost
```

```
If the boost is installed in /home/czheng/local. Take note of the directory, because we will use it later where to find boost headers and libraries.
```

```
File Edit View Search Terminal Help Ja 14:57
```

```
gcc.compile.c++ bin.v2/libs/test/build/gcc-4.8/release/link-static/threading-multi/xml_report_formatter.o  
gcc.archive bin.v2/libs/test/build/gcc-4.8/release/link-static/threading-multi/libboost_unit_test_framework.a  
common.copy /home/iu/Downloads/boost_1_61_0/boost/lib/libboost_unit_test_framework.a  
gcc.compile.c++ bin.v2/libs/test/build/gcc-4.8/release/link-static/threading-multi/test_main.o  
gcc.archive bin.v2/libs/test/build/gcc-4.8/release/link-static/threading-multi/libboost_test_exec_monitor.a  
common.copy /home/iu/Downloads/boost_1_61_0/boost/lib/libboost_test_exec_monitor.a  
...updated 14283 targets...  
iu@bielinux[boost_1_61_0] [ 2:57午後]
```

①boostディレクトリ中に、確かにincludeとlibができる

# includeとlibの確認

```
File Edit View Terminal Help  
→ c) Run ./b2 and set the install path.  
  
$ ./b2 install --prefix=<YOUR_BOOST_INSTALL_DIRECTORY>  
  
For example, if you want install boost in /home/czheng  
/local/boost, the command is :  
$ ./b2 install --prefix=/home/czheng/local/boost
```

```
File Edit View Terminal Help  
d two sub-directories  
/home/czheng/local/boost  
/home/czheng/local/boost/include  
/home/czheng/local/boost/lib  
  
iu@bielinux[boost_1_61_0] pwd [ 3:03 午後]  
/home/iu/Downloads/boost_1_61_0  
iu@bielinux[boost_1_61_0] ls -ld boost/include boost/lib  
drwxrwxr-x 3 iu iu 4096 6月 29 14:33 boost/include  
drwxrwxr-x 2 iu iu 4096 6月 29 14:52 boost/lib  
iu@bielinux[boost_1_61_0] [ 3:03 午後]
```



# 統 : less README

File Edit View Search Terminal Help

→ d) Set the LD\_LIBRARY\_PATH env

The `~/.bash_profile` (`$HOME/.bash_profile`) or `~/.profile` file is executed when you login using console or remotely using ssh.

Append the following command to `~/.bash_profile` or `~/.profile` file:

```
$ export LD_LIBRARY_PATH=/home/czheng/local/boost/lib  
:$LD_LIBRARY_PATH
```

Save and close the file.

OR

just type the command:

```
$ export LD_LIBRARY_PATH=/home/czheng/local/boost/lib  
:$LD_LIBRARY_PATH
```

:

① Bridgerの「less README」で次の項目に移行。  
LD\_LIBRARY\_PATHという環境変数(environment variable)をセットする。手段は2つ。②1つは環境設定ファイル(`~/.bash_profile` or `~/.profile`)に書き込むやり方。③そしてもう1つは、そのターミナル上でのみ有効な指定方法



# 統 : less README

File Edit View Search Terminal Help

→ d) Set the LD\_LIBRARY\_PATH environment variable:

The `~/.bash_profile` (`$HOME/.bash_profile`) or `~/.profile` file is executed when you login using console or remotely using ssh.

Append the following command to `~/.bash_profile` or `~/.profile` file:

```
$ export LD_LIBRARY_PATH=/home/czheng/local/boost/lib  
:$LD_LIBRARY_PATH
```

Save and close the file.

OR

just type the command:

```
$ export LD_LIBRARY_PATH=/home/czheng/local/boost/lib  
:$LD_LIBRARY_PATH  
:
```

ここでは①を採用する。Bio-Linuxのデフォルトはzshなので、`.zshrc`に書き込むやり方を示す(第4回W10-3; 第5回W17-2)。②に対応するのは`/home/iu/Downloads/boost_1_61_0/boost`

①

②

# LD\_LIBRARY\_PATH

第5回W17-2のやり方を踏襲して、①echoで必要な情報を~/.zshrcに書き込む。②書き込み前と③書き込み後の最後の5行分を表示して確認

```

File Edit View Search Terminal Help
→ d) Set the LD_LIBRARY_PATH environment variable:
The ~/.bash_profile ($HOME/.bash_profile) or ~/.profile file is executed when you login using console or remotely using ssh.
Append the following line to your
./profile file:
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH
Save and close the file.
OR
just type the command in terminal:
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH
:

```

```

iu@bielinu[boost_1_61_0] tail -n 5 ~/.zshrc [4:06午後]
# fi
export PATH=$PATH:/home/iu/Downloads/FastQC:/home/iu/bin
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
iu@bielinu[boost_1_61_0] echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH' >> ~/.zshrc
iu@bielinu[boost_1_61_0] tail -n 5 ~/.zshrc [4:06午後]
export PATH=$PATH:/home/iu/Downloads/FastQC:/home/iu/bin
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH
iu@bielinu[boost_1_61_0] [4:06午後]

```

# LD\_LIBRARY\_PATH

File Edit View Search Terminal Help

→ d) Set the LD\_LIBRARY\_PATH environment variable:

The ~/.bash\_profile (\$HOME/.bash\_profile) or ~/.profile file is executed when you login using console or remotely using ssh.

Append the following line to your ~/.profile file:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH
```

Save and close the terminal window.

OR

just type the command:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH
```

:



①

```
iu@bielinux[boost_1_61_0] echo $LD_LIBRARY_PATH [ 4:14 午後 ]
iu@bielinux[boost_1_61_0] source ~/.zshrc [ 4:14 午後 ]
iu@bielinux[boost_1_61_0] echo $LD_LIBRARY_PATH [ 4:14 午後 ]
/home/iu/Downloads/boost_1_61_0/boost/lib:
iu@bielinux[boost_1_61_0] [ 4:14 午後 ]
```

# 続 : less README

①

2. Building Bridger [Make sure Boost has been successfully]

a) Unpack the Bridger and change to the Bridger directory.

```
$ tar zxvf Bridger_r2013-06-02.tar.gz  
$ cd Bridger_r2013-06-02
```

b) Configure Bridger. If Boost is installed somewhere other than /usr/local, you will need to tell the installer where to find it using --with-boost option.

```
$ ./configure --with-boost=/home/czheng/local/boost/  
Note: please replace "/home/czheng/local/boost/" with  
your own directory "<YOUR_BOOST_INSTALL_DIRECTORY>"
```

②

Bridgerの「less README」で次の項目(①2. Building Bridger)に移行。②の部分をみればわかるが、less READMEを開いているターミナルと、インストールを実行しているターミナルの2つを切り替えながらやっています

# 統: less README

File Edit View Search Terminal Help

2. Building Bridger [Make sure Boost has been installed successfully]

a) Unpack the Bridger and change to the Bridger directory.

```
$ tar zxvf Bridger_r2013-06-02.tar.gz  
$ cd Bridger_r2013-06-02
```

b) Configure Bridger. If Boost is installed somewhere other than /usr/local, you will need to tell the installer where to find it using --with-boost option.

```
$ ./configure --with-boost=/home/czheng/local/boost/  
Note: please replace "/home/czheng/local/boost/" with  
your own directory "<YOUR_BOOST_INSTALL_DIRECTORY>"
```

①の部分は、「もうやってるんですけど…」とは突っ込まない。②若干古いバージョンになっているが、このあたりはイチイチ突っ込まないのもお約束。③のConfigure Bridgerに移行



①Bridgerのディレクトリに移動して、②lsでconfigureがあることを確認し、③--with-boostオプションつきで実行。約1分

# configure

File Edit View Search Terminal Help

2. Building Bridger [Make sure Boost has been installed successfully]

a) Unpack the Bridger and change to the Bridger directory.

```
$ tar zxvf Bridger_r2014-12-01.tar.gz
$ cd Bridger_r2014-12-01
```

b) Configure Bridger than /usr/local, the installer ion.

```
$ ./configure
Note: please run your own directory
```

iu@bielinux[boost\_1\_61\_0] cd ~/Downloads/Bridger\_r2014-12-01

iu@bielinux[Bridger\_r2014-12-01] pwd  
/home/iu/Downloads/Bridger\_r2014-12-01

iu@bielinux[Bridger\_r2014-12-01] ls

aclocal.m4	ChangeLog	COPYING
AUTHORS	config.h	INSTALL
autom4te.cache	config.h.in	libtool
ax_boost_base.m4	config.log	LICENCE
boost.m4	config.status	m4
Bridger.pl	configure	Makefile
build-aux	configure.ac	Makefile.am
		stamp-h1

iu@bielinux[Bridger\_r2014-12-01] ./configure --with-boost=/home/iu/Downloads/boost\_1\_61\_0/boost

# configure実行後

```
File Edit View Search Terminal Help Ja 16:17
2. Building Bridger [Make sure Boost has been installed successfully]

a) Unpack the Bridger and change to the Bridger directory.
y.

$ tar zxvf Bridger_r2013-03-21.tar.gz
$ cd Bridger_r2013-03-21

b) Configure Bridger to install to /usr/local instead of /usr/local,
   the installer will complain about missing files.
   the installer will complain about missing files.

$ ./configure
Note: please run make to build Bridger
your own directory

See config.h for further configuration information.
Email <changzmaths@gmail.com> with questions and bug reports.

iu@bielinux[Bridger_r2014-12-01] [ 5:06午後]
```

# 続: less README

① Edit View Search Terminal Help

c) Make Bridger.

\$ make

note: If you build boost suffessfully without using --prefix option, the following commands may need before "make":

```
export LIBS="-L/home/czheng/boost_1_47_0/stage/lib"
" (replace "/home/czheng/boost_1_47_0/" with your own directory)
export CPPFLAGS="-I/home/czheng/boost_1_47_0/"
```

3. Test the installation. Test data are provided with sofew are distribution in the sample\_test directory.

```
$ cd src
$ ./Assemble -h
you would see:
=====
```

Bridgerの「less README」で次の項目(①make)に移行。赤枠部分は、--prefixオプションをつけずにboostを実行してうまくいった場合の話らしい。今回は--prefixオプションをつけたので、無関係と判断

# make

c) Make Bridger.

\$ make

```
note: If you build Bridger with a different prefix option, the following environment variables may be useful:
      export LIBS="-L${prefix}/lib -lBridger"
      " (replace "/home/czhang/Downloads/Bridger" with your source directory)
      export CPPFLAGS="-I${prefix}/include -I${prefix}/include/boost_1_61_0/boost/include"
```

3. Test the installation by running ./Assemble. You would see:

```
$ cd src
$ ./Assemble -h
=====

```

config.status: config.h is unchanged  
 config.status: executing depfiles commands  
 config.status: executing libtool commands

-- Bridger r2013-03-21 Configuration Results --  
 C++ compiler: g++ -Wall -Wno-deprecated -g -gdwarf-2 -Wunused -Wuninitialized -fpermissive -m64 -O3 -g -O2 -I/home/czhang/Downloads/boost\_1\_61\_0/boost/include  
 GCC version: gcc (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4  
 Host System type: x86\_64-unknown-linux-gnu  
 Install prefix: /usr/local  
 Install eprefix: \${prefix}

See config.h for further configuration information.  
 Email <changzm@maths@gmail.com> with questions and bug reports.

iu@bielinux[Bridger\_r2014-12-01] make [ 5:17午後]

# make実行後

c) Make Bridger.

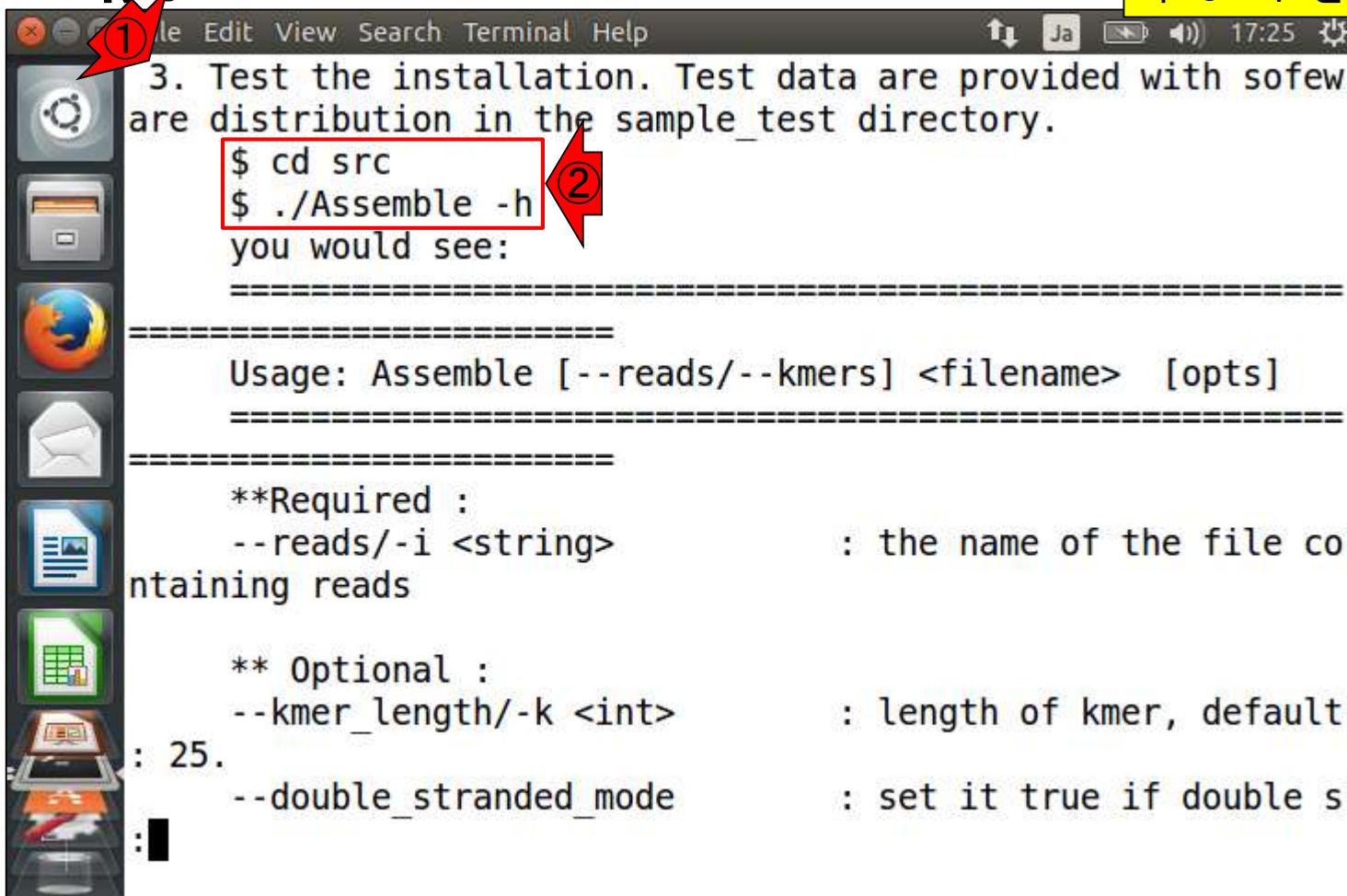
```
$ make

note: If you bu
prefix option, the fo
:
export LIBS
" (replace "/home/czh
ory)
export CPPR

iu@bielinux[~/Downloads/Bridger_r2014-12-01]
puts
^
g++ -O3 -Wall scanForPairedEndReads.cpp sam/libbam.a -lz -o
rsem-scan-for-paired-end-reads
In file included from scanForPairedEndReads.cpp:13:0:
utils.h:26:13: warning: 'verbose' defined but not used [-Wun
used-variable]
static bool verbose = true; // show detail intermediate out
puts
^
make[2]: Leaving directory `/home/iu/Downloads/Bridger_r2014
-12-01/plugins/rsem'
make[2]: Entering directory `/home/iu/Downloads/Bridger_r201
4-12-01'
make[2]: Leaving directory `/home/iu/Downloads/Bridger_r2014
-12-01'
make[1]: Leaving directory `/home/iu/Downloads/Bridger_r2014
-12-01'
iu@bielinux[Bridger_r2014-12-01] [ 5:21午後]
```

# 統：less README

Bridgerの「less README」で次の項目  
①インストール確認 ; test)に移行。②基本はこれを打って確認するだけのようだ



```
3. Test the installation. Test data are provided with software distribution in the sample_test directory.  
$ cd src  
$ ./Assemble -h  
you would see:  
=====  
Usage: Assemble [--reads/--kmers] <filename> [opts]  
=====  
=====  
**Required :  
--reads/-i <string> : the name of the file containing reads  
  
** Optional :  
--kmer_length/-k <int> : length of kmer, default : 25.  
--double_stranded_mode : set it true if double s  
:
```

# ./Assemble -h

File Edit View Search Terminal Help 17:25

3. Test the installation. Test data are provided with software distribution in the sample\_test directory.

```
$ cd src
$ ./Assemble -h
you would see:
=====
Usage: Assemble
=====
**Required :
--reads/-i <string>
containing reads

** Optional :
--kmer_length/-k
: 25.
--double_stranded
```

File Edit View Search Terminal Help 17:38

iu@bielinux[Bridger\_r2014-12-01] clear [ 5:26午後]

iu@bielinux[Bridger\_r2014-12-01] pwd /home/iu/Downloads/Bridger\_r2014-12-01 [ 5:26午後]

iu@bielinux[Bridger\_r2014-12-01] ls aclocal.m4 ChangeLog COPYING AUTHORS config.h INSTALL autom4te.cache config.h.in libtool ax\_boost\_base.m4 config.log LICENCE boost.m4 config.status m4 Bridger.pl configure Makefile build-aux configure.ac Makefile.am stamp-h1

iu@bielinux[Bridger\_r2014-12-01] cd src [ 5:38午後]

iu@bielinux[src] ./Assemble -h [ 5:38午後]

# ./Assemble -h実行後

画面が一気に流れるが、確かにREADMEファイル中に①で書かれている通り、赤枠のようなUsage(利用法)が表示される



3. Test the installation. Test data are provided with software distribution in the sample\_test directory.

```
$ cd src  
$ ./Assemble -h  
you would see:
```

```
=====  
Usage: Assemble  
=====
```

```
**Required :  
--reads/-i <string>  
containing reads
```

```
** Optional :  
--kmer_length/-k  
: 25.  
--double_stranded
```

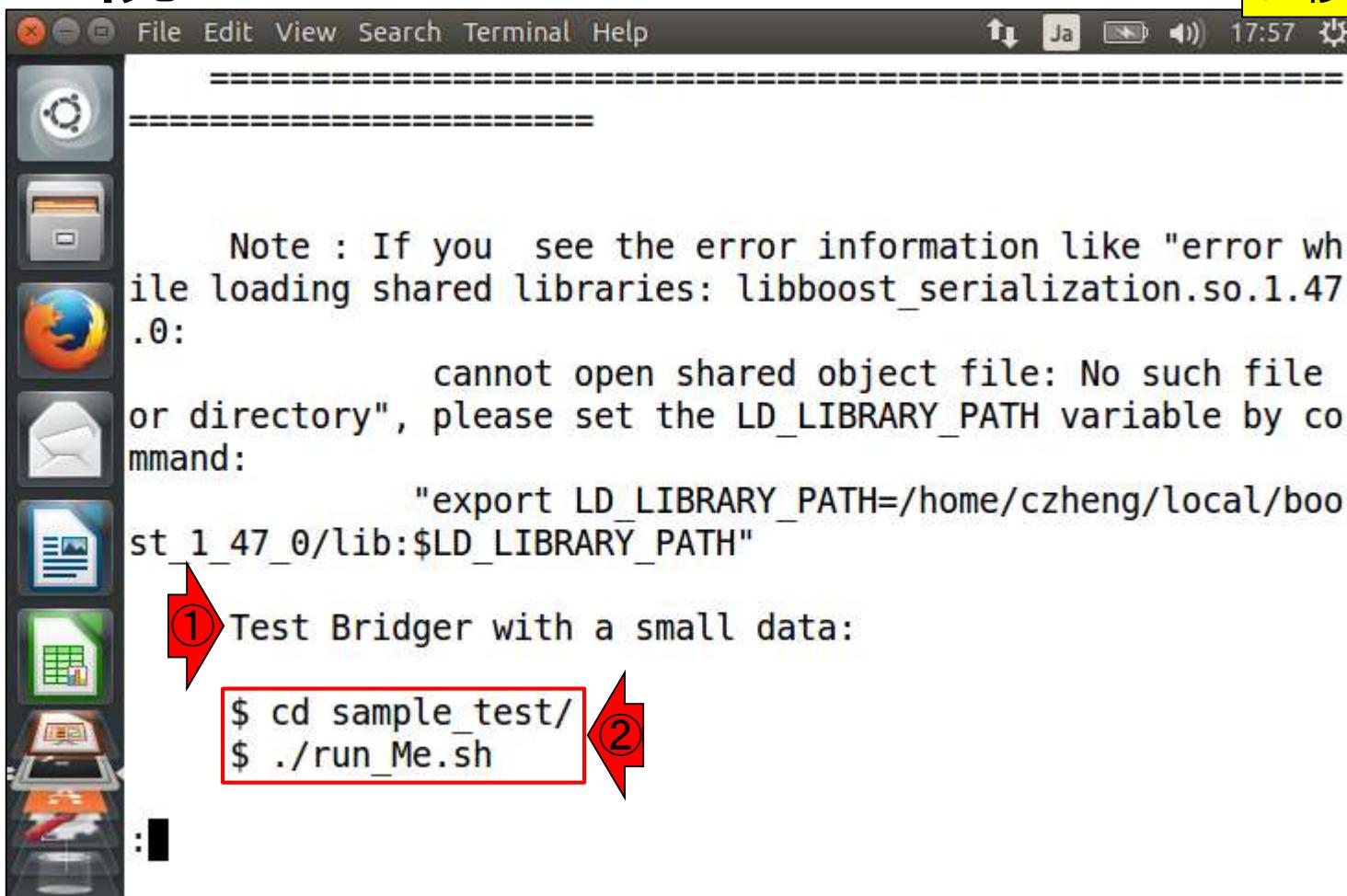
```
--min_kmer_entropy <float>      : minimum entropy of kmer used  
to extend, default: 0.0  
--min_junction_coverage <int> : minimum of the coverage of  
a junction, default: 2.  
--min_ratio_non_error <float> : min ratio for low/high alter-  
native extension that is  
not an error, default:  
t: 0.05.  
--pair_gap_length  
default: 200.  
--out_dir/-o <string>  
t, default : ./RawGraphs  
--help/-h  
n.  
=====
```

iu@bielinux[src] █

[ 5:41午後 ]

# 統 : less README

Bridgerの「less README」で次の項目(①small dataでBridger実行)に移行。②で確認するようだ



The screenshot shows a terminal window on an Ubuntu desktop. The terminal displays the contents of the Bridger README file. It includes notes about shared library loading errors and instructions for setting the LD\_LIBRARY\_PATH environment variable. Below this, there is a command to test with small data. A red box highlights the command line, and two red numbered arrows point to it from the bottom left.

```
=====
=====
Note : If you see the error information like "error while loading shared libraries: libboost_serialization.so.1.47.0:
cannot open shared object file: No such file or directory", please set the LD_LIBRARY_PATH variable by command:
"export LD_LIBRARY_PATH=/home/czheng/local/boost_1_47_0/lib:$LD_LIBRARY_PATH"
① Test Bridger with a small data:
$ cd sample_test/
$ ./run_Me.sh
②
```

# run\_Me.sh

① Bridgerディレクトリに戻り、② ls。③ 確かにsample\_testディレクトリが存在する

```

Note : If you see error message like "ld: warning: libboost_system.so.1.47.0: cannot open shared object file: No such file or directory", please run command:
"export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:$LD_LIBRARY_PATH"

Test Bridger with:
$ cd sample_test
$ ./run_Me.sh
iu@bielinux[~/Downloads/Bridger_r2014-12-01]
iu@bielinux[src] cd ~/Downloads/Bridger_r2014-12-01
iu@bielinux[Bridger_r2014-12-01] pwd
/home/iu/Downloads/Bridger_r2014-12-01
iu@bielinux[Bridger_r2014-12-01] ls
aclocal.m4           ChangeLog        COPYING
AUTHORS              config.h         INSTALL
autom4te.cache       config.h.in      libtool
ax_boost_base.m4     config.log       LICENCE
boost.m4              config.status    m4
Bridger.pl           configure       Makefile
build-aux            configure.ac    Makefile.am
iu@bielinux[Bridger_r2014-12-01]

```

The terminal window shows the execution of the script and its output. Red arrows point to specific elements:

- ① Points to the terminal icon in the dock.
- ② Points to the 'ls' command output, specifically the 'sample\_test' directory.
- ③ Points to the 'sample\_test' directory in the 'ls' output.

# run\_Me.sh

File Edit View Search Terminal Help

```

=====
Note : If you s
ile loading shared li
.0:
        cannot
or directory", please
mmand:
"export
st_1_47_0/lib:$LD_LIB
Test Bridger with
$ cd sample_test
$ ./run_Me.sh
=====
iu@bielinux[~/Downloads/Bridger_r2014-12-01/sample_test] 19:23 [ 7:22 午後 ]
iu@bielinux[Bridger_r2014-12-01] cd sample_test/
iu@bielinux[sample_test] ls [ 7:22 午後 ]
clean.pl      run_abundance_estimation.sh
reads.left.fq.gz  run_Me_assembly.sh
reads.right.fq.gz  run_Me.sh
iu@bielinux[sample_test] more run_Me.sh [ 7:22 午後 ]
#!/bin/bash -ve
### run Bridger ####
./Bridger.pl --seqType fq --left reads.left.fq.gz --right r
eads.right.fq.gz --SS_lib_type RF --CPU 2
#####
iu@bielinux[sample_test] [ 7:22 午後 ]

```

①sample\_testディレクトリに移動し、②ls。③確かにrun\_Me.shファイルが存在する。④中身を確認。⑤1行目の「#!/bin/bash -ve」がシェルスクリプトであることを明示する役割を果たしている(第4回W2-3)

# ./run\_Me.sh

①コマンドの実体はこれだけ。Trinityとオプション名も同じでわかりやすい。その都度打ち込めばいいじゃないかと思うかもしれないが、このようにしてファイルに残しておくのが一般的。②./run\_Me.shを実行

```

Note : If you see error message like "ld: warning: libBridger.so.1, version 0:
cannot open shared object file: No such file or directory", please run command:
"export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:$LD_LIBRARY_PATH"
Test Bridger with sample data

$ cd sample_test
$ ./run_Me.sh
=====
iu@bielinux[Bridger_r2014-12-01] cd sample_test/
iu@bielinux[sample_test] ls [ 7:22 午後]
clean.pl          run_abundance_estimation.sh
reads.left.fq.gz  run_Me_as_DS.sh
reads.right.fq.gz run_Me.sh
iu@bielinux[sample_test] more run_Me.sh [ 7:22 午後]
#!/bin/bash -ve

### run Bridger ###

1 ./Bridger.pl --seqType fq --left reads.left.fq.gz --right reads.right.fq.gz --SS_lib_type RF --CPU 2
=====
##### Done #####
2 iu@bielinux[sample_test] ./run_Me.sh [ 7:22 午後]

```

# Errorで終了



```
Note : If you see an error message like "ld: warning: libz.so.1, which
file loading shared library file: cannot open shared object file: No such
or directory", please run the following command:
"export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:$LD_LIBRARY_PATH"

Test Bridger with:
$ cd sample_test
$ ./run_Me.sh
```

=====

```
CMD: /home/iu/Downloads/Bridger_r2014-12-01/plugins/fastool/
fastool --rev --to-fasta /home/iu/Downloads/Bridger_r2014-12-
-01/sample_test/reads.left.fq.gz > left.fa
CMD finished (0 seconds)
CMD finished (0 seconds)

Done converting input files.

CMD: cat left.fa right.fa > both.fa
CMD finished (0 seconds)

### Splicing Graphs Reconstruction ###

CMD: /home/iu/Downloads/Bridger_r2014-12-01/src/Assemble --r
eads both.fa -k 25 --pair_end --fr_strand 1 2>Assemble.log
Error, cmd: /home/iu/Downloads/Bridger_r2014-12-01/src/Assem
ble --reads both.fa -k 25 --pair_end --fr_strand 1 2>Assem
ble.log died with ret 256 !
iu@bielinus[sample_test] [ 7:53午後]
```

①

# 原因の特定を試みる

①ls -ltで直近に作成されたものを確認。②bridger\_out\_dirというディレクトリができるているようなので…

```
Note : If you s File Edit View Search Terminal Help Ja 20:04
ile loading shared li .0:
cannot or directory", please mmand:
"export st_1_47_0/lib:$LD_LIB
Test Bridger with
$ cd sample_test
$ ./run_Me.sh
:
### Splicing Graphs Reconstruction ####
CMD: /home/iu/Downloads/Bridger_r2014-12-01/src/Assemble --reads both.fa -k 25 --pair_end --fr_strand 1 2>Assemble.log
Error, cmd: /home/iu/Downloads/Bridger_r2014-12-01/src/Assemble --reads both.fa -k 25 --pair_end --fr_strand 1 2>Assemble.log died with ret 256 !
iu@bielinux[sample_test] ls -lt | head [ 7:53午後]
total 428
drwxrwxr-x 3 iu iu 4096 6月 29 19:53 bridger_out_dir
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estimation.sh
-rwxr-xr-x 1 iu iu 800 6月 2 2013 clean.pl
-rwxr-xr-x 1 iu iu 151 6月 2 2013 run_Me_as_DS.sh
-rwxr-xr-x 1 iu iu 163 6月 2 2013 run_Me.sh
-rw----- 1 iu iu 208855 6月 2 2013 reads.right.fq.gz
-rw----- 1 iu iu 204913 6月 2 2013 reads.left.fq.gz
iu@bielinux[sample_test] [ 8:01午後]
```

# 原因の特定を試みる

Note : If you see an error message like "ld: warning: libz.so.1, which is needed by /home/iu/Downloads/Bridger\_r2014-12-01/src/Assemble --reads both.fa -k 25 --pair\_end --fr strand 1 2>Assemble.log died with ret 256 !", it means that the shared library "libz.so.1" is missing or corrupted. You can try to install it using the command:

```
"export LD_LIBRARY_PATH=$PWD/lib:$LD_LIBRARY_PATH"
Test Bridger with:
$ cd sample_test
$ ./run_Me.sh
```

File Edit View Search Terminal Help

Error, cmd: /home/iu/Downloads/Bridger\_r2014-12-01/src/Assemble --reads both.fa -k 25 --pair\_end --fr strand 1 2>Assemble.log died with ret 256 !

iu@bielinux[sample\_test] ls -lt | head [ 7:53 午後]

drwxrwxr-x	3 iu iu	4096	6月 29	19:53	bridger_out_dir			
-rwxr-xr-x	1 iu iu	497	6月 2	2013	run_abundance_estimation.sh			
-rwxr-xr-x	1 iu iu	800	6月 2	2013	clean.pl			
-rwxr-xr-x	1 iu iu	151	6月 2	2013	run_Me_as_DS.sh			
-rwxr-xr-x	1 iu iu	163	6月 2	2013	run_Me.sh			
-rw-----	1 iu iu	208855	6月 2	2013	reads.right.fq.gz			
-rw-----	1 iu iu	204913	6月 2	2013	reads.left.fq.gz			
iu@bielinux[sample_test] ls -l bridger_out_dir [ 8:01 午後]								
total	1164							
-rw-rw-r--	1 iu iu	47	6月 29	19:53	Assemble.log			
-rw-rw-r--	1 iu iu	1183240	6月 29	19:53	both.fa			
drwxrwxr-x	2 iu iu	4096	6月 29	19:53	RawGraphs			
iu@bielinux[sample_test]								
								[ 8:13 午後 ]

①ls -ltで直近に作成されたものを確認。②bridger\_out\_dirというディレクトリができているようなので、③中身を確認。④Assemble.logというログファイルが確かにある

# 原因の特定を試みる

⑤bridger\_out\_dirディレクトリにある  
Assemble.logの中身を確認。赤枠がエ  
ラーメッセージで、「RawGraphsという  
ディレクトリを作成できない」らしい…。

```
Note : If you see an error message like "ld: warning: libz.so.1, which  
file loading shared library file: cannot open shared object file: No such  
.0:  
or directory", please run command:  
"export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:$LD_LIBRARY_PATH"  
Test Bridger with:  
$ cd sample_test  
$ ./run_Me.sh
```

①

```
iu@bielinux[sample_test] ls -lt | head  
total 428  
drwxrwxr-x 3 iu iu 4096 6月 29 19:53 bridger_out_dir  
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estimation.sh  
-rwxr-xr-x 1 iu iu 800 6月 2 2013 clean.pl  
-rwxr-xr-x 1 iu iu 151 6月 2 2013 run_Me_as_DS.sh  
-rwxr-xr-x 1 iu iu 163 6月 2 2013 run_Me.sh  
-rw----- 1 iu iu 208855 6月 2 2013 reads.right.fq.gz  
-rw----- 1 iu iu 204913 6月 2 2013 reads.left.fq.gz
```

②

```
iu@bielinux[sample_test] ls -l bridger_out_dir [ 8:32午後]  
total 1164  
-rw-rw-r-- 1 iu iu 47 6月 29 19:53 Assemble.log  
-rw-rw-r-- 1 iu iu 1183240 6月 29 19:53 both.fa  
drwxrwxr-x 2 iu iu 4096 6月 29 19:53 RawGraphs
```

③

```
iu@bielinux[sample test] more bridger_out_dir/Assemble.log  
[Error] Cannot create directory ./RawGraphs/ !
```

④

```
iu@bielinux[sample test]
```

⑤

# 原因の特定を試みる

①ここにある…。できているのにできていないとエラーを吐いて終了しないでください…

```
File Edit View Search Terminal Help
=====
Note : If you see an error message like this:
file loading shared library ...
cannot open shared object file: No such file or directory", please run command:
"export LD_LIBRARY_PATH=$PWD/lib:$LD_LIBRARY_PATH"
Test Bridger with:
$ cd sample_test
$ ./run_Me.sh
iu@bielinux[sample_test] ls -lt | head
total 428
drwxrwxr-x 3 iu iu 4096 6月 29 19:53 bridger_out_dir
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estimation.sh
-rwxr-xr-x 1 iu iu 800 6月 2 2013 clean.pl
-rwxr-xr-x 1 iu iu 151 6月 2 2013 run_Me_as_DS.sh
-rwxr-xr-x 1 iu iu 163 6月 2 2013 run_Me.sh
-rw----- 1 iu iu 208855 6月 2 2013 reads.right.fq.gz
-rw----- 1 iu iu 204913 6月 2 2013 reads.left.fq.gz
iu@bielinux[sample_test] ls -l bridger_out_dir [ 8:32午後]
total 1164
-rw-rw-r-- 1 iu iu 47 6月 29 19:53 Assemble.log
-rw-rw-r-- 1 iu iu 1183240 6月 29 19:53 both.fa
drwxrwxr-x 2 iu iu 4096 6月 29 19:53 RawGraphs
iu@bielinux[sample test] more bridger_out_dir/Assemble.log
[Error] Cannot create directory ./RawGraphs/ !
iu@bielinux[sample_test] [ 8:32午後]
```



# Bridgerのまとめ

```

File Edit View Search Terminal Help
=====
=====
Note : If you see error message like "ld: warning: libz.so.1, version 1.4.7
      cannot be loaded because it contains an invalid symbol
      or directory", please run command:
      "export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:$LD_LIBRARY_PATH"
Test Bridger with command:
$ cd sample_test
$ ./run_Me.sh
:

```

iu@bielinux[sample\_test] ls -lt | head [ 8:32 午後]

```

total 428
drwxrwxr-x 3 iu iu 4096 6月 29 19:53 bridger_out_dir
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estimation.sh
-rwxr-xr-x 1 iu iu 800 6月 2 2013 clean.pl
-rwxr-xr-x 1 iu iu 151 6月 2 2013 run_Me_as_DS.sh
-rwxr-xr-x 1 iu iu 163 6月 2 2013 run_Me.sh
-rw----- 1 iu iu 208855 6月 2 2013 reads.right.fq.gz
-rw----- 1 iu iu 204913 6月 2 2013 reads.left.fq.gz
iu@bielinux[sample_test] ls -l bridger_out_dir [ 8:32 午後]
total 1164
-rw-rw-r-- 1 iu iu 47 6月 29 19:53 Assemble.log
-rw-rw-r-- 1 iu iu 1183240 6月 29 19:53 both.fa
drwxrwxr-x 2 iu iu 4096 6月 29 19:53 RawGraphs
iu@bielinux[sample_test] more bridger_out_dir/Assemble.log
[Error] Cannot create directory ./RawGraphs/ !
iu@bielinux[sample_test] [ 8:32 午後]

```

インストール自体は成功しても、サンプルデータの実行でコケルことはときどきあります。尚、Bridgerの後継プログラムであるBinPackerも、よくわからないエラーに遭遇して実行できませんでした。頑張ってエラーの解決を試みるヒト、諦めて別のプログラムの利用に切り替えるヒト、好きな道へどうぞ

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 発現量推定

## (Rで)塩基配列解析

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

### What's new?

- このウェブページ  
リソース Rと必要  
法(Windows2015  
年1月まで)

- 解析 | 基礎 | 平均-分散プロット | について (last modified 2015/11/11)
- 解析 | 基礎 | 平均-分散プロット | [Technical replicates](#) (last modified 2014/02/18)
- 解析 | 基礎 | 平均-分散プロット | [Biological replicates](#) (last modified 2014/02/21)
- 解析 | 新規転写物同定(ゲノム配列を利用) (last modified 2015/08/25)
- 解析 | 発現量推定(ランスクリプトーム配列を利用) (last modified 2016/06/30) 
- 解析 | クラスタリング | について (last modified 2016/05/19)
- 解析 | クラスタリング | サンプル間 | [hclust](#) (last modified 2015/02/26)
- 解析 | クラスタリング | サンプル間 | [TCC\(Sun 2013\)](#) (last modified 2015/11/15)
- 解析 | クラスタリング | 遺伝子間(基礎) | [MBCluster.Seq\(Si 2014\)](#) (last modified 2015/05/20)

ランスクリプトーム配列とRNA-seqデータを入力として、発現量を得る流れを最後に行います。大抵の場合、カウント情報も出力に含まれるので、カウントデータ取得目的としても利用可能。スライドを見るだけ

### 解析 | 発現量推定(ランスクリプトーム配列を利用) NEW

新規転写物(新規isoform)の発見などが目的でなく、既知転写物の発現量を知りたいだけの場合には、やたらと時間がかかるゲノム配列へのマッピングを避けるのが一般的です。有名なCufflinksも一応GTF形式のアノテーションファイルを与えることでゲノム全体にマップするモードがあるらしいので、一応リストアップしています。転写物へのマッピングの場合には、splice-aware alignerを用いたジャンクションリードのマッピングを行う必要がないので、高速にマッピング可能なbasic alignerで十分です。しかし、複数個所にマップされるリードは考慮する必要があり、確率モデルのパラメータを最尤法に基づいて推定するexpectation-maximization(EM)アルゴリズムがよく用いられます。マッピングを行わずに、k-merを用いてalignment-freeで行う発現量推定を行うSaifishやRNA-Skimは従来法に比べて劇的に高速化がなされているようです。間違いがいくつか含まれているとは思います。2016年6月に調べた結果をリストアップします：

#### R用:

- [AllelicImbalance](#): Gådin et al., BMC Bioinformatics, 2015
- [tximport](#): Soneson et al., F1000Res., 2015
- [RNAontheBENCH](#) (github上有): Germain et al., Nucleic Acids Res., 2016
- [SARTools](#) (github上有): Varela et al., PLoS One, 2016

#### R以外:

- [Cufflinks](#): Trapnell et al., Nat Biotechnol., 2010
- [NEUMA](#): Lee et al., Nucleic Acids Res., 2011
- [IsoEM](#): Nicolae et al., Algorithms Mol. Biol., 2011
- [RSEM](#): Li and Dewey, BMC Bioinformatics, 2011
- [eXpress](#): Roberts and Pachter, Nat Methods, 2013
- [ReXpress](#): Roberts et al., Bioinformatics, 2013
- [TIGAR](#): Narai et al., Bioinformatics, 2013
- [eXpress-D](#): Roberts et al., BMC Bioinformatics, 2013
- [PennSeq](#): Hu et al., Nucleic Acids Res., 2014
- [Saifish](#): Patro et al., Nat Biotechnol., 2014
- [Quinn's pipeline](#) (allele-specific): Quinn et al., Bioinformatics, 2014
- [RNA-Skim](#): Zhang and Wang, Bioinformatics, 2014
- [QuASAR](#) (allele-specific): Harvey et al., Bioinformatics, 2015
- [TIGER2](#): Narai et al., BMC Genomics, 2014
- [SUPPA](#): Alamancos et al., RNA, 2015
- folded Skellam mixture model(allele-specific): [Lu et al., BMC Genomics, 2015](#)
- [EMSAR](#): Lee et al., BMC Bioinformatics, 2015
- [PGSeq](#): Liu et al., PLoS One, 2015
- [NLDMseq](#): Liu et al., BMC Bioinformatics, 2015
- [ASE-TIGAR](#) (allele-specific): Narai et al., BMC Genomics, 2016
- [SplAdder](#): Kahles et al., Bioinformatics, 2016
- [RapMap](#): Srivastava et al., Bioinformatics, 2016

#### Review、ガイドライン、パイプライン系:

- 手法比較: Kanitz et al., Genome Biol., 2015
- 手法比較(transcript-based approach vs. 'union exon'-based approach): Zhao et al., PLoS One, 2015
- パイプライン(Ion Proton用): Yuan et al., BMC Genomics, 2016
- パイプライン(single-cell用): Ntranos et al., Genome Biol., 2016

# TIGAR2

①TIGAR2のインストールから利用の流れを説明します。うまくいかなかった場合でも日本語でやりとりできるのが何よりです。スライドを見るだけ

BMC Genomics. 2014;15 Suppl 10:S5. doi: 10.1186/1471-2164-15-S10-S5. Epub 2014 Dec 12.

## TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads.

Nariai N, Kojima K, Mimori T, Sato Y, Kawai Y, Yamauchi-Kabata Y, Nagasaki M.

### Abstract

**BACKGROUND:** High-throughput RNA sequencing (RNA-Seq) enables quantification and identification of transcripts at single-base resolution. Recently, longer sequence reads become available thanks to the development of new types of sequencing technologies as well as improvements in chemical reagents for the Next Generation Sequencers. Although several computational methods have been proposed for quantifying gene expression levels from RNA-Seq data, they are not sufficiently optimized for longer reads (e.g. >250 bp).

**RESULTS:** We propose TIGAR2, a statistical method for quantifying transcript isoforms from fixed and variable length RNA-Seq data. Our method models substitution, deletion, and insertion errors of sequencers based on gapped-alignments of reads to the reference cDNA sequences so that sensitive read-aligners such as Bowtie2 and BWA-MEM are effectively incorporated in our pipeline. Also, a heuristic algorithm is implemented in variational Bayesian inference for faster computation. We apply TIGAR2 to both simulation data and real data of human samples and evaluate performance of transcript quantification with TIGAR2 in comparison to existing methods.

**CONCLUSIONS:** TIGAR2 is a sensitive and accurate tool for quantifying transcript isoform abundances from RNA-Seq data. Our method performs better than existing methods for the fixed-length reads (100 bp, 250 bp, 500 bp, and 1000 bp of both single-end and paired-end) and variable-length reads, especially for reads longer than 250 bp.

### 解析 | 発現量推定(トランスクリプトーム配列を利用) NEW

転写物(新規isoform)の発見などが目的でなく、既知転写物の発現量を知りたいだけの場合には、多く時間がかかるゲノム配列へのマッピングを避けるのが一般的です。有名なCufflinksも一応GTFのアノテーションファイルを与えることでゲノム全体にマップするのを避けるモードがあるらしいので、リストアップしています。転写物へのマッピングの場合には、splice-aware alignerを用いたジャンクリードのマッピングを行う必要がないので、高速にマッピング可能なbasic alignerで十分です。但し数個所にマップされるリードは考慮する必要があり、確率モデルのパラメータを最尤法に基づいてするexpectation-maximization (EM)アルゴリズムがよく用いられます。マッピングを行わずに、k-merでalignment-freeで行う発現量推定を行うSaifishやRNA-Skimmerは従来法に比べて劇的に高速化しているようです。間違いがいくつか含まれているとは思います。2016年6月に調べた結果をリストします：

- AllelicImbalance: Gådin et al., BMC Bioinformatics, 2015
- tximport: Soneson et al., F1000Res., 2015
- RNAontheBENCH (github上有る): Germain et al., Nucleic Acids Res., 2016
- SARTools (github上有る): Varela et al., PLoS One, 2016

↓

- Cufflinks: Trapnell et al., Nat Biotechnol., 2010
- NEUMA: Lee et al., Nucleic Acids Res., 2011
- IsoEM: Nicolae et al., Algorithms Mol. Biol., 2011
- RSEM: Li and Dewey, BMC Bioinformatics, 2011
- eXpress: Roberts and Pachter, Nat Methods, 2013
- ReXpress: Roberts et al., Bioinformatics, 2013
- TIGAR: Nariai et al., Bioinformatics, 2013
- eXpress-D: Roberts et al., BMC Bioinformatics, 2013
- PennSeq: Hu et al., Nucleic Acids Res., 2014
- Saifish: Patro et al., Nat Biotechnol., 2014
- Quinn's pipeline(allele-specific): Quinn et al., Bioinformatics, 2014
- RNA-Skimmer: Zhang and Wang, Bioinformatics, 2015
- QuASAR(allele-specific): Harvey et al., Bioinformatics, 2015
- TIGER2: Nariai et al., BMC Genomics, 2014
- SUPPA: Alamancos et al., RNA, 2015
- folded Skellam mixture model(allele-specific): Liu et al., BMC Genomics, 2015
- EMSAR: Lee et al., BMC Bioinformatics, 2015
- PGSeq: Liu et al., PLoS One, 2015
- NLDMseq: Liu et al., BMC Bioinformatics, 2015
- ASE-TIGAR(allele-specific): Nariai et al., BMC Genomics, 2016
- SplAdder: Kahles et al., Bioinformatics, 2016
- RapMap: Srivastava et al., Bioinformatics, 2016

w. ガイドライン、バイブル系:

- 手法比較: Kanitz et al., Genome Biol., 2015
- 手法比較(transcript-based approach vs. 'union exon'-based approach): Zhao et al., PLoS One, 2015
- パーナー/アセンブリ用: Jan et al., BMC Genomics, 2016
- パーナー/アセンブリ用: Jan et al., BMC Genomics, 2016
- パーナー/アセンブリ用: Jan et al., Genome Biol., 2016

# TIGAR2

- [TIGAR2: Nariai et al., BMC Genomics, 2014](#)

① まずは master.zip (約6MB)を~/Downloadsにダウンロード済み。

```
cd ~/Downloads
# wget -c https://github.com/nariai/tigar2/archive/master.zip
pwd
ls -l mas*

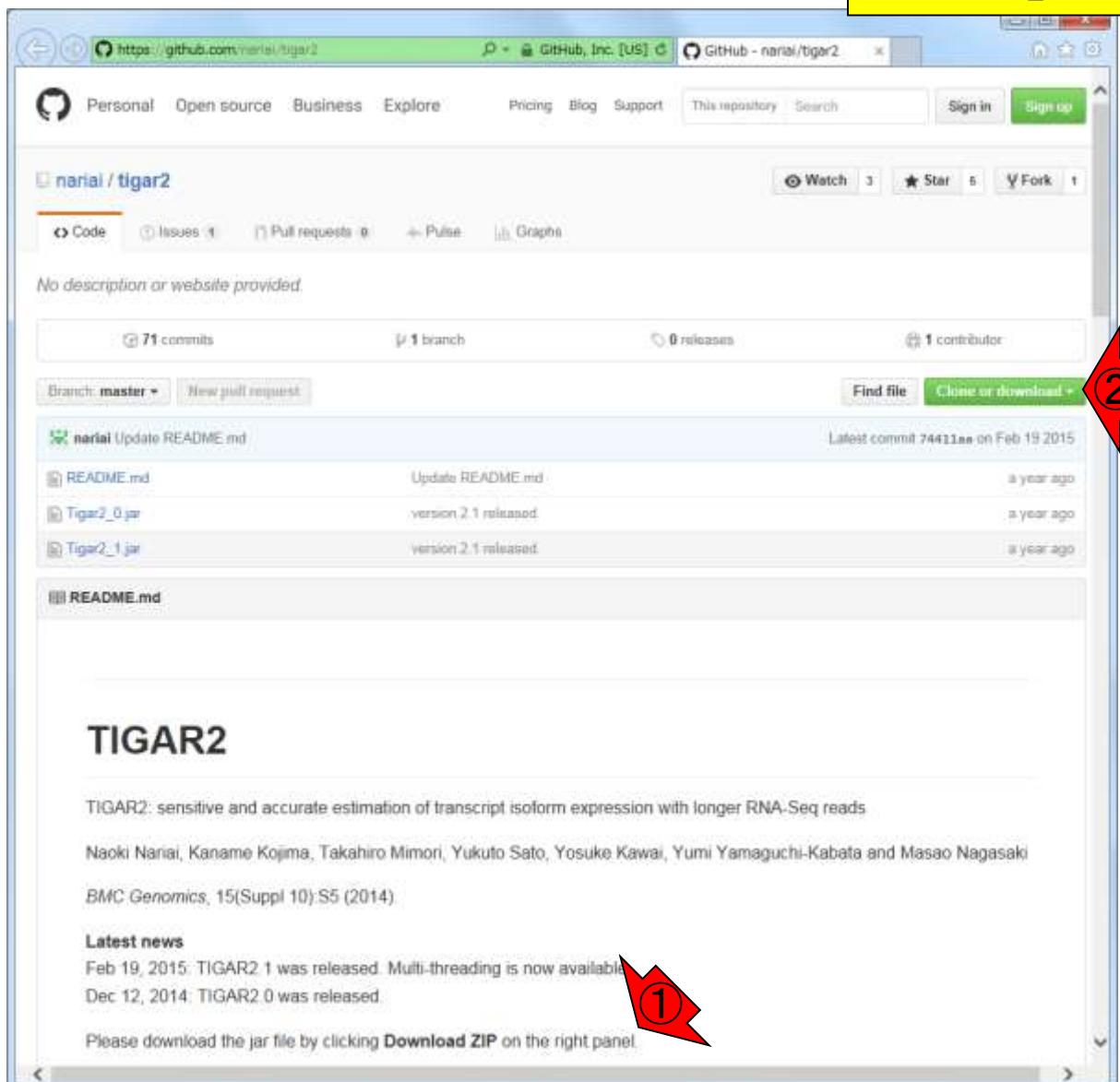
### 解凍 ###
unzip master.zip

### Githubと照合 ###
pwd
head tigar2-master/README.md

### 動作確認 ###
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar
```

# TIGAR2

①right panel上にある「Download Zip」というのをクリックしてjarファイルをダウンロードせよ、と書いてある。②のことかと思いつつ、クリックすると…



# TIGAR2

①right panel上にある「Download Zip」というのをクリックしてjarファイルをダウンロードせよ、と書いてある。②のことかと思いつつ、クリックすると…③確かに「Download Zip」を発見したので右クリックでこのURL情報を取得(<https://github.com/nariai/tigar2/archive/master.zip>)。スライドを見るだけ

①

②

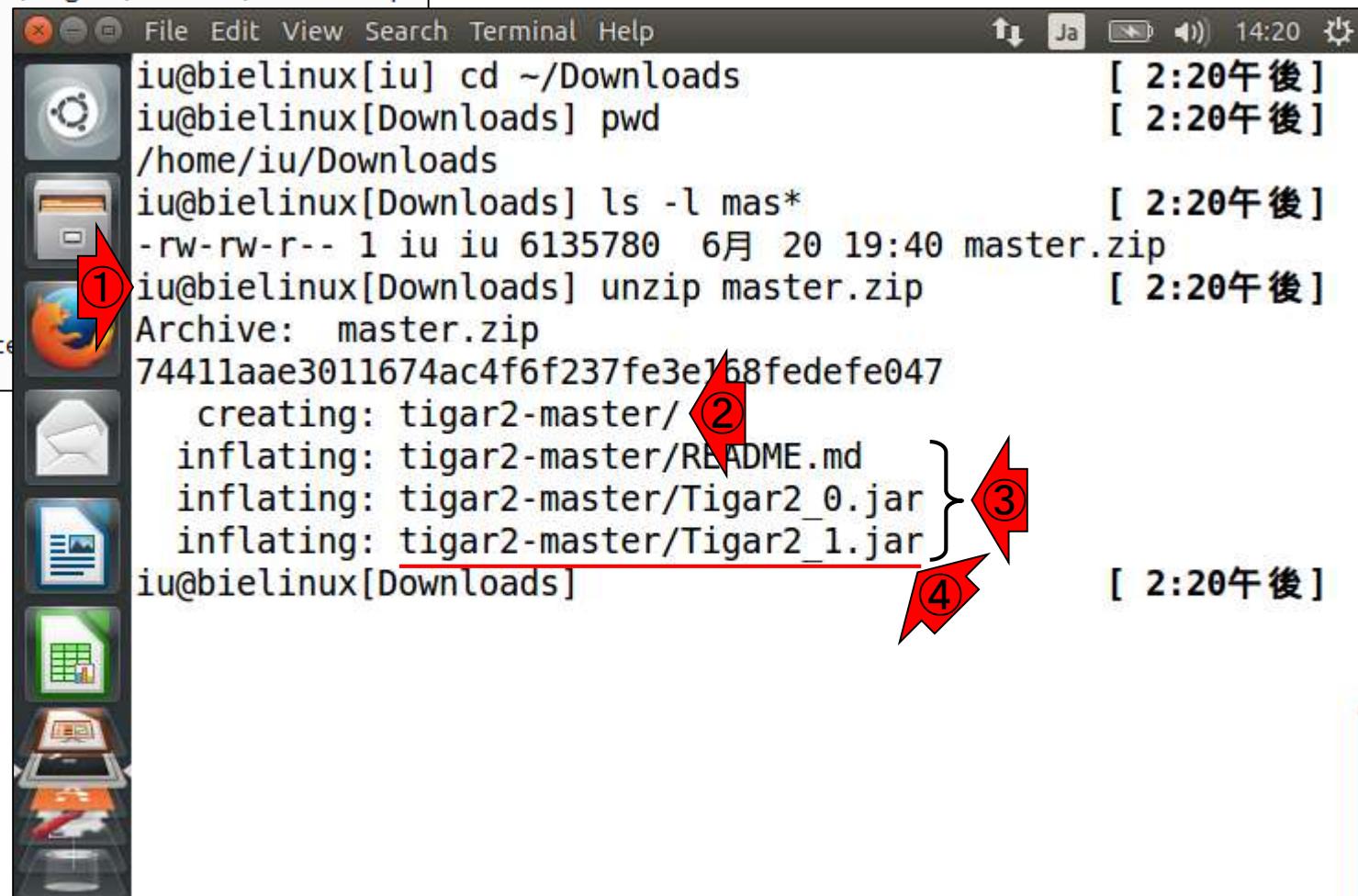
③

# 解凍

• [TIGAR2: Nariai et al., BMC Genomics, 2014](#)(スライド 190)

講習会ではmaster.zip(約6MB)を~/Downloadsにダウンロード済み。

```
cd ~/Downloads  
#wget -c https://github.com/nariai/tigar2/archive/master.zip  
pwd  
ls -l mas*  
  
### 解凍 ###  
unzip master.zip  
  
### Githubと照合 ###  
pwd  
head tigar2-master/README.md  
  
### 動作確認 ###  
java -jar ~/Downloads/tigar2-mas
```



The screenshot shows a terminal window with a dark theme. On the left, there is a vertical dock with icons for a terminal, file browser, and other applications. The terminal window has a title bar with the user's name and session details.

```
iu@bielinux[iu] cd ~/Downloads  
iu@bielinux[Downloads] pwd  
/home/iu/Downloads  
iu@bielinux[Downloads] ls -l mas*  
-rw-rw-r-- 1 iu iu 6135780 6月 20 19:40 master.zip  
iu@bielinux[Downloads] unzip master.zip  
Archive: master.zip  
74411aae3011674ac4f6f237fe3e168fedefe047  
    creating: tigar2-master/  
    inflating: tigar2-master/README.md  
    inflating: tigar2-master/Tigar2_0.jar  
    inflating: tigar2-master/Tigar2_1.jar
```

Red numbered arrows point to specific parts of the terminal output:

- ① Points to the command "unzip master.zip".
- ② Points to the line "creating: tigar2-master/".
- ③ Points to the three lines starting with "inflating": "tigar2-master/README.md", "Tigar2\_0.jar", and "Tigar2\_1.jar".
- ④ Points to the timestamp "[ 2:20午後]" at the end of the log.

講習会では、~/Downloadsにダウンロード済みなので、①unzipで解凍。②tigar2-masterというディレクトリが作成されて、③3つのファイルができたと解釈。④これが最新版だろう

# Githubと照合

The screenshot illustrates a comparison between a GitHub repository and a terminal session.

**GitHub Repository (Left):**

- URL: <https://github.com/nariai/tigar2>
- Repository Name: nariai / tigar2
- Code tab: 71 commits
- Issues tab: 1
- Pull requests tab: 0
- Branch: master
- Files listed:
  - README.md
  - Tigar2\_0.jar
  - Tigar2\_1.jar
  - README.md

A red arrow labeled ② points to the list of files.

**Terminal Session (Right):**

```

iu@bielinux[iu] cd ~/Downloads
iu@bielinux[Downloads] pwd
/home/iu/Downloads
iu@bielinux[Downloads] ls -l mas*
-rw-rw-r-- 1 iu iu 6135780 6月 20 19:40 master.zip
iu@bielinux[Downloads] unzip master.zip
Archive: master.zip
74411aae3011674ac4f6f237fe3e168fedefe047
  creating: tigar2-master/
  inflating: tigar2-master/README.md
  inflating: tigar2-master/Tigar2_0.jar
  inflating: tigar2-master/Tigar2_1.jar
iu@bielinux[Downloads]
  
```

A red arrow labeled ① points to the command "unzip master.zip".

Timestamps in the terminal session indicate the operations occurred at 2:20 PM:

- [ 2:20午後 ]

# Githubと照合

① README.mdの中身って、赤枠内の記述内容と同じではないか?! ② ココにもREADME.mdと書いてあるし…

narai / tigar2

No description or website provided.

71 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL  
https://github.com/narai/tigar2.git

Open in Desktop Download ZIP

① README.md

② README.md

## TIGAR2

TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads

Naoki Narai, Kaname Kojima, Takahiro Mimori, Yukuto Sato, Yosuke Kawai, Yumi Yamaguchi-Kabata and Masao Nagasaki

BMC Genomics, 15(Suppl 10):S5 (2014)

Latest news

Feb 19, 2015: TIGAR2 1 was released. Multi-threading is now available

Dec 12, 2014: TIGAR2 0 was released.

Please download the jar file by clicking Download ZIP on the right panel.

# Githubと照合

③headで最初の数行を表示して確認。赤枠部分を比較して予想通りと確信。こんな感じで、独特的な見栄えで最初はわかりづらい(個人の感想です)githubのノリに慣れていく

The screenshot illustrates a comparison between a GitHub repository page and a terminal window.

**GitHub Repository Page (Left):**

- URL: https://github.com/nariai/tigar2
- Repository Name: nariai / tigar2
- Code, Issues, Pull requests, Pulse tabs
- No description or website provided.
- 71 commits, 1 branch (master)
- Branch: master
- Files listed:
  - nariai Update README.md
  - README.md
  - Tigar2\_0.jar
  - Tigar2\_1.jar
  - README.md
- Latest news:
  - Feb 19, 2015: TIGAR2 1 was released. Multi-threading is now supported.
  - Dec 12, 2014: TIGAR2 0 was released.
- Please download the jar file by clicking Download ZIP on the right.

**Terminal Window (Right):**

- File Edit View Terminal Help menu
- Terminal title: iu@bielinux[Downloads] pwd
- Terminal content:

```
iu@bielinux[Downloads] pwd  
/home/iu/Downloads  
iu@bielinux[Downloads] head tigar2-master/README.md  
=====  
TIGAR2  
=====
```

TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads

Naoki Nariai, Kaname Kojima, Takahiro Mimori, Yukuto Sato, Yosuke Kawai, Yumi Yamaguchi-Kabata and Masao Nagasaki

\*BMC Genomics\*, 15(Suppl 10):S5 (2014).
- Terminal prompt: iu@bielinux[Downloads]
- Timestamp: [ 3:18午後 ]

Red annotations and arrows highlight specific parts of the GitHub page and terminal output:

- ① Points to the "TIGAR2" section on the GitHub page.
- ② Points to the "TIGAR2" section on the GitHub page.
- ③ Points to the "TIGAR2" section on the GitHub page.
- ④ Points to the terminal window's title bar.
- ⑤ Points to the terminal window's timestamp.

# README.md

<https://github.com/nariai/tigar2>

Github, Inc. [US]

Github - nariai/tigar2

## TIGAR2

TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads

Naoki Nariai, Kaname Kojima, Takahiro Miori, Yukuto Sato, Yosuke Kawai, Yumi Yamaguchi-Kabata and Masao Nagasaki

BMC Genomics. 15(Suppl 10):S5 (2014)

### Latest news

Feb 19, 2015: TIGAR2.1 was released. Multi-threading is now available.

Dec 12, 2014: TIGAR2.0 was released.

①

Use download the jar file by clicking **Download ZIP** on the right panel.

Usage: java -jar Tigar2\_1.jar FASTA SAM OUT

FASTA : reference FASTA file  
SAM : target SAM/BAM file  
OUT : output file

#### Options:

--thread\_num INT : number of thread  
--alpha\_zero DOUBLE : tuning parameter alpha\_zero.  
--is\_paired : paired-end data, default = FALSE.  
--frag\_dist\_mean DOUBLE: mean of the fragment length distribution. default = estimation from data  
--frag\_dist\_std DOUBLE: standard dev of the fragment length distribution. default = estimation from data

## Recommended pipeline to run TIGAR2

### 1. Prepare cDNA reference sequences in FASTA format.

e.g.) human  
<http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refRNA.fa.gz>

e.g.) mouse  
<http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refRNA.fa.gz>

### 2. Build bowtie2 index

Githubサイト上のREADME.mdを眺めながら、基本的な利用法(Usage)を学んでいく。①Usageの赤下線部に「java -jar Tigar2\_1.jar …」と書いている。これはTigar2\_1.jarのクラスパス(第5回W4-4; W17-2)を設定した後の話ではないかと昔の記憶をたどる

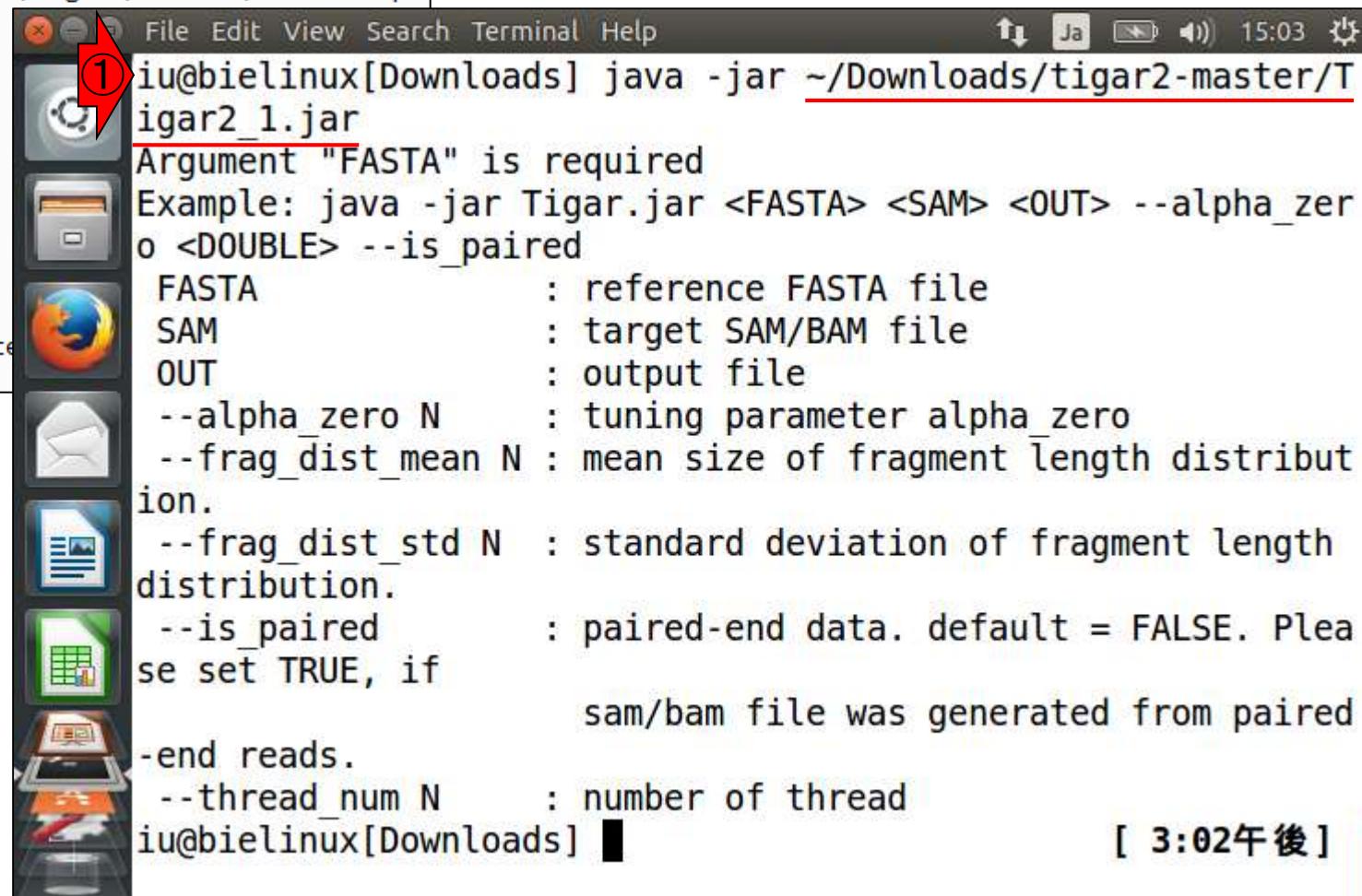
ここでは、①「java -jar Tigar2\_1.jarの相対パス指定」で動作確認。うまく動いているようだ

# 動作確認

• [TIGAR2: Nariai et al., BMC Genomics, 2014](#)(スライド 190)

講習会ではmaster.zip(約6MB)を~/Downloadsにダウンロード済み。

```
cd ~/Downloads
# wget -c https://github.com/nariai/tigar2/archive/master.zip
pwd
ls -l mas*
### 解凍 ####
unzip master.zip
### Githubと照合 ####
pwd
head tigar2-master/README.md
### 動作確認 ####
java -jar ~/Downloads/tigar2-maste
```



iu@bielinux[Downloads] java -jar ~/Downloads/tigar2-master/Tigar2\_1.jar

Argument "FASTA" is required  
 Example: java -jar Tigar.jar <FASTA> <SAM> <OUT> --alpha\_zero <DOUBLE> --is\_paired

FASTA	: reference FASTA file
SAM	: target SAM/BAM file
OUT	: output file
--alpha_zero N	: tuning parameter alpha_zero
--frag_dist_mean N	: mean size of fragment length distribution.
--frag_dist_std N	: standard deviation of fragment length distribution.
--is_paired	: paired-end data. default = FALSE. Please set TRUE, if
-end reads.	sam/bam file was generated from paired
--thread_num N	: number of thread

iu@bielinux[Downloads] [ 3:02午後 ]

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 推奨パイプライン1

①TIGAR2の推奨パイプラインに従って行う。②  
Step1は、トランскриプトーム配列の準備。ここではTrinity実行結果ファイル(Trinity1.fasta)を用いる

Recommended pipeline to run TIGAR2

1. Prepare cDNA reference sequences in FASTA format.

e.g.) human  
<http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMrna.fa.gz>

e.g.) mouse  
<http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMrna.fa.gz>

2. Build bowtie2 index

```
mkdir ref  
bowtie2-build refMrna.fa ./ref/refMrna
```

3. Run bowtie2

For single-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna sample.fastq > sample.sam
```

For paired-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna -1 sample_1.fastq -2 sample_2.fastq > sample.sam
```

4. Run TIGAR2

For single-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```

For paired-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format

# Step1

①実際の作業はこれら。②赤枠内をコピペ。発現量推定を行いたいトランскриプトーム配列ファイル(Trinity1.fasta)の場所に移動し、配列数や総塩基数のおさらいをしている。Trinity実行結果はヒトによって異なる。以降の解析結果が同じでないと不安なヒトは、wgetのコメントアウト(#)を外し、同じリファレンスを用いましょう

①

②

### 1. FASTA形式のリファレンス配列を準備 ###

```
cd ~/Documents/srp017156/20160804  
ls -la Trinity1.fasta  
#wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/  
grep -c ">" Trinity1.fasta  
grep -v ">" Trinity1.fasta | wc
```

### 2. マッピングプログラムbowtie2用のインデックスを作成 ###

```
bowtie2-build --version  
mkdir rof  
bowtie2-build Trinity1.fasta ./rof/trenitey  
ls -l rof
```

### 3. マッピングプログラムbowtie2を実行 ###

### トリム前のpaired-endでとりあえずやる ###

```
bowtie2 --version  
bowtie2 -h  
  
bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \  
-1 data1.fq.gz -2 data2.fq.gz > test.sam  
pwd  
ls -l test.sam
```

# Step1

- 推奨バイブルインでTIGAR2を実行 (スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 1. FASTA形式のリファレンス配列を準備 ###
```

```
cd ~/Documents/srp017156/20160804
```

```
ls -la Trinity1.fasta
```

```
#wget -c http://www.iu.a.u-tokyo.ac.jp/~tigardata/Trinity1.fasta
```

```
grep -c ">" Trinity1.fasta
```

```
grep -v ">" Trinity1.fasta | wc
```

```
### 2. マッピングプログラムbowtie2###
```

```
bowtie2-build --version
```

```
mkdir rof
```

```
bowtie2-build Trinity1.fasta ./rof
```

```
ls -l rof
```

```
### 3. マッピングプログラムbowtie2###
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
  -1 data1.fq.gz -2 data2.fq
```

```
pwd
```

```
ls -l test.sam
```



```
iu@bielinux[Downloads] cd ~/Documents/srp017156/20160804
iu@bielinux[20160804] ls -la Trinity1.fasta [ 4:49午後 ]
-rw-rw-r-- 1 iu iu 3020393 6月 27 21:06 Trinity1.fasta
iu@bielinux[20160804] grep -c ">" Trinity1.fasta
2603
iu@bielinux[20160804] grep -v ">" Trinity1.fasta | wc
45921 45921 2724160
iu@bielinux[20160804] [ 4:49午後 ]
```

①配列数は2,603個、②総塩基数は  
2,724,160 – 45,921 = 2,678,239 bp  
だったことを確認

# 推奨パイプライン2

①マッピングプログラムbowtie2用のインデックスを作成。これはBLAST実行前にデータベース側の配列を前処理するのと同じような作業という理解でよい

Recommended pipeline to run TIGAR2

1. Prepare cDNA reference sequences in FASTA format.

e.g.) human  
http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMrna.fa.gz

e.g.) mouse  
http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMrna.fa.gz
2. Build bowtie2 index
3. Run bowtie2

For single-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna sample.fastq > sample.sam
```

For paired-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna -1 sample_1.fastq -2 sample_2.fastq > sample.sam
```

4. Run TIGAR2

For single-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```

For paired-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format

# Step2

- ・推奨バイブルайнでTIGAR2を実行(スライド201)  
[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

①実際の作業はこれら。②赤枠内をkopペ。Trinity1.fastaを入力としてbowtie2-buildプログラムを実行するわけだが、実行結果ファイルの拡張子の左側をtreniteyとして、rofというディレクトリに保存するように指定している

### 2. マッピングプログラムbowtie2用のインデックスを作成 ###

```
bowtie2-build --version  
mkdir rof  
bowtie2-build Trinity1.fasta ./rof/trenitey  
ls -l rof
```

### 3. マッピングプログラムbowtie2を実行 ###

### トリム前のpaired-endでとりあえずやる ###

```
bowtie2 --version  
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \  
-1 data1.fq.gz -2 data2.fq.gz > test.sam
```

```
pwd  
ls -l test.sam  
ls -lh test.sam
```

### 4. TIGAR2を実行 ###

```
pwd  
ls -l Trinity1.fasta test.sam  
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \  
--thread_num 2 Trinity1.fasta test.sam \  

```

# Step2

- 推奨バイブルインでTIGAR2を実行 (スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 2. マッピングプログラムbowtie2用のインデックスを作成 ###
bowtie2-build --version
```

```
mkdir rof
```

```
bowtie2-build Trinity1.fasta ./rof
```

```
ls -l rof
```

```
### 3. マッピングプログラムbowtie2用のインデックスを作成 ###
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
-1 data1.fq.gz -2 data2.fq.gz
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-maste
```

```
--thread_num 2 Trinity1.fasta
```

```
iu@bielinu[20160804] bowtie2-build --version [ 5:36午後]
bowtie2-build version 2.2.4
64-bit
Built on lgw01-04
Fri Dec 12 17:13:13 UTC 2014
Compiler: gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1)
Options: -O3 -m64 -msse2 -funroll-loops -g3 -DPOPCNT_CAPABILITY
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8,
8, 8, 8, 8}
iu@bielinu[20160804] mkdir rof [ 5:36午後]
iu@bielinu[20160804] bowtie2-build Trinity1.fasta ./rof/trynitye
```

# Step2

bowtie2-build実行が無事終了したので、①rofディレクトリの中身を確認。②拡張子の左側のtreniteyの意味がわかったのではないですか

- 推奨バイブルインでTIGAR2を実行

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 2. マッピングプログラムbowtie2用のインデックスを作成 ###
bowtie2-build --version
```

```
mkdir rof
```

```
bowtie2-build Trinity1.fasta ./rof
```

```
ls -l rof
```

```
### 3. マッピングプログラムbowtie2用のインデックスを作成
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
-1 data1.fq.gz -2 data2.fq
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-maste
```

```
--thread_num 2 Trinity1.fasta
```

The terminal window shows the following output:

```

sideBwtSz: 48
sideBwtLen: 192
numSides: 13950
numLines: 13950
ebwtTotLen: 892800
ebwtTotSz: 892800
color: 0
reverse: 1
Total time for backward call to driver() for mirror index: 0
0:00:02
iu@bielinux[20160804] ls -l rof
total 12584
-rw-rw-r-- 1 iu iu 5422527 6月 30 17:42 trenitey.1.bt2
-rw-rw-r-- 1 iu iu 669564 6月 30 17:42 trenitey.2.bt2
-rw-rw-r-- 1 iu iu 23435 6月 30 17:42 trenitey.3.bt2
-rw-rw-r-- 1 iu iu 669560 6月 30 17:42 trenitey.4.bt2
-rw-rw-r-- 1 iu iu 5422527 6月 30 17:42 trenitey.rev.1.bt2
-rw-rw-r-- 1 iu iu 669564 6月 30 17:42 trenitey.rev.2.bt2
iu@bielinux[20160804]

```

Red arrows point to the terminal output at the top and the file listing in the rof directory. A red arrow also points to the timestamp [ 5:42午後] at the end of the file listing.

# 推奨パイプライン3

①マッピングプログラムbowtie2を実行。②paired-endの場合

```
1. Prepare cDNA reference sequences in FASTA format.  
e.g.) human  
http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMrna.fa.gz  
e.g.) mouse  
http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMrna.fa.gz  
2. Build bowtie2 index  
mkdir ref  
bowtie2-build refMrna.fa ./ref/refMrna  
3. Run bowtie2  
For single-end data  
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna sample.fastq > sample.sam  
For paired-end data  
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna -1 sample_1.fastq -2 sample_2.fastq > sample.sam  
4. Run TIGAR2  
For single-end data  
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --alpha_zero 0.1 sample_out.txt  
For paired-end data  
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt  
Output format
```

# Step3

- 推奨パイプラインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

①bowtie2のバージョンや実行時のオプション情報を調べ、TIGAR2の推奨オプション情報から②自分の環境に合わせたオプションを選択。-xは./rof/treniteyがインデックスだと認識させるのに必要

```
### 3. マッピングプログラムbowtie2を実行 ###
### トリム前のpaired-endでとりあえずやる ###

bowtie2 --version
bowtie2 -h

bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \
    -1 data1.fq.gz -2 data2.fq.gz > test.sam

pwd
ls -l test.sam
ls -lh test.sam

### 4. TIGAR2を実行 ###
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
    --thread_num 2 Trinity1.fasta test.sam \
    --is_paired --alpha_zero 0.1 test_out.txt

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```

# Step3

- 推奨パイプラインでTIGAR2を実行(スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ####
### トリム前のpairendでとりあえずやる ####
```

```
bowtie2 --version
bowtie2 -h
bowtie2 -v -k 100 --very-sensitive
    -1 data1.fq.gz -2 data2.fq
pwd
ls -l test.sam
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ####
```

```
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master.jar
    --thread_num 2 Trinity1.fasta
    --is_paired --alpha_zero 0.1
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```

```
iu@bielinux[20160804] bowtie2 --version [11:13午前]
/usr/bin/.../lib/bowtie2/bin/bowtie2-align-s version 2.2.4
64-bit
Built on lgw01-04
Fri Dec 12 17:13:13 UTC 2014
Compiler: gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1)
Options: -O3 -m64 -msse2 -funroll-loops -g3 -DPOPCNT_CAPABILITY
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8,
8, 8, 8, 8}
iu@bielinux[20160804] bowtie2 -h [11:14午前]
```

①bowtie2のバージョンは2.2.4。Bio-Linuxにプレインストールされているので使える。②bowtie2のオプションを表示。ここで-pがCPU数、-kが複数個所にマップされたリードの最大レポート数(-k 100で100か所分まで表示)であることなどが分かる

# Step3

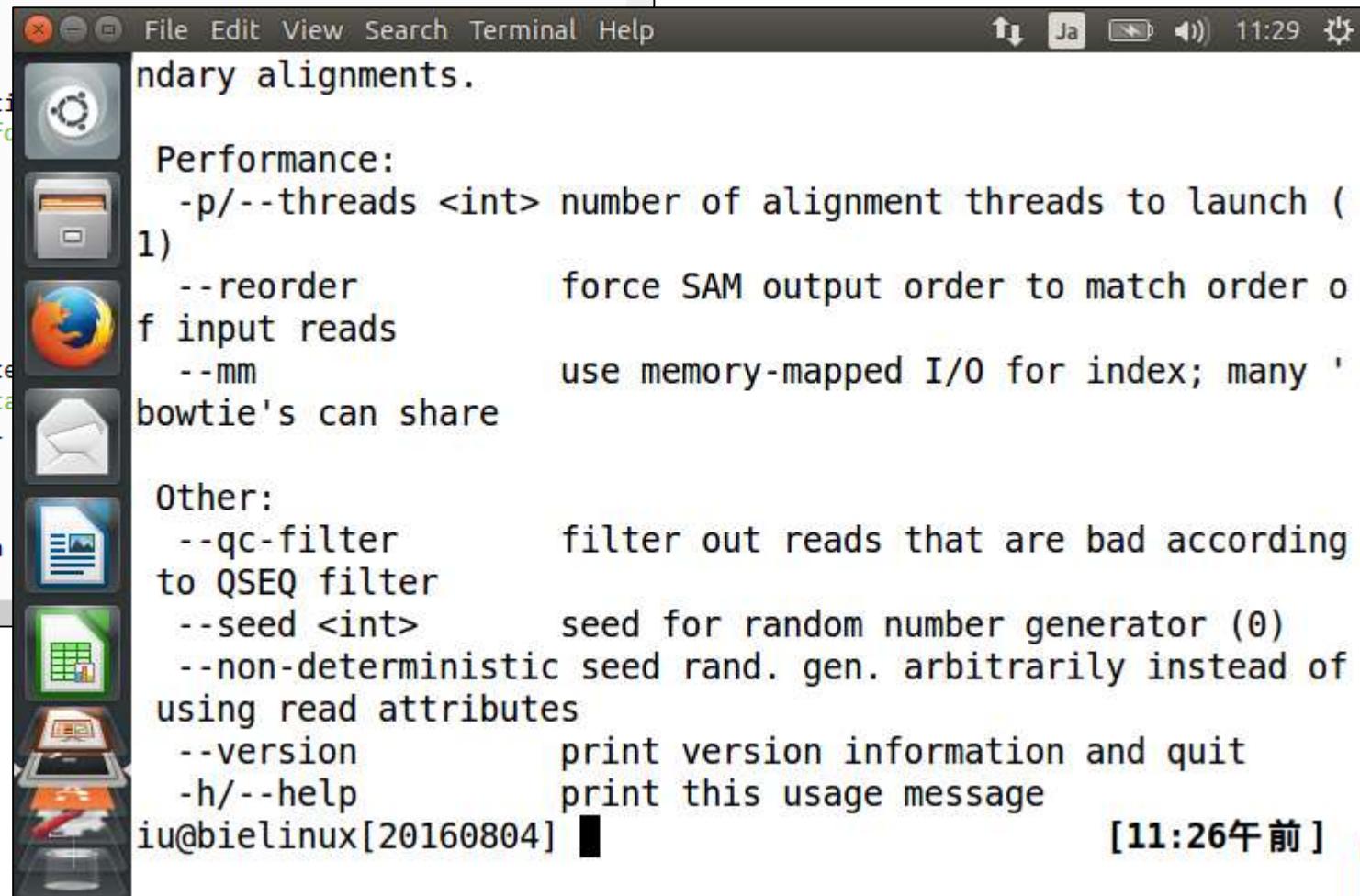
- 推奨パイプラインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ####
### トリム前のpaired-endでとりあえずやる ####
bowtie2 --version
bowtie2 -h
bowtie2 -v -k 100 --very-sensitive
      -1 data1.fq.gz -2 data2.fq
pwd
ls -l test.sam
ls -lh test.sam

### 4. TIGAR2を実行 ####
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master.jar
      --thread_num 2 Trinity1.fasta
      --is_paired --alpha_zero 0.1

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```



terminal window showing the bowtie2 -h help output. The output includes sections for Performance, Other, and command-line options like --version and -h.

```
File Edit View Search Terminal Help
Performance:
-p/--threads <int> number of alignment threads to launch (1)
--reorder force SAM output order to match order of input reads
--mm use memory-mapped I/O for index; many bowtie's can share
Other:
--qc-filter filter out reads that are bad according to QSEQ filter
--seed <int> seed for random number generator (0)
--non-deterministic seed rand. gen. arbitrarily instead of using read attributes
--version print version information and quit
-h/-help print this usage message
iu@bielinux[20160804] [11:26午前]
```

①bowtie2実行本番。出力  
ファイルはtest.sam。約5分

# Step3

- 推奨バイブルインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ###
### トリム前のpaired-endでとりあえずやる ####
bowtie2 --version
bowtie2 -h
```

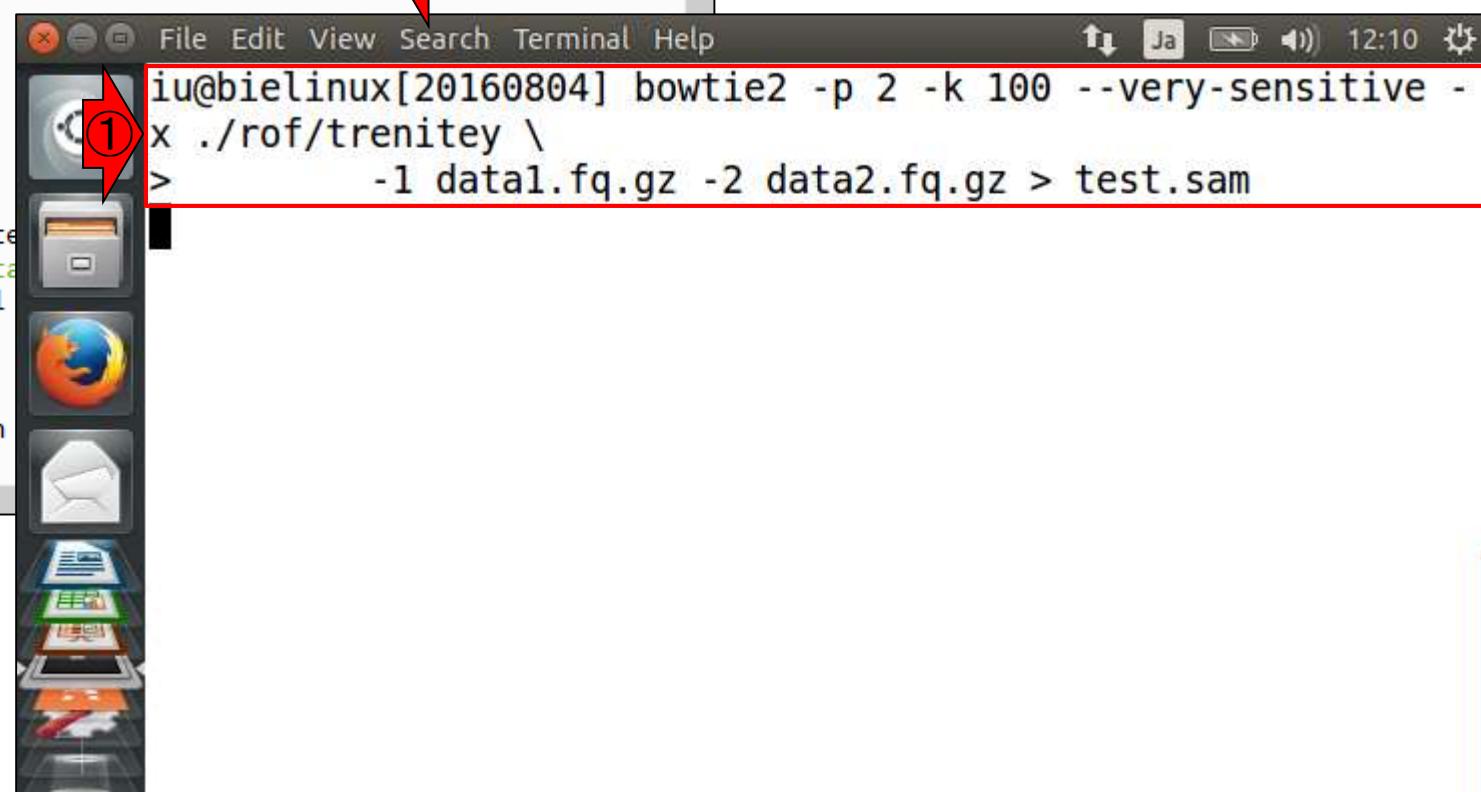
```
bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \
    -1 data1.fq.gz -2 data2.fq.gz > test.sam
```

```
pwd
ls -l test.sam
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
pwd
```

```
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master.jar
--thread_num 2 Trinity1.fasta
--is_paired --alpha_zero 0.1
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```



# Step3

- 推奨パイプラインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ###
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
-1 data1.fq.gz -2 data2.fq.gz
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
```

```
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-maste
```

```
--thread_num 2 Trinity1.fasta
```

```
--is_paired --alpha_zero 0.1
```

```
ls -l test*
```

```
head -n 5 test_out.txt
```

```
grep ">" Trinity1.fasta | head -n
```

```
iu@bielinux[20160804] bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \
> -1 data1.fq.gz -2 data2.fq.gz > test.sam
977202 reads; of these:
    977202 (100.00%) were paired; of these:
        20266 (2.07%) aligned concordantly 0 times
        928849 (95.05%) aligned concordantly exactly 1 time
        28087 (2.87%) aligned concordantly >1 times
        -
        20266 pairs aligned concordantly 0 times; of these:
            6750 (33.31%) aligned discordantly 1 time
            -
            13516 pairs aligned 0 times concordantly or discordantly
            of these:
                27032 mates make up the pairs; of these:
                    18451 (68.26%) aligned 0 times
                    6867 (25.40%) aligned exactly 1 time
                    1714 (6.34%) aligned >1 times
    99.06% overall alignment rate
iu@bielinux[20160804]
```

[12:14午後]

# Step3

・推奨バイブルインでTIGAR2を実行(スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ###
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
-1 data1.fq.gz -2 data2.fq.gz
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
```

```
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-maste
```

```
--thread_num 2 Trinity1.fasta
```

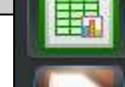
```
--is_paired --alpha_zero 0.1
```

```
ls -l test*
```

```
head -n 5 test_out.txt
```

```
grep ">" Trinity1.fasta | head -n
```

File Edit View



Bowtie2によるマッピング結果の解釈。①全977,202リード中、②928,849リード(95.05%)もマップされていることに對し、驚いている。理由は、③forward側ファイルとしてトリムなしのdata1.fq.gzを与えて実行したから。マップされる側のリファレンス配列もdata1.fq.gzを入力としたTrinity実行結果ファイルだからかもしれない…

```
iu@bielinux[20160804] bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \
> -1 data1.fq.gz -2 data2.fq.gz > test.sam
977202 reads; of these: ③
977202 (100.00%) were paired; of these:
20266 (2.07%) aligned concordantly 0 times
928849 (95.05%) aligned concordantly exactly 1 time
28087 (2.87%) aligned concordantly >1 times
-----
20266 pairs aligned concordantly 0 times; of these:
6750 (33.31%) aligned discordantly 1 time
-----
13516 pairs aligned 0 times concordantly or discordantly
of these:
27032 mates make up the pairs; of these:
18451 (68.26%) aligned 0 times
6867 (25.40%) aligned exactly 1 time
1714 (6.34%) aligned >1 times
99.06% overall alignment rate
iu@bielinux[20160804]
```

[12:14午後]

# Step3

・推奨バイブルインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ###
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
-1 data1.fq.gz -2 data2.fq.gz
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
```

```
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-maste
```

```
--thread_num 2 Trinity1.fasta
```

```
--is_paired --alpha_zero 0.1
```

```
ls -l test*
```

```
head -n 5 test_out.txt
```

```
grep ">" Trinity1.fasta | head -n
```

```

iu@bielinux[20160804] bowtie2 -p 2 -k 100 --very-sensitive -x ./rof/trenitey \
> -1 data1.fq.gz -2 data2.fq.gz > test.sam
977202 reads; of these:
    977202 (100.00%) were paired; of these:
        20266 (2.07%) aligned concordantly 0 times
        928849 (95.05%) aligned concordantly exactly 1 time ①
        28087 (2.87%) aligned concordantly >1 times
        -----
        20266 pairs aligned concordantly 0 times; of these:
            6750 (33.31%) aligned discordantly 1 time
            -----
            13516 pairs aligned 0 times concordantly or discordantly
            of these:
                27032 mates make up the pairs; of these:
                    18451 (68.26%) aligned 0 times
                    6867 (25.40%) aligned exactly 1 time
                    1714 (6.34%) aligned >1 times
99.06% overall alignment rate
iu@bielinux[20160804]

```

[12:14午後]

# Step3

- 推奨パイプラインでTIGAR2を実行(スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 3. マッピングプログラムbowtie2を実行 ###
```

```
### トリム前のpaired-endでとりあえ
```

```
bowtie2 --version
```

```
bowtie2 -h
```

```
bowtie2 -p 2 -k 100 --very-sensitive
```

```
  -1 data1.fq.gz -2 data2.fq
```

```
pwd
```

```
ls -l test.sam
```

```
ls -lh test.sam
```

```
### 4. TIGAR2を実行 ###
```

```
pwd
```

```
ls -l Trinity1.fasta test.sam
```

```
java -jar ~/Downloads/tigar2-master
```

```
  --thread_num 2 Trinity1.fasta
```

```
  --is_paired --alpha_zero 0.1
```

```
ls -l test*
```

```
head -n 5 test_out.txt
```

```
grep ">" Trinity1.fasta | head -n
```

```
iu@bielinux[20160804] pwd  
/home/iu/Documents/srp017156/20160804  
iu@bielinux[20160804] ls -l test.sam [ 1:34午後]  
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam  
iu@bielinux[20160804] ls -lh test.sam [ 1:34午後]  
-rw-rw-r-- 1 iu iu 656M 7月 1 13:31 test.sam  
iu@bielinux[20160804] [ 1:34午後]
```

①lsでファイルサイズを確認。  
②656MB。結構デカいですね

# 推奨パイプライン4

Recommended pipeline to run TIGAR2

1. Prepare cDNA reference sequences in FASTA format.

```
e.g.) human  
http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMrna.fa.gz
```

```
e.g.) mouse  
http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMrna.fa.gz
```
2. Build bowtie2 index

```
mkdir ref  
bowtie2-build refMrna.fa ./ref/refMrna
```
3. Run bowtie2  
For single-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMrna sample.fastq > sample.sam
```
4. Run TIGAR2  
For single-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```
- For paired-end data  

```
java -jar Tigar2_1.jar --thread_num 8 refMrna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format

# Step4

- 推奨バイブルインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
    --thread_num 2 Trinity1.fasta test.sam \
    --is_paired --alpha_zero 0.1 test_out.txt

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```



①TIGAR2を実行。出力ファイルはtest\_out.txt。約5分

# Step4

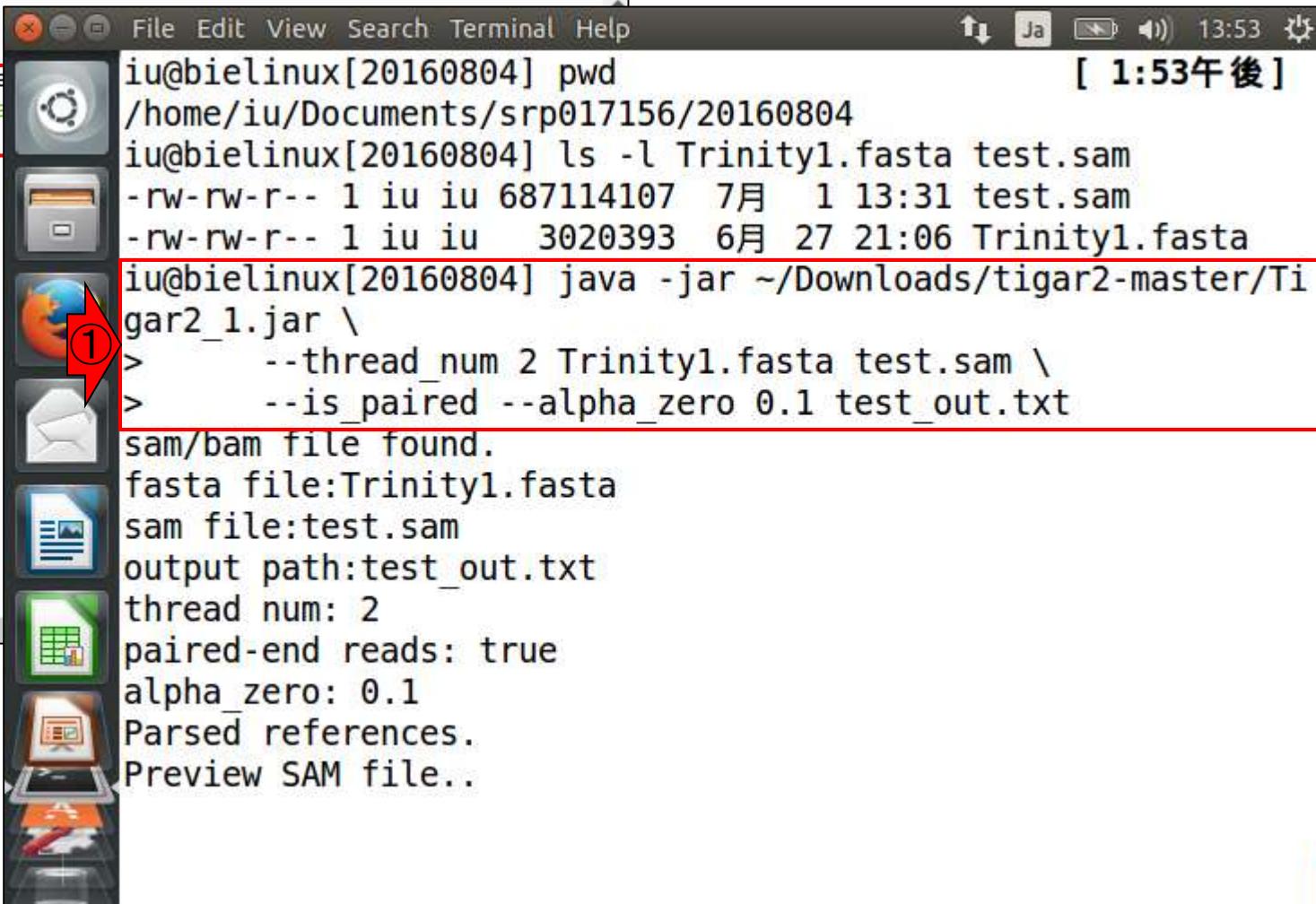
- 推奨パイプラインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
```

```
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Ti
gar2_1.jar \
--thread_num 2 Trinity1.fasta \
--is_paired --alpha_zero 0.1
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```



```
iu@bielinux[20160804] pwd
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l Trinity1.fasta test.sam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam
-rw-rw-r-- 1 iu iu 3020393 6月 27 21:06 Trinity1.fasta
iu@bielinux[20160804] java -jar ~/Downloads/tigar2-master/Ti
gar2_1.jar \
> --thread_num 2 Trinity1.fasta test.sam \
> --is_paired --alpha_zero 0.1 test_out.txt
sam/bam file found.
fasta file:Trinity1.fasta
sam file:test.sam
output path:test_out.txt
thread num: 2
paired-end reads: true
alpha_zero: 0.1
Parsed references.
Preview SAM file..
```

# Step4

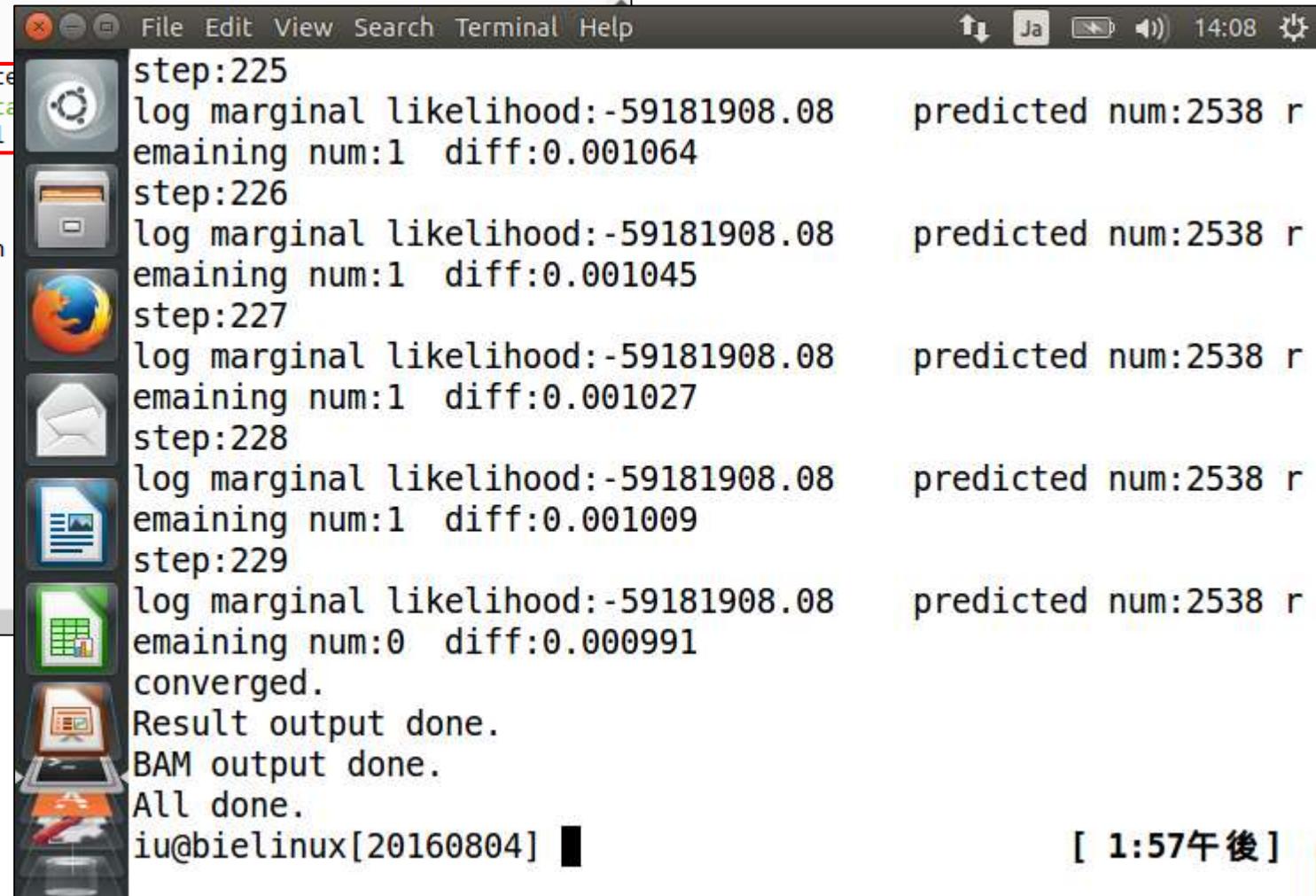
- 推奨パイプラインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
```

```
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master.jar
--thread_num 2 Trinity1.fasta
--is_paired --alpha_zero 0.1
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```



```
step:225
log marginal likelihood:-59181908.08      predicted num:2538 r
emaining num:1 diff:0.001064
step:226
log marginal likelihood:-59181908.08      predicted num:2538 r
emaining num:1 diff:0.001045
step:227
log marginal likelihood:-59181908.08      predicted num:2538 r
emaining num:1 diff:0.001027
step:228
log marginal likelihood:-59181908.08      predicted num:2538 r
emaining num:1 diff:0.001009
step:229
log marginal likelihood:-59181908.08      predicted num:2538 r
emaining num:0 diff:0.000991
converged.
Result output done.
BAM output done.
All done.
iu@bielinux[20160804] [ 1:57午後]
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADMEを眺めつつ、BoostとBridgerのインストール、サンプルデータでコケル
- 発現量推定
  - TIGAR2のダウンロード、解凍、動作確認
  - 推奨パイプラインに従って実行、結果の解釈、FPKM (RPKM)値を手計算



# 出力ファイル形式

The screenshot shows a GitHub page for the TIGAR2 project. The main content is a text file detailing the output format. A red box highlights the first four items: ID, LENGTH, Z, and FPKM. Red arrows numbered 1 through 3 point from these highlighted terms to corresponding explanatory text further down the page.

**Output format**

ID: transcript (mRNA) ID that the program predicted

LENGTH: transcript length

Z: the number of expected fragments that the program assigned to the transcript

FPKM: normalized expression level (Fragments Per Kilobase of exon per Million mapped fragments)

THETA: estimated parameter (transcript abundance), essentially Z divided by total fragments.

**5. Visualization**

You can visualize the optimized alignment by TIGAR2 as follows:

```
samtools sort sample_out.txt.opt.bam sample_opt_sorted  
samtools index sample_opt_sorted
```

Please start NGV\_2.3.14 or later to view the optimized alignment of reads.

Please note that the current version of NGV does not support all cases; please specify the memory limit for Java:

- e.g.) java -Xmx1g -Xms1g
- e.g.) java -Xmx32g -Xms32g
- e.g.) java -Xmx64g -Xms64g

You can also choose to build an FM-index for faster search.

\* Build FM-Index for [Build]

**Explanatory text (highlighted by a yellow box and connected to the numbered arrows):**

出力ファイル(test\_out.txt)の形式についての説明。全部で5列からなる。  
①FPKM値が目的の発現量情報。  
②Zは、マップされたフラグメント数。paired-endなのでfragmentという表現になる。  
single-endのときのマップされたリード数のpaired-end版という理解でよい。  
③THETAは、全フラグメントに対するZの割合という理解でよいが、事实上使うことはない

**Annotations:**

- ① points to the FPKM definition
- ② points to the Z definition
- ③ points to the THETA definition

# 出力ファイル確認

- 推奨パイプラインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
    --thread_num 2 Trinity1.fasta test.sam \
    --is_paired --alpha_zero 0.1 test_out.txt
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```

```
File Edit View Terminal Help
iu@bielinux[20160804] ls -l test* [ 3:05午後]
-rw-rw-r-- 1 iu iu 151845 7月 1 13:56 test_out.txt
-rw-rw-r-- 1 iu iu 171995059 7月 1 13:56 test_out.txt.opt.bam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam
iu@bielinux[20160804] head -n 5 test_out.txt [ 3:06午後]
ID LENGTH Z FPKM THETA
TRINITY_DN1285_c0_g1_i1 643 11.00 17.6142113 1.125663e-05
TRINITY_DN1238_c0_g1_i1 325 56.00 177.4132920 5.730647e-05
TRINITY_DN1264_c0_g1_i1 303 8.00 27.1849693 8.186639e-06
TRINITY_DN1239_c0_g1_i1 297 4.00 13.8670803 4.093320e-06
iu@bielinux[20160804] [ 3:06午後]
```

# 出力ファイル確認

・推奨バイブルインでTIGAR2を実行(スライド201)

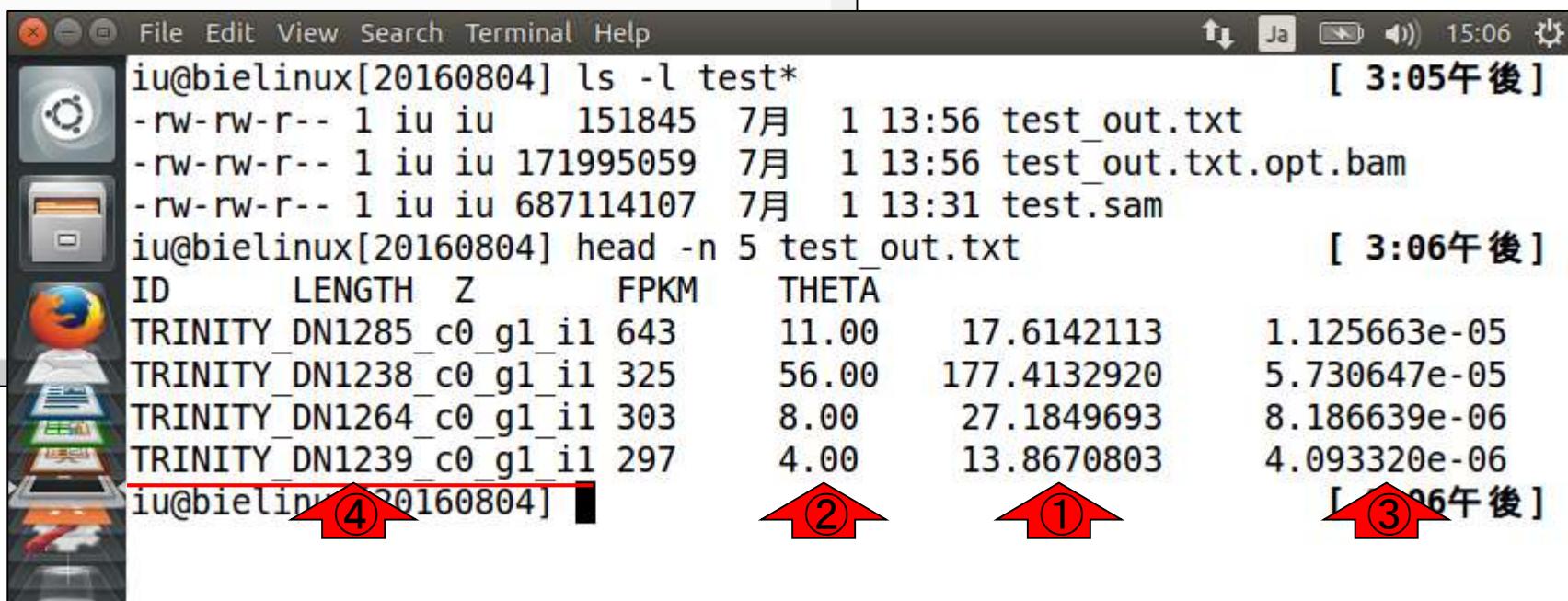
[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

### 4. TIGAR2を実行 ###

```
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
--thread_num 2 Trinity1.fasta test.sam \
--is_paired --alpha_zero 0.1 test_out.txt
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```

①目的の発現量情報に相当するFPKM値、  
②マップされたフラグメント数(Z)、③全フラグメントに対するZの割合(THETA)。④ID情報は、リファレンス配列(Trinity1.fasta)のdescription行に記載されているものと同じ



```
File Edit View Terminal Help
iu@bielinux[20160804] ls -l test*
-rw-rw-r-- 1 iu iu 151845 7月 1 13:56 test_out.txt
-rw-rw-r-- 1 iu iu 171995059 7月 1 13:56 test_out.txt.opt.bam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam
iu@bielinux[20160804] head -n 5 test_out.txt
ID LENGTH Z FPKM THETA
TRINITY_DN1285_c0_g1_i1 643 11.00 17.6142113 1.125663e-05
TRINITY_DN1238_c0_g1_i1 325 56.00 177.4132920 5.730647e-05
TRINITY_DN1264_c0_g1_i1 303 8.00 27.1849693 8.186639e-06
TRINITY_DN1239_c0_g1_i1 297 4.00 13.8670803 4.093320e-06
iu@bielinux[20160804]
```

The terminal window shows the execution of a pipeline to run TIGAR2. It lists the files generated: test\_out.txt, test\_out.txt.opt.bam, and test.sam. Then, it displays the top 5 lines of test\_out.txt, which is a tab-separated table. Red arrows point to specific elements in this table:

- ④ points to the 'ID' column header.
- ② points to the 'Z' column header.
- ① points to the 'FPKM' column header.
- ③ points to the 'THETA' column header.

①つまりリファレンス配列([Trinity1.fasta](#))のdescription行の赤下線部分と同じ

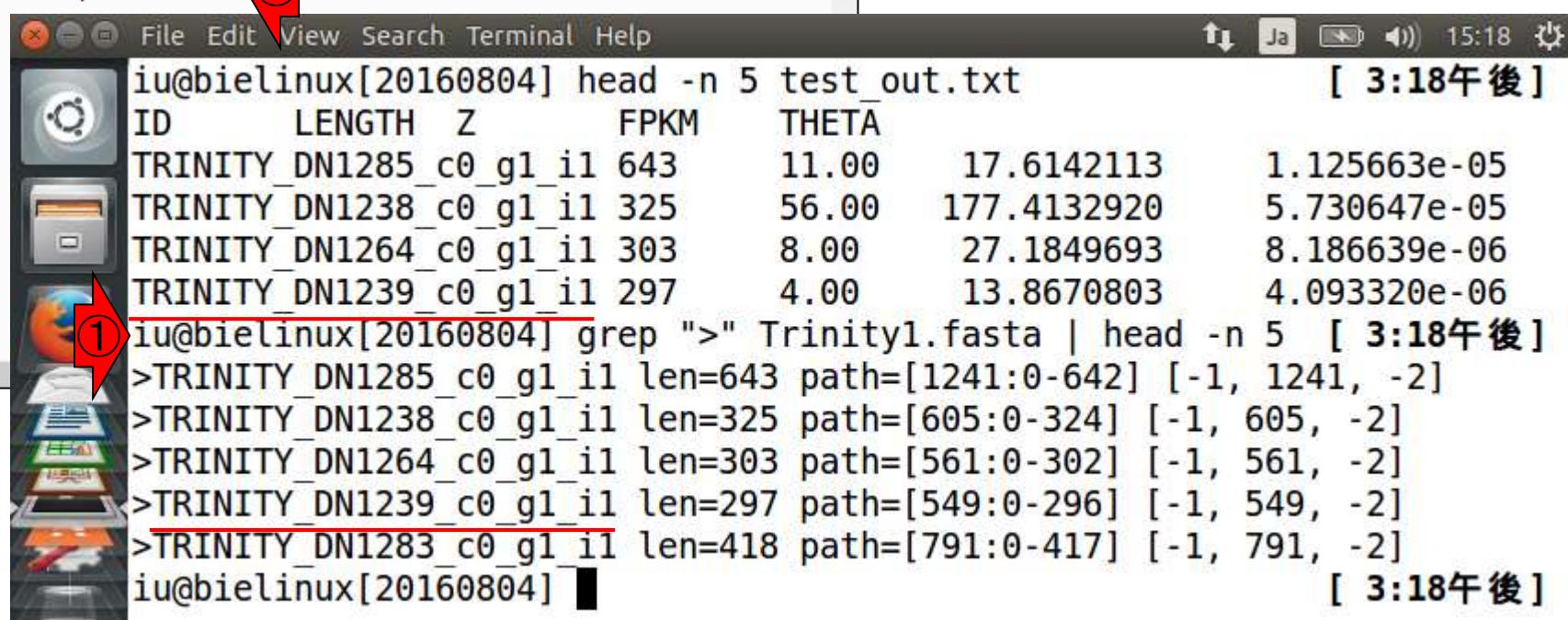
# 出力ファイル確認

- 推奨パイプラインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
  --thread_num 2 Trinity1.fasta test.sam \
  --is_paired --alpha_zero 0.1 test\_out.txt

ls -l test*
head -n 5 test\_out.txt
grep ">" Trinity1.fasta | head -n 5
```



```
iu@bielinux[20160804] head -n 5 test\_out.txt [ 3:18午後]
ID      LENGTH   Z      FPKM    THETA
TRINITY_DN1285_c0_g1_i1 643     11.00    17.6142113  1.125663e-05
TRINITY_DN1238_c0_g1_i1 325     56.00    177.4132920 5.730647e-05
TRINITY_DN1264_c0_g1_i1 303     8.00     27.1849693  8.186639e-06
TRINITY_DN1239_c0_g1_i1 297     4.00     13.8670803  4.093320e-06
iu@bielinux[20160804] grep ">" Trinity1.fasta | head -n 5 [ 3:18午後]
>TRINITY_DN1285_c0_g1_i1 len=643 path=[1241:0-642] [-1, 1241, -2]
>TRINITY_DN1238_c0_g1_i1 len=325 path=[605:0-324] [-1, 605, -2]
>TRINITY_DN1264_c0_g1_i1 len=303 path=[561:0-302] [-1, 561, -2]
>TRINITY_DN1239_c0_g1_i1 len=297 path=[549:0-296] [-1, 549, -2]
>TRINITY_DN1283_c0_g1_i1 len=418 path=[791:0-417] [-1, 791, -2]
iu@bielinux[20160804]
```

# FPKM値を手計算

参考

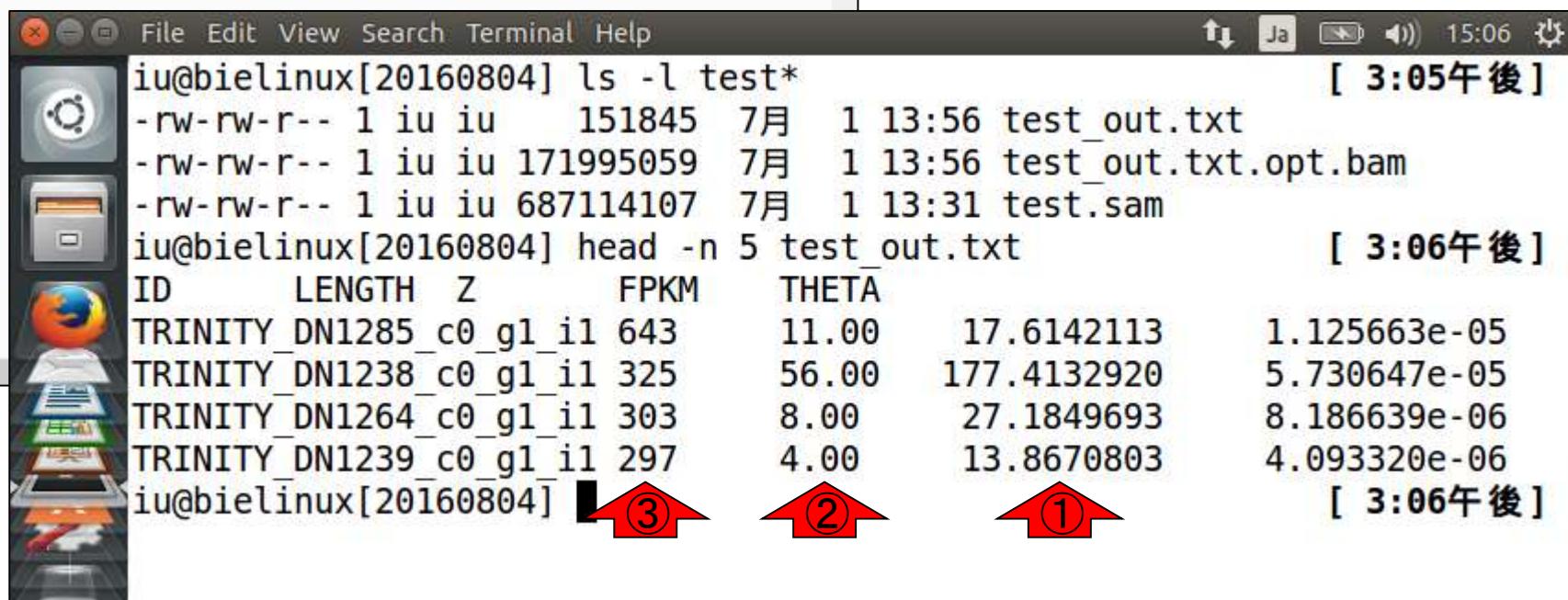
- 推奨バイブルайнでTIGAR2を実行 (スライド201)

TIGAR2の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
pwd
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
  --thread_num 2 Trinity1.fasta test.sam \
  --is_paired --alpha_zero 0.1 test_out.txt

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```

①FPKM値は、②マップされたフラグメント数(Z)とこの列の総和(フラグメント数の総和)、および③配列長(LENGTH)情報を用いて手計算可能



```
iu@bielinux[20160804] ls -l test* [ 3:05午後]
-rw-rw-r-- 1 iu iu 151845 7月 1 13:56 test_out.txt
-rw-rw-r-- 1 iu iu 171995059 7月 1 13:56 test_out.txt.opt.bam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam

iu@bielinux[20160804] head -n 5 test_out.txt [ 3:06午後]
ID      LENGTH   Z      FPKM      THETA
TRINITY_DN1285_c0_g1_i1 643      11.00    17.6142113  1.125663e-05
TRINITY_DN1238_c0_g1_i1 325      56.00    177.4132920 5.730647e-05
TRINITY_DN1264_c0_g1_i1 303      8.00     27.1849693  8.186639e-06
TRINITY_DN1239_c0_g1_i1 297      4.00     13.8670803  4.093320e-06

iu@bielinux[20160804]
```

③ LENGTH  
② Z  
① FPKM

# FPKM値を手計算

参考

①3列目にあるマップされたフラグメント数(Z)の総和は、②で計算可能

- FPKM値を手計算 (スライド226)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長情報(LENGTH)および3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任意の文字で構いません。\$3は3列目という意味です。

```
cd ~/Documents/srp017156/20160804
# wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes
# cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}' <span style="color: red;">②
head -n 5 test_out.txt
```

### Rを起動してFPKM値を手計算 ###

R -q

11 \* (1000000/971222) \* (1000/642) \* "TRINITY\_DN1285\_c0\_g1\_i1"

56 \* (1000000/971222) \* (1000/325) \* "TRINITY\_DN1238\_c0\_g1\_i1"

8 \* (1000000/971222) \* (1000/303) \* "TRINITY\_DN1264\_c0\_g1\_i1"

q(save="no")

The terminal window shows the following session:

```
iu@bielinux[20160804] ls -l test*
-rw-rw-r-- 1 iu iu 151845 7月 1 13:56 test_out.txt
-rw-rw-r-- 1 iu iu 171995059 7月 1 13:56 test_out.txt.opt.bam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam

iu@bielinux[20160804] head -n 5 test_out.txt
ID      LENGTH  Z      FPKM      THETA
TRINITY_DN1285_c0_g1_i1 643      11.00    17.6142113   1.125663e-05
TRINITY_DN1238_c0_g1_i1 325      56.00    177.4132920  5.730647e-05
TRINITY_DN1264_c0_g1_i1 303      8.00     27.1849693   8.186639e-06
TRINITY_DN1239_c0_g1_i1 297      4.00     13.8670803   4.093320e-06

iu@bielinux[20160804]
```

Red arrows point to two specific parts of the output:

- ②** Points to the command `cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'` in the terminal.
- ①** Points to the first four columns of the table output: ID, LENGTH, Z, and FPKM.

# FPKM値を手計算

参考

## FPKM値を手計算 (スライド22)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長情報(LENGTH)および3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任意の文字で構成。\$3は3列目という意味です。

①3列目にあるマップされたフラグメント数(Z)の総和は、971,222。この計算結果を確かめたければ、自分で②[test\\_out.txt](#)をホストOS上のエクセルなどで開いて3列目の総和を計算すればよい

```
cd ~/Documents/srp017156/20160804  
#wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes  
#cp ~/Desktop/backup/test_out.txt .  
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'  
head -n 5 test_out
```

The terminal window shows the command being run and its output. A red arrow labeled ① points to the 'oe' variable in the awk command. Another red arrow labeled ② points to the 'oe' variable in the output, which is '971222'. The terminal window also displays the R session history and the command to start R.

```
## Rを起動してFPKM値を手計算する  
R -q  
11 * (1000000/971222)  
56 * (1000000/971222)  
8 * (1000000/971222)  
q(save="no")  
  
iu@bielinux[20160804] cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'  
971222  
iu@bielinux[20160804] [ 5:15午後]
```

# FPKM値を手計算

参考

赤枠部分の実行結果。①の転写物の②FPKM値計算結果(17.61421)は、ピタリと一致。2016.07.21のスライド19(RPKM補正)の計算式と同じことに気づく

## FPKM値を手計算 (スライド226)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長情報(LENGTH)および3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任意の文字で構いません。\$3は3列目という意味です。

```
cd ~/Documents/srp017156/20160804
# wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes
# cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

教科書p132-137



## Rを起動してFPKM値を計算

```
R -q
11 * (1000000/971222)
56 * (1000000/971222)
8 * (1000000/971222)
q(save="no")
```

①

iu@bielinux[20160804] cat test\_out.txt | awk '{oe=oe+\$3} END{print oe;}'  
971222

iu@bielinux[20160804] head -n 5 test\_out.txt [ 5:15午後]

ID	LENGTH	Z	FPKM	THETA	
TRINITY_DN1285_c0_g1_i1	643	11.00	17.6142113	1.125663e-05	
TRINITY_DN1238_c0_g1_i1	325	56.00	177.4132920	5.730647e-05	
TRINITY_DN1264_c0_g1_i1	303	8.00	27.1849693	8.186639e-06	
TRINITY_DN1239_c0_g1_i1	297	4.00	13.8670803	4.093320e-06	

iu@bielinux[20160804] R -q [ 5:19午後]

```
> 11 * (1000000/971222) * (1000/643) # "TRINITY_DN1285_c0_g1_i1"
[1] 17.61421
```

②

①

FPKM値は、①この転写物上にマップされたフラグメント数に対して、

# FPKM値を手計算

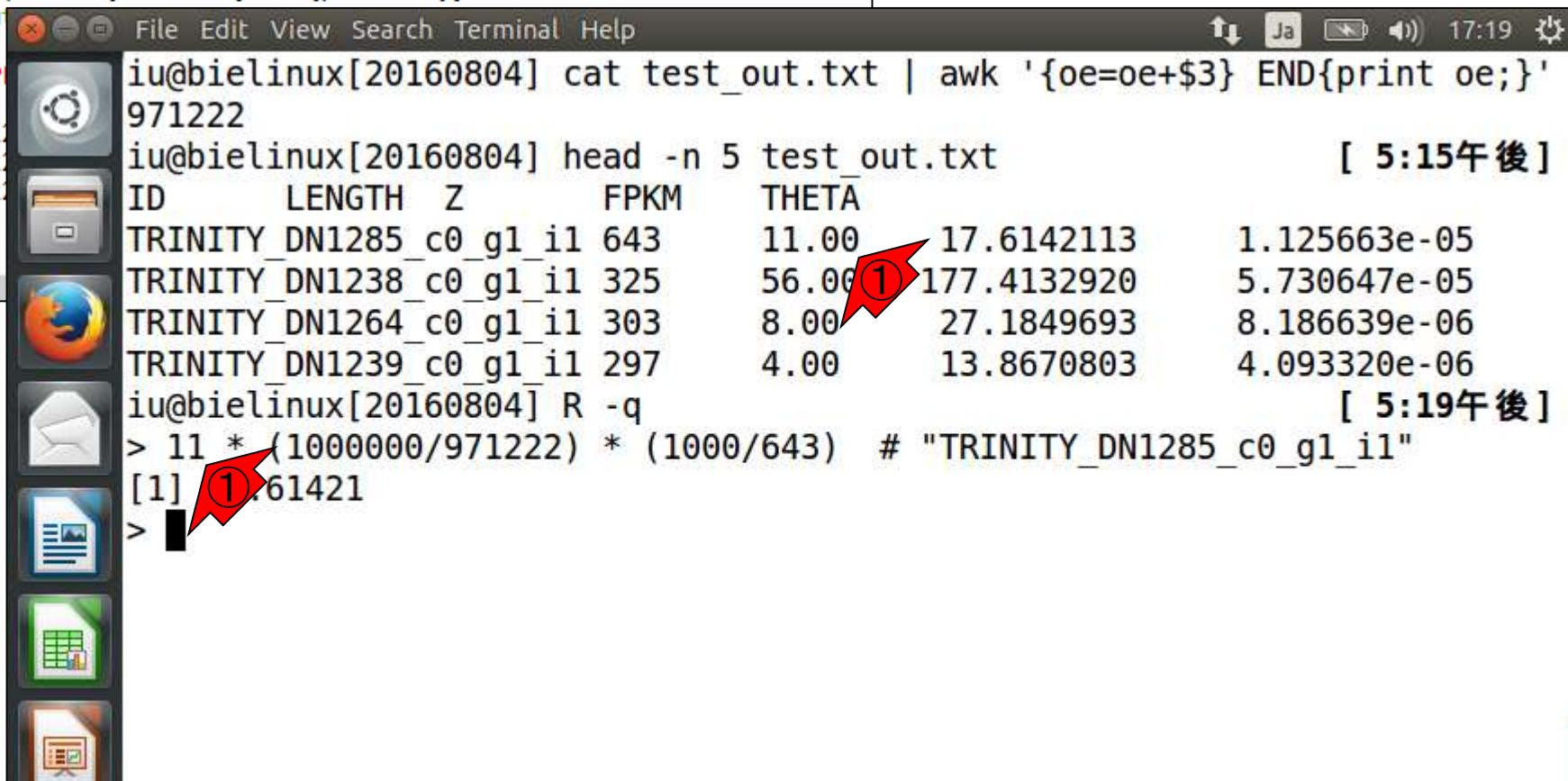
[参考](#)

- FPKM値を手計算 (スライド226)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長情報(LENGTH)および3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任意の文字で構いません。\$3は3列目という意味です。

```
cd ~/Documents/srp017156/20160804
# wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes
# cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

```
head -n 5 test_out
```



```
## Rを起動してFPKMを計算
R -q
11 * (1000000/971222)
56 * (1000000/971222)
8 * (1000000/971222)
q(save="no")
```

```
iu@bielinux[20160804] cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

```
971222
```

```
iu@bielinux[20160804] head -n 5 test_out.txt
```

ID	LENGTH	Z	FPKM	THETA	
TRINITY_DN1285_c0_g1_i1	643	11.00	17.6142113	1.125663e-05	
TRINITY_DN1238_c0_g1_i1	325	56.00	177.4132920	5.730647e-05	
TRINITY_DN1264_c0_g1_i1	303	8.00	27.1849693	8.186639e-06	
TRINITY_DN1239_c0_g1_i1	297	4.00	13.8670803	4.093320e-06	

```
iu@bielinux[20160804] R -q
```

```
> 11 * (1000000/971222) * (1000/643) # "TRINITY_DN1285_c0_g1_i1"
```

```
[1] 17.61421
```

```
>
```

参考

# FPKM値を手計算

FPKM値は、①この転写物上にマップされたフラグメント数に対して、②マップされた総フラグメント数が1000000だった場合(fragments per one million)、

## FPKM値を手計算 (スライド226)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長情報(LENGTH)および3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任意の文字で構いません。\$3は3列目という意味です。

```
cd ~/Documents/srp017156/20160804
# wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes
# cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

```
head -n 5 test_out
```

The screenshot shows a terminal window with the following session:

```
## Rを起動してFPKMを計算
R -q
11 * (1000000/971222)
56 * (1000000/971222)
8 * (1000000/971222)
q(save="no")
```

Then, the user runs:

```
iu@bielinux[20160804] cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

Output:

```
971222
```

Next, the user runs:

```
iu@bielinux[20160804] head -n 5 test_out.txt
```

Output:

ID	LENGTH	Z	FPKM	THETA	
TRINITY_DN1285_c0_g1_i1	643	11.00	17.6142113	1.125663e-05	
TRINITY_DN1238_c0_g1_i1	325	56.00	177.4132920	5.730647e-05	
TRINITY_DN1264_c0_g1_i1	303	8.00	27.1849693	8.186639e-06	
TRINITY_DN1239_c0_g1_i1	297	4.00	13.8670803	4.093320e-06	

At 5:15 PM, the user runs:

```
iu@bielinux[20160804] R -q
```

Then, they calculate:

```
> 11 * (1000000/971222) * (1000/643) # "TRINITY_DN1285_c0_g1_i1"
```

Output:

```
[1] 1.61421
```

Red annotations highlight the following:

- ① A red arrow points to the value 17.6142113 in the THETA column.
- ② A red arrow points to the value 1.61421 in the R output.
- ③ A red arrow points to the first line of the R command where the denominator 971222 is underlined.

# FPKM値を手計算 参考

## FPKM値を手計算 (スライド226)

TIGAR2実行結果ファイル([test\\_out.txt](#))の4列目のFPKM値を、2列目の配列長幅3列目のマップされたフラグメント数情報(Z)を用いて手計算する。[oe](#)の部分は任ん。\$3は3列目という意味です。

FPKM値は、①この転写物上にマップされたフラグメント数に対して、②マップされた総フラグメント数が1000000だった場合(fragments per one million)、および③配列長が1000 bpだった場合(fragments per one kilobase)で補正した値です

```
cd ~/Documents/srp017156/20160804
# wget -c http://www.iu.a.u-tokyo.ac.jp/~kadota/bioinfo_ngs_sokushu_2016/3/tes
# cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
```

```
head -n 5 test_out
```

The terminal window shows the following sequence of commands:

- Script Execution:** The user runs an R script to calculate the total mapped fragments (oe).
- Output:** The output shows the total mapped fragments as 971222.
- Data View:** The user views the first 5 lines of the test\_out.txt file.
- Table Headers:** The table headers are ID, LENGTH, Z, FPKM, and THETA.
- Data Rows:** The table displays five rows of data corresponding to the entries in the previous table.
- R Calculation:** The user performs an R calculation to calculate the FPKM value for the first entry.
- Result:** The calculated FPKM value is 11.61421.

Red arrows numbered 1, 2, and 3 point to specific parts of the R command:

- ① Points to the value 11 in the R command.
- ② Points to the division by 643 in the R command.
- ③ Points to the multiplication by 1000 in the R command.

ID	LENGTH	Z	FPKM	THETA
TRINITY_DN1285_c0_g1_i1	643	11.00	17.6142113	1.125663e-05
TRINITY_DN1238_c0_g1_i1	325	56.00	177.4132920	5.730647e-05
TRINITY_DN1264_c0_g1_i1	303	8.00	27.1849693	8.186639e-06
TRINITY_DN1239_c0_g1_i1	297	4.00	13.8670803	4.093320e-06

①TIGAR2は、②RapMap論文や、③手法比較論文中でも高評価です。スライドを見るだけ

# 発現量推定

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、ranscripトーム、正規化、発現変動、統計、モラ  
(last modified 2016/06/03, since 2011)

### What's new?

- このウェブページ  
リソース Rと必要  
法(Windows2015  
ホンダ 2015)

- [解析 | 基礎 | 平均-分散プロット | について \(last modified 2015/11/11\)](#)
- [解析 | 基礎 | 平均-分散プロット | Technical replicates \(last modified 2015/06/03\)](#)
- [解析 | 基礎 | 平均-分散プロット | Biological replicates \(last modified 2015/06/03\)](#)
- [解析 | 新規転写物同定\(ゲノム配列を利用\) \(last modified 2015/08/25\)](#)
- [解析 | 発現量推定\(ranscripトーム配列を利用\) \(last modified 2015/06/03\)](#)
- [解析 | クラスタリング | について \(last modified 2016/05/25\)](#)
- [解析 | クラスタリング | サンプル間 | hclust \(last modified 2015/02/26\)](#)
- [解析 | クラスタリング | サンプル間 | TCC\(Sun 2013\) \(last modified 2015/06/03\)](#)
- [解析 | クラスタリング | 遺伝子間\(基礎\) | MBCluster.Seq\(Si 2014\) \(last modified 2015/06/03\)](#)

### 解析 | 発現量推定(ranscripトーム配列を利用) NEW

新規転写物(新規isoform)の発見などが目的でなく、既知転写物の発現量を知りたいだけの場合には、やたらと時間がかかるゲノム配列へのマッピングを避けるのが一般的です。有名なCufflinksも一応GTF形式のアノテーションファイルを与えることでゲノム全体にマップするのを避けるモードがあるらしいので、一応リストアップしています。転写物へのマッピングの場合には、splice-aware alignerを用いたジャンクションリードのマッピングを行う必要がないので、高速にマッピング可能なbasic alignerで十分です。但し、複数個所にマップされるリードは考慮する必要があり、確率モデルのパラメータを最尤法に基づいて推定するexpectation-maximization (EM)アルゴリズムがよく用いられます。マッピングを行わずに、k-merを用いてalignment-freeで行う発現量推定を行うSailfishやRNA-Skimmerは従来法に比べて劇的に高速化がなされているようです。間違いがいくつか含まれているとは思います。2016年6月に調べた結果をリストアップします:

R用:

- [AllelicImbalance: Gådin et al., BMC Bioinformatics, 2015](#)
- [tximport: Soneson et al., F1000Res., 2015](#)
- [RNAontheBENCH \(github上有る\): Germain et al., Nucleic Acids Res., 2016](#)
- [SARTools \(github上有る\): Varet et al., PLoS One, 2016](#)

R以外:

- [Cufflinks: Trapnell et al., Nat Biotechnol., 2010](#)
- [NEUMA: Lee et al., Nucleic Acids Res., 2011](#)
- [IsoEM: Nicolae et al., Algorithms Mol. Biol., 2011](#)
- [RSEM: Li and Dewey, BMC Bioinformatics, 2011](#)
- [eXpress: Roberts and Pachter, Nat Methods, 2013](#)
- [ReXpress: Roberts et al., Bioinformatics, 2013](#)
- [TIGAR: Nariai et al., Bioinformatics, 2013](#)
- [eXpress-D: Roberts et al., BMC Bioinformatics, 2013](#)
- [PennSeq: Hu et al., Nucleic Acids Res., 2014](#)
- [Sailfish: Patro et al., Nat Biotechnol., 2014](#)
- [Quinn's pipeline\(allele-specific\): Quinn et al., Bioinformatics, 2014](#)
- [RNA-Skimmer: Zhang and Wang, Bioinformatics, 2014](#)
- [QUASAR\(allele-specific\): Harvey et al., Bioinformatics, 2015](#)
- [TIGER2: Nariai et al., BMC Genomics, 2014](#)
- [SUPPA: Alamancos et al., RNA, 2015](#)
- [folded Skellam mixture model\(allele-specific\): Lu et al., BMC Genomics, 2015](#)
- [EMSAR: Lee et al., BMC Bioinformatics, 2015](#)
- [PGSeq: Liu et al., PLoS One, 2015](#)
- [NLDMseq: Liu et al., BMC Bioinformatics, 2015](#)
- [ASE-TIGAR\(allele-specific\): Nariai et al., BMC Genomics, 2016](#)
- [SplAdder: Kahles et al., Bioinformatics, 2016](#)
- [RapMap: Srivastava et al., Bioinformatics, 2016](#)

Review、ガイドライン、パイプライン系:

- 手法比較: [Kanitz et al., Genome Biol., 2015](#)
- 手法比較(transcript-based approach vs. union of gene-based approach): [Zhao et al., PLoS One, 2015](#)
- パイプライン(Ion Proton用): [Yuan et al., BMC Genomics, 2016](#)
- パイプライン(single-cell用): [Ntranos et al., Genome Biol., 2016](#)

3

# おまけ

- 推奨パイプラインでTIGAR2を実行 (スライド201)

[TIGAR2](#)の、「Recommended pipeline to run TIGAR2」の部分です。

```
### 4. TIGAR2を実行 ###
```

```
pwd
```

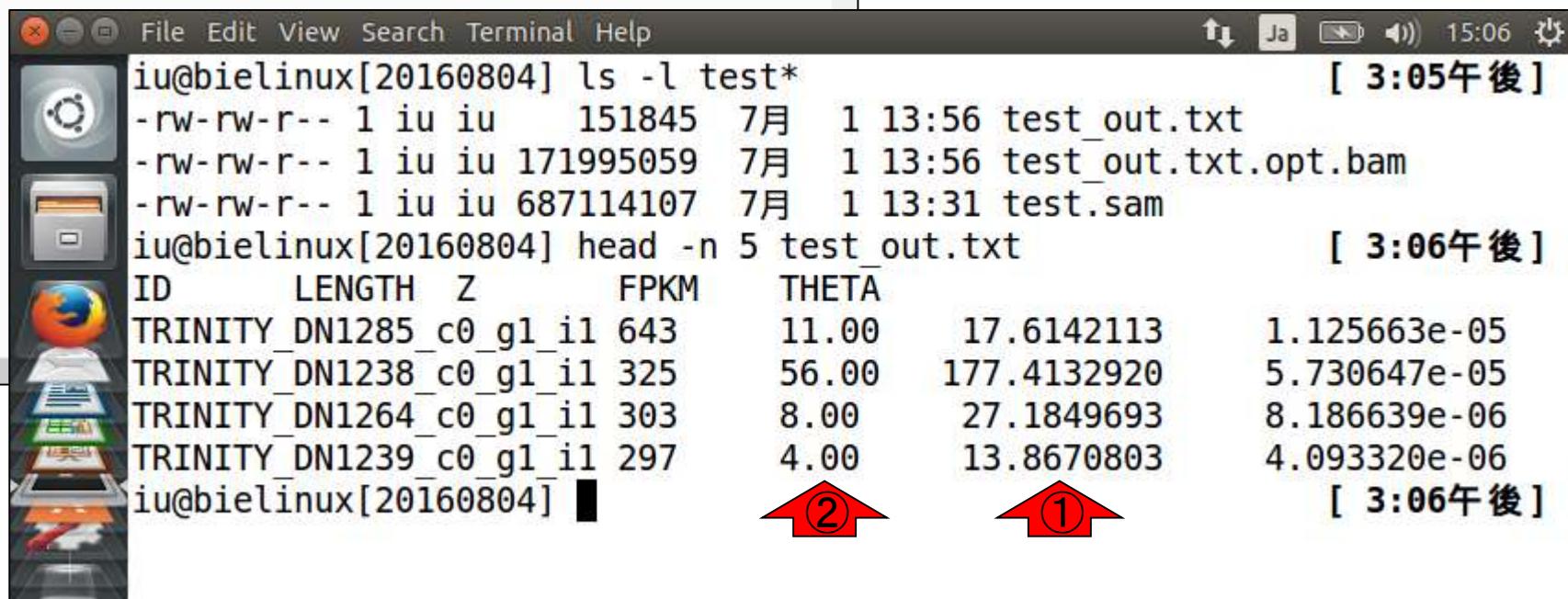
```
ls -l Trinity1.fasta test.sam
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar \
--thread_num 2 Trinity1.fasta test.sam \
--is_paired --alpha_zero 0.1 test_out.txt
```

```
ls -l test*
```

```
head -n 5 test_out.txt
```

```
grep ">" Trinity1.fasta | head -n 5
```

①講習会ではFPKM値を得る目的でTIGAR2を用いたが、②Z値はいわゆるカウント情報に相当するもの。それゆえ、TCCを用いた発現変動解析を引き続いて行いたい場合は、②のカウント情報を用います。つまり、トランスクリプトーム配列へのマッピング(bowtie2)からカウント情報取得(TIGAR2)および発現変動解析(TCC)の一連の流れを「bowtie2 → TIGAR2 → TCC」で行うこともできます



```
iu@bielinux[20160804] ls -l test*
-rw-rw-r-- 1 iu iu 151845 7月 1 13:56 test_out.txt
-rw-rw-r-- 1 iu iu 171995059 7月 1 13:56 test_out.txt.opt.bam
-rw-rw-r-- 1 iu iu 687114107 7月 1 13:31 test.sam
iu@bielinux[20160804] head -n 5 test_out.txt
ID      LENGTH   Z      FPKM    THETA
TRINITY_DN1285_c0_g1_i1 643      11.00    17.6142113  1.125663e-05
TRINITY_DN1238_c0_g1_i1 325      56.00    177.4132920 5.730647e-05
TRINITY_DN1264_c0_g1_i1 303      8.00     27.1849693  8.186639e-06
TRINITY_DN1239_c0_g1_i1 297      4.00     13.8670803  4.093320e-06
iu@bielinux[20160804]
```