

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。講習会后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](mailto:kadota@iu.a.u-tokyo.ac.jp)

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



# 利用プログラムの簡単な解説

- 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
- 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
- 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

# 2016.08.04は

## (Rで)塩基配列解析

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

### What's new

- このウェブページ
- 更新法(Win)
- 本ホームページ

- 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](#) **①**
- 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 2015/04/03)
- 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 [トランスクリプトーム解析 | 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、コピー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 | [トランスクリプトーム解析1\(2016.07.21\)](#) (last modified 2016/06/18)
- 第1部 | 統計解析 | [トランスクリプトーム解析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) | [トランスクリプトームアセンブリ、発現量推定\(2016.08.04\)](#) **②** (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*ト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、*Lactobasillus 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、うまくいかなかった原因が、②ゲノムへのマッピングでわかっ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*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)



□ 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)



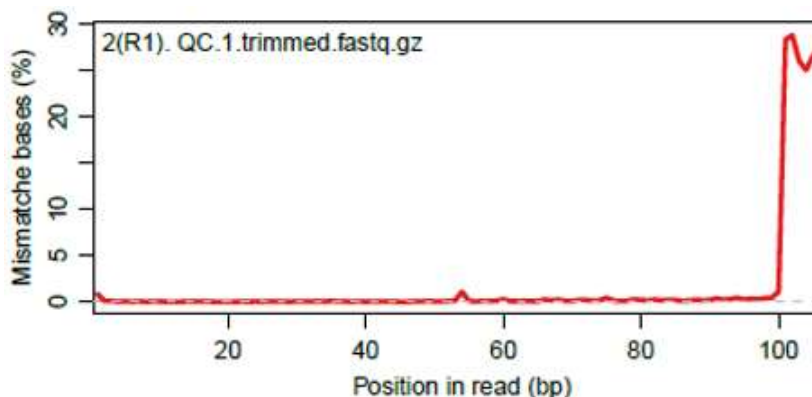
□ マップされる側: [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.30.dna.toplevel.fa.gz](http://Lactobacillus_casei_12a.GCA_000309565.2.30.dna.toplevel.fa.gz)

□ 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)

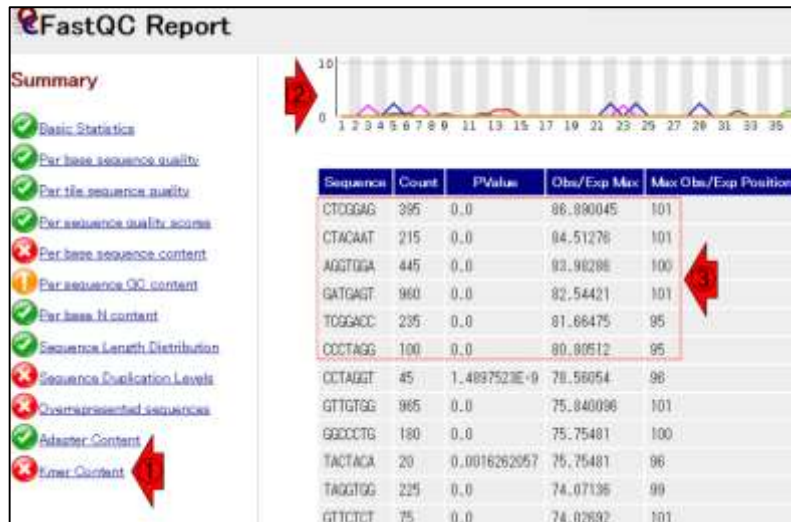
- 乳酸菌に由来しない配列を多く含む領域(forward側の100-107bp付近;  $f_{100-107}$ 問題)が見つかった



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

# おさらい3

- 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ピング率が0.4 → 34.6%と劇的に改善
- FastQC段階でf<sub>100-107</sub>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



# おさらい2

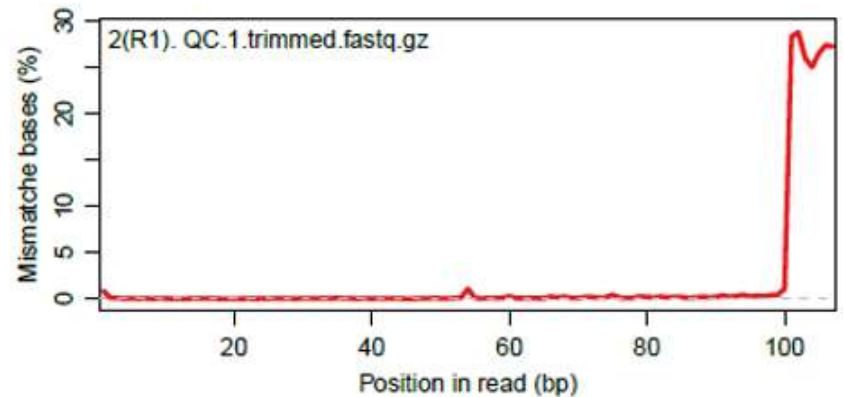
①常識的にはこの論理展開は矛盾している。②*de novo* transcriptome assemblyをやるのは、通常リファレンスゲノム配列がない場合。それゆえ、Rockhopper2の失敗原因を③ゲノムマッピング結果から突き止めるという流れは通常ない(近縁種へのマッピングなどの手段を除く)

## ■ *de novo* トランスクリプトームアセン

- ② □ 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)

- ③ □ マップされる側 : [Lactobacillus\\_casei\\_12a.GCA\\_000309565.2.30.dna.toplevel.fa.gz](http://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)

# 問題設定

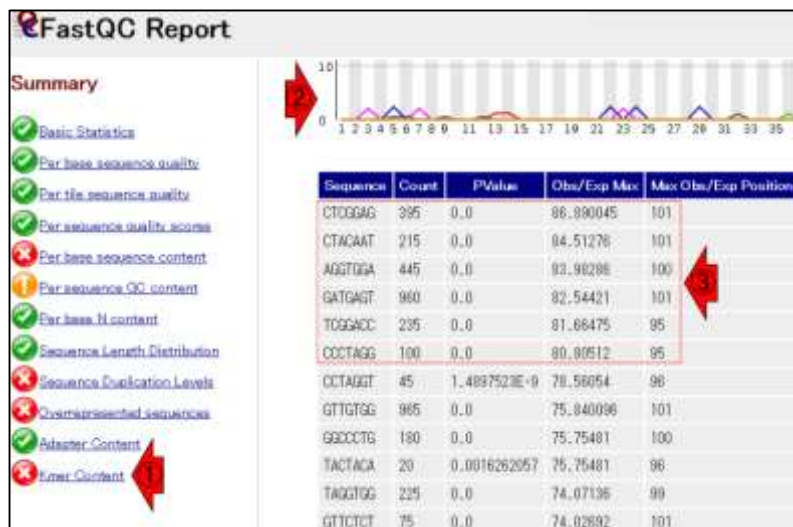
もちろん実質的には答えが既知の状態ではある。つまり、forward側の100-107bp付近をトリムすれば794 transcripts程度にはなる(第5回W17-4)というのが1つの到達目標である。我々は「過去に $f_{100-107}$ 問題を経験した」ので、②の無様な結果と③FastQCの結果をじっくり眺めて解析戦略を練る

## ■ *de novo*トランスクリプトームアセンブリ

- 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 (② 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*ト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

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(2016.08.04)

- 講義資料PDF(2016.07.xx版; 約xxxMB)
- 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.fa` 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bzip2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

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

## 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
```



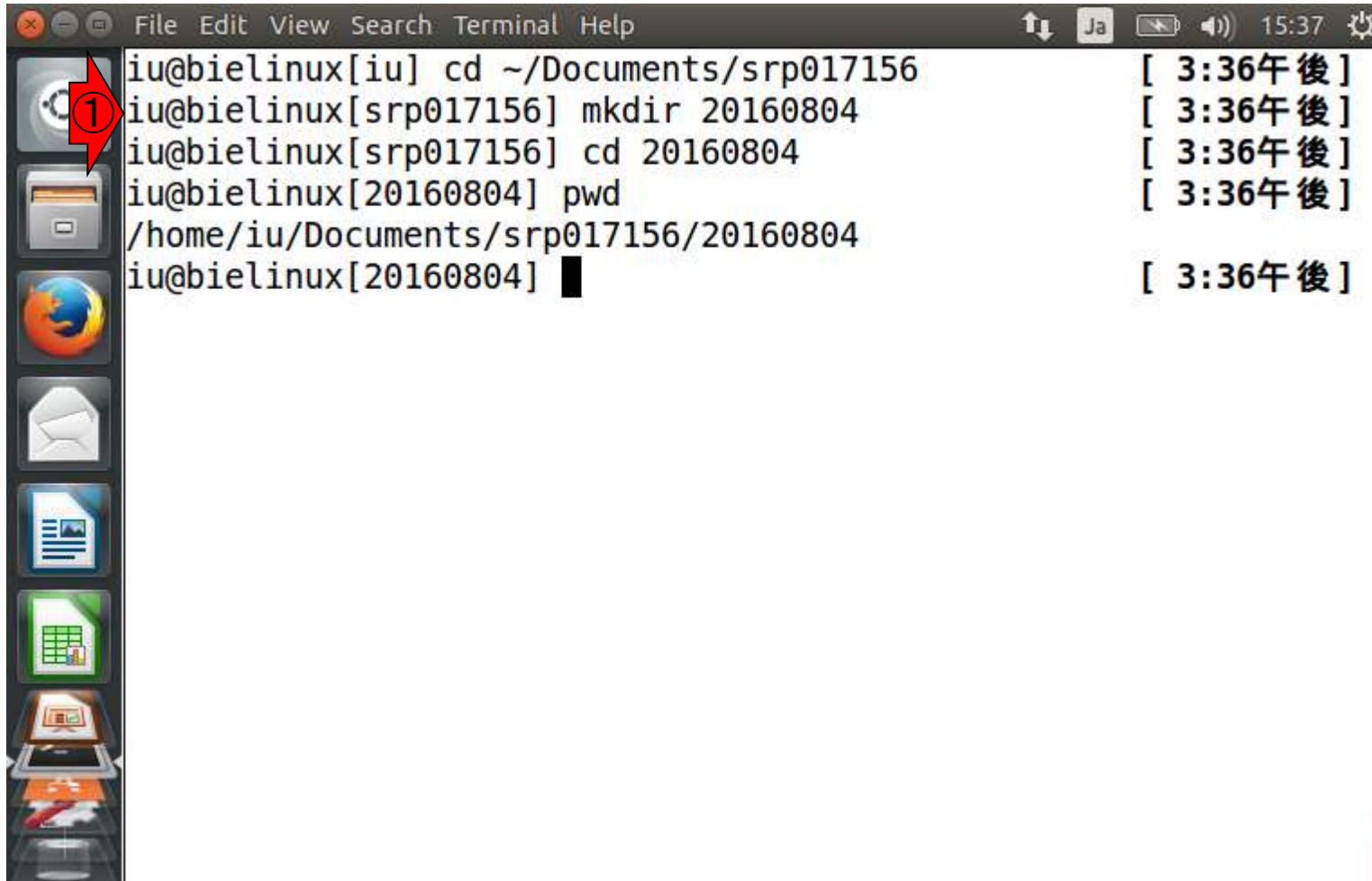
#ファイル名変更

```
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、

# 事前準備



```
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 ]
```

The terminal window shows a sequence of commands to create and navigate into a directory named '20160804' within the path ~/Documents/srp017156. A red arrow with the number '1' points to the 'mkdir' command.

# 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 [ 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] █ [ 3:38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、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
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/srp017156/20160804/20160804_FaQCs.gz 20160804/
#cp ~/Desktop/backup/20160804_FaQCs.gz 20160804/
ls -l
```

```
#ファイル名変更
mv QC.1.trimmed.fastq.gz QC.1.trimmed.fastq.gz.gz
mv QC.2.trimmed.fastq.gz QC.2.trimmed.fastq.gz.gz
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**



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

# 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/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
```

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





# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ト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

(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。画面が流れて全体像がわかりづら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

## FastQC

FastQCを実行し、結果を共有フォルダ(~/Desktop)に保存する。実行体はFastQC (ver. 0.11.4)であり、第4回WS9のリードの場合に10番目以降の塩基を一定幅でトリミングするオプション(第5回W19-1)。約30秒。

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

```
pwd
ls -l
fastqc2 -v
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。①forward側、②reverse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
```

```
File Edit View Search Terminal Help
iu@bielinux[20160804] pwd [ 4:20午後 ]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l [ 4:20午後 ]
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] fastqc2 -v [ 4:20午後 ]
FastQC v0.11.4
iu@bielinux[20160804] fastqc2 -q --nogroup data1.fq.gz --outdir
=/home/iu/Desktop/mac_share
iu@bielinux[20160804] fastqc2 -q --nogroup data2.fq.gz --outdir
=/home/iu/Desktop/mac_share
iu@bielinux[20160804] ls -l ~/Desktop/mac_share/data*
-rwxrwxrwx 1 iu iu 421637 6月 22 16:20 /home/iu/Desktop/mac_sh
are/data1_fastqc.html
-rwxrwxrwx 1 iu iu 46123 6月 22 16:20 /home/iu/Desktop/mac_sh
are/data1_fastqc.zip ①
-rwxrwxrwx 1 iu iu 392817 6月 22 16:20 /home/iu/Desktop/mac_sh
are/data2_fastqc.html
-rwxrwxrwx 1 iu iu 42128 6月 22 16:20 /home/iu/Desktop/mac_sh
are/data2_fastqc.zip ②
iu@bielinux[20160804] [ 4:21午後 ]
```

# 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)」

FastQC Report

Wed 22 Jun 2016  
data1.fq.gz

### 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

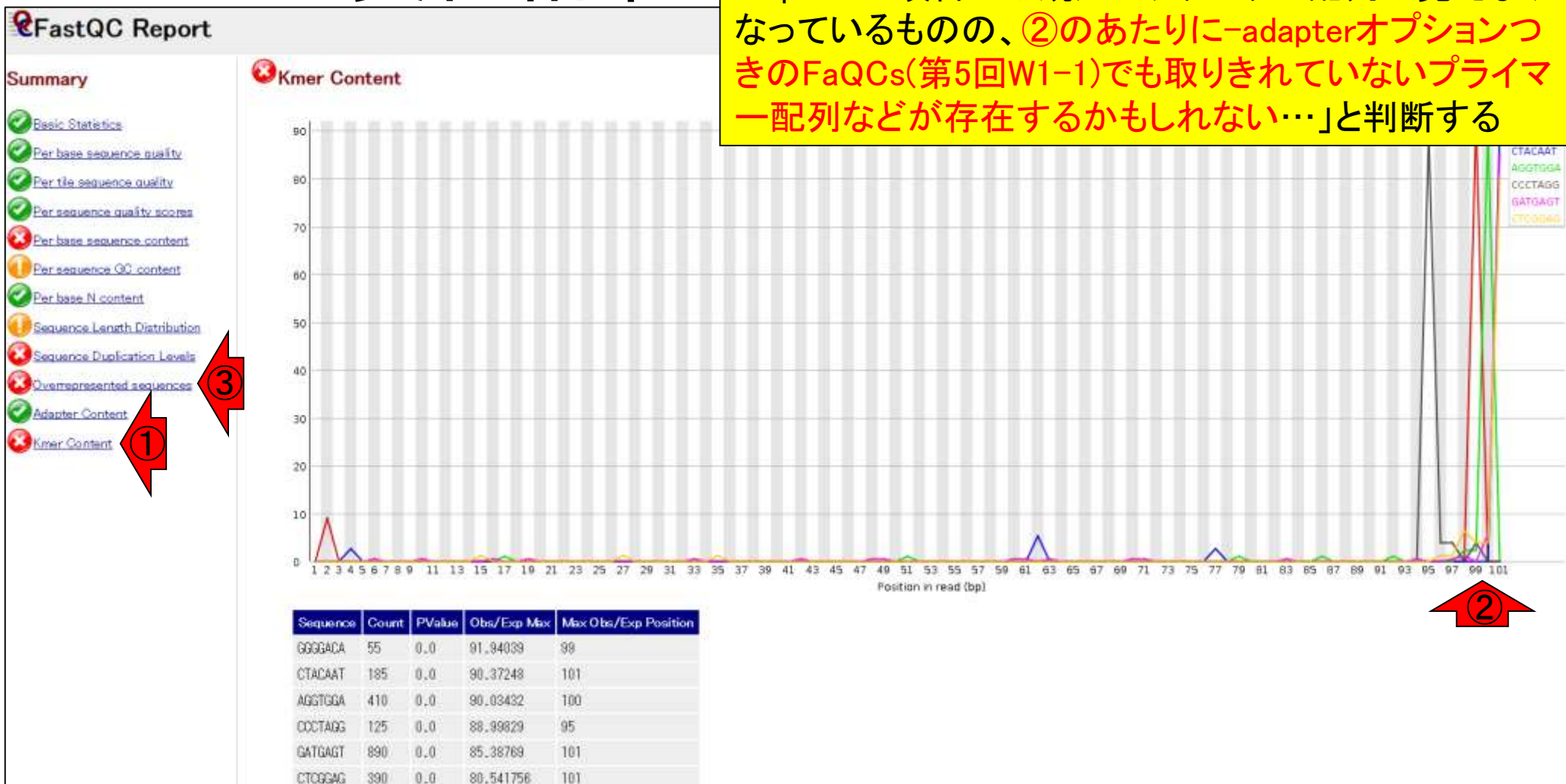
### Per base sequence quality

Produced by FastQC (version 0.11.4)



# FastQC実行結果f

①Kmer Content項目の全体像。②99-101塩基目あたりにピークが集中しているのを、「③Overrepresented sequences項目では既知のアダプター配列は見えないもの、②のあたりに-adapterオプション付きのFaQCs(第5回W1-1)でも取りきれていないプライマー配列などが存在するかもしれない…」と判断する



# 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)」

FastQC Report

Wed 22 Jun 2016  
data2.fq.gz

### 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

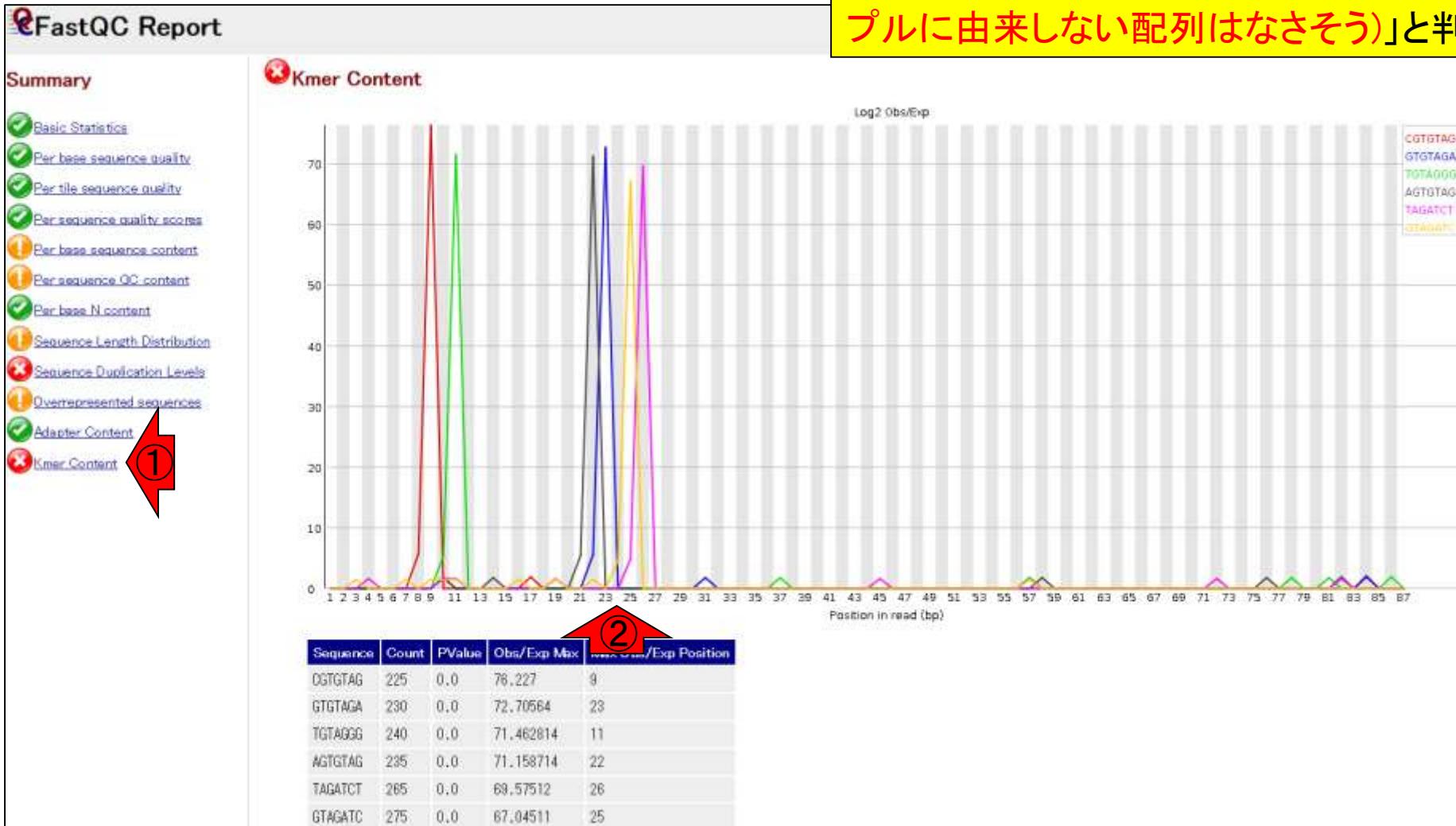
Measure	Value
Filename	data2.fq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	977202
Sequences flagged as poor quality	0
Sequence length	50-93
%GC	49

### Per base sequence quality

Produced by FastQC (version 0.11.4)

# FastQC実行結果r

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*ト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。①paired-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ていないのでエラーが出るとおられます。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

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あたりの数値は、第5回W5-2と同じなので安心

```
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%
```



①無事計算終了。②結果はRockhopper\_Resultsディレクトリに格納される(第5回W5-2)

# Rockhopper2おさらい

```
File Edit View Search Terminal Help 10:04
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_Results/summary.txt
Details of assembled transcripts written to file: Rockhopper_Results/transcripts.txt

FINISHED. ①

iu@bielinux[20160804] ls [10:03午前]
data1.fq.gz data2.fq.gz Rockhopper_Results ②
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

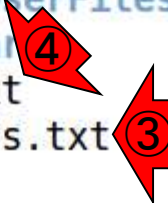
# Rockhopper2おさらい

```
File Edit View Search Terminal Help 10:04
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 intermedia
-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]
```



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で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

```
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_Results/summary.txt
Details of assembled transcripts written to file:  Rockhopper_Results/transcripts.txt
FINISHED.
iu@bielinux[20160804] [10:10午前]
```

# Tips: diff

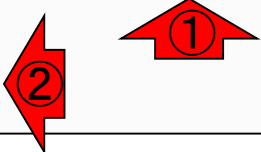
①Rockhopper2実行時にターミナル画面に出  
力されるものをresult.txtに保存。②diffコマンドは  
2つのファイルの差分を出力する。比較している2  
つのファイルが全く同じ場合には何も起こらない

## • Tips: diff

Rockhopper実行時に画面に出力されるものをresult.txtに保存してスクリップさせる。  
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
```





① Rockhopper2実行時にターミナル画面出力されるものを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。

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd [ 1:52午後 ]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l [ 1:52午後 ]
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 [ 1:52午後 ]
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] █ [ 1:52午後 ]
    
```



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。

```

File Edit View Search Terminal Help
iu@bielinux[20160804] pwd [ 1:52午後 ]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l [ 1:52午後 ]
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 [ 1:52午後 ]
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] ls -l Rockhopper_Results [ 1:52午後 ]
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] █ [ 1:52午後 ]
    
```





# Tips: diff

## • Tips: diff

Rockhopper実行時に画面上に出力されるものをresult.txtに保存してスクリプトを編集し、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、何も表示されず(全く同じ)ことを確認。

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

```

File Edit View Search Terminal Help
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 [ 1:52午後]
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] █ [ 1:52午後]
    
```

③Rockhopper2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き、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直下に見当たらないこと、intermediaryという名前が中間ファイル貯蔵庫っぽく響きだったから。真偽は不明



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

## 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き、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になっているのであろう。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
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
```





# 確認

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だから当たり前だが)アセンブリ実行にそれなりに時間がかかった。②diffの結果は不変

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き、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になっているのであろう。今には念を入れて、一旦

Rockhopper Result  
か、diff結果で

```
cd ~/Documents
pwd
ls
rm -rf Rockhopper
rm -f result.txt
ls -l
java -Xmx2000m
ls
ls -l Rockhopper
diff result.txt
```

```
iu@bielinux[~/Documents/srp017156/20160804]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls [ 2:16午後]
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] rm -rf Rockhopper_Results [ 2:16午後]
iu@bielinux[20160804] rm -f result.txt [ 2:16午後]
iu@bielinux[20160804] ls -l [ 2:16午後]
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 [ 2:17午後]
data1.fq.gz data2.fq.gz result.txt Rockhopper_Results
iu@bielinux[20160804] ls -l Rockhopper_Results [ 2:17午後]
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] [ 2:17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*ト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



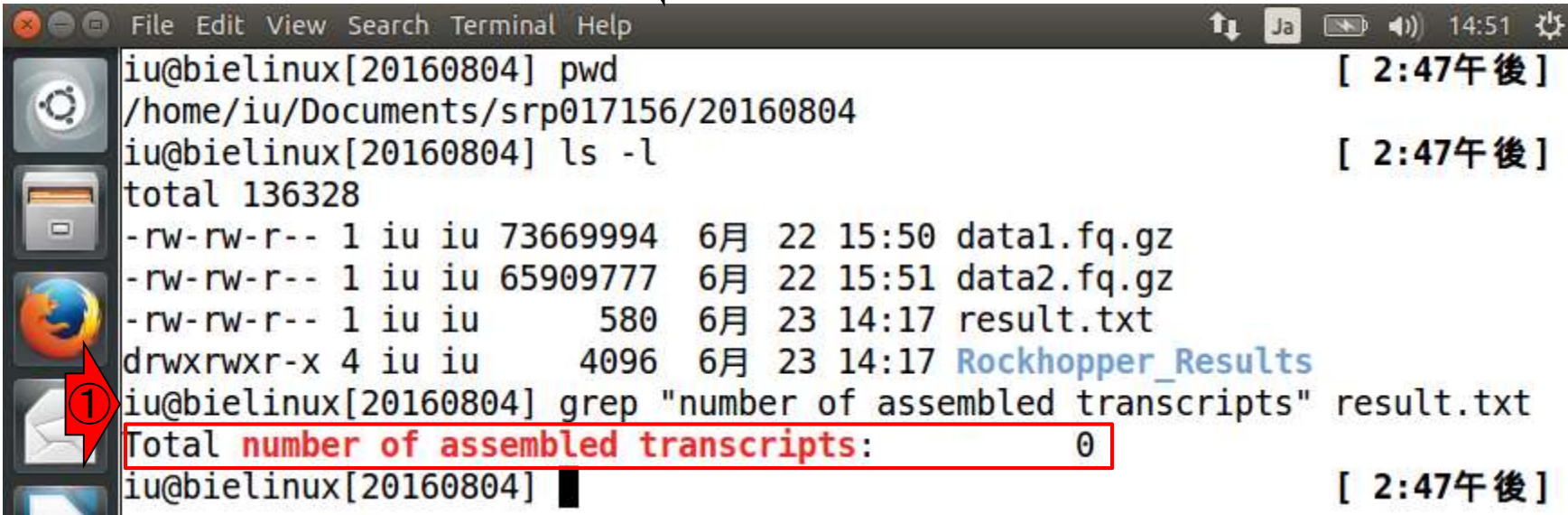
# 情報抽出(grep)

(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

## 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。どのような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
```



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)

## 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
```



# 情報抽出2(arep)

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

```

File Edit View Search 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.tx
t
iu@bielinux[20160804] java -Xmx2000m Rockhopper data2.fq.gz > result_r.tx
t
iu@bielinux[20160804] ls -l [ 3:07午後 ]
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 ]
    
```





# 情報抽出2(grep)

赤枠のgrep実行結果に注目!①入力ファイルは result\*.txtとしているので、②該当する3つのファイルの結果が返されている。③配列数は、forward側のみが1個(第5回W6-2)、reverse側のみが423個(第5回W6-4)、paired-endが0個(第5回W5-2)という結果。いずれも以前の結果と全く同じで安心

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.tx
```

```
iu@bielinux[20160804] ls -l
```

[ 3:07午後 ]

```
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

**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はアセンブリ、を繰り返し行う。**評価基準:** 配列数

## ■ *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のピークはなかった

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ト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

以前の経験から①paired-endでうまくいかなかった主な原因は②forward側にあると判断。③FastQC実行結果から、おそらく問題は5'末端ではなく3'末端のほうにあるだろうという思考回路のもとで、とりあえずforward側のみでトリミングからアセンブリまでの具体的な戦略を練る

## de novoトランスクリプト

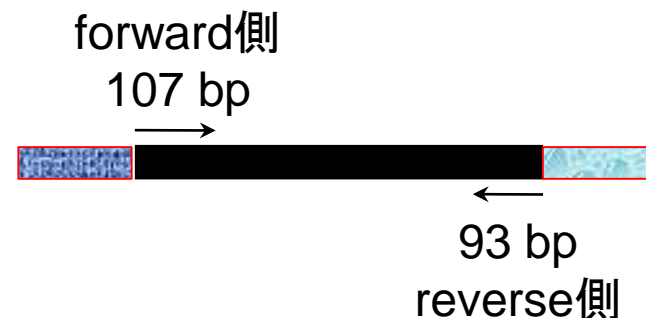
□ バクテリア用のアセンブラRockhopper2を実行して変な結果に遭遇した

- paired-end(data1.fq.gzとdata2.fq.gz): 0 transcript ①
- single-end (forward側のみ; data1.fq.gz): 1 transcript ②
- 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 ③
- 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

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。赤枠部分をコピー

## • 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塩基目以降をトリミングしたいので) 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塩基目以降をトリミングしたいので) 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
gunzip -c data1.fq.gz
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
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
① 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
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 5:12午後 ]
```



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

# 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塩基目以降をトリミングしたいので)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/sr
pwd
ls -l data1*
gunzip -c data1.fq.gz
gunzip -c data1.fq.gz
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
GATCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
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
GATCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 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変(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塩基目以降をトリミングしたいので) 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 trim1.fq.gz | head -n 4

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
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGTACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
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
GATCTGGGCTGTTCCCTTTCGACAATGGACCTTATCGCTCACTGTACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 5:12午後 ]

```





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、②(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…

# 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) 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/sr
pwd
ls -l data1*
gunzip -c data1.fq.gz
gunzip -c data1.fq.gz
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
GATCTGGGCTGTTCCCTTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJJHHAHHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -l 99 - | [ 5:12午後 ]
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
GATCTGGGCTGTTCCCTTTTCGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJJHHAHHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] █ [ 5:12午後 ]
```

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) 99塩基目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/s
pwd
ls -l data1*
gunzip -c data1.
gunzip -c data1.
gunzip -c trim1.
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 INFILE] [-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 INFILE] = 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*ト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

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に影響されるのかが気になる(結論としては独立)

## 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 99」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.gz
gunzip -c trim1.fq.gz | head -n 4
```



コピー実行結果。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 99」としていいのかわかりたい。

```
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
```

```

iu@bielinux[20160804] pwd [ 7:12午後 ]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1* [ 7:12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 [ 7:12午後 ]
@SRR616268.20 2291:6:1101:1720:2221 length=107
GATCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
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 ]
@SRR616268.20 2291:6:1101:1720:2221 length=107
TCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATGGT
ATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CFFFFHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDED>
CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 7:12午後 ]

```



# fastx-trimmer -l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塩基目以降をトリミングしたいので) 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
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
GATCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAACCTCCGAAA
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDDCCCCFFFF
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx-trimmer -l 99 - | [ 5:12午後 ]
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
GATCTGGGCTGTTCCCTTTGACAATGGACCTTATCGCTCACTGTCTGACTCCCGGAGTAAGATCGATG
GTATTCGGAGTTTATCTGAATTCAGTAAC
+SRR616268.20 2291:6:1101:1720:2221 length=107
CCCCFFFFHHHHHJJJJJJJJJJGHHIIJJJJJJJJJJJJJJJJJJJJJJHHIJJJJJHHAEHFFFDDE
D>CDDDDDDDBCCEDCDDCCDDCCDEEDD
iu@bielinux[20160804] [ 5:12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*ト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

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。赤枠を丸々コピーすると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

## 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塩基目という意味。
#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
```



# 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。

```

cd ~/Documents/sr
pwd
ls -l data1*
#-f 1: 残す最初(f
#-l 107: 残す最後
#つまり、トリムし
gunzip -c data1.
java -Xmx2000m R
grep "number of
#-f 1: 残す最初(f
#-l 104: 残す最後
#例えば元が104塩基
#FaQCs実行で元々1
gunzip -c data1.
java -Xmx2000m R
grep "number of

```

```

iu@bielinux[~/Documents/srp017156/20160804]
iu@bielinux[20160804] pwd [ 9:31午後]
/home/iu/Documents/srp017156/20160804
iu@bielinux[20160804] ls -l data1* [ 9:32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 | fastx_trimmer -f 1 -l 10
7 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_0
1_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] [ 9:33午後]
iu@bielinux[20160804] gunzip -c data1.fq.gz | fastx_trimmer -f 1 -l 10
4 - | gzip > trim1.fq.gz
iu@bielinux[20160804] java -Xmx2000m Rockhopper trim1.fq.gz > result_0
1_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] [ 9:34午後]

```



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
    
```





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

## 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、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

```

### -f 1 and -l 98
gunzip -c data1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_101.txt
grep "number of assembled transcripts" result_03_f_101.txt

### -f 1 and -l 99
gunzip -c data1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_098.txt
grep "number of assembled transcripts" result_03_f_098.txt

### -f 3 and -l 98
gunzip -c data1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_098.txt
grep "number of assembled transcripts" result_03_f_098.txt

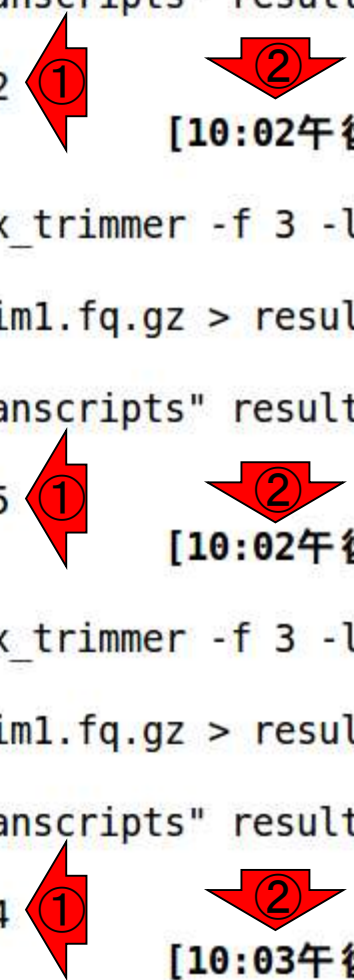
### -f 3 and -l 95
gunzip -c data1.fq.gz
java -Xmx2000m Rockhopper trim1.fq.gz > result_03_f_095.txt
grep "number of assembled transcripts" result_03_f_095.txt

```

```

iu@bielinux[20160804] grep "number of assembled transcripts" result_03_f_101.txt
Total number of assembled transcripts: 372
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
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
iu@bielinux[20160804]

```





# 結果の全体像を把握

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

```

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] █

```

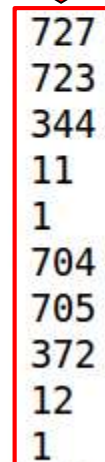
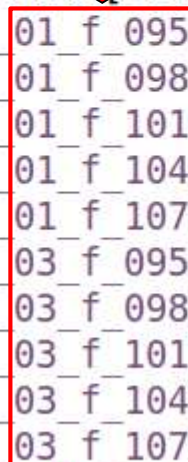


[11:33午前]

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

```
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*
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

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条件と一気にやった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。数が多け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のほうがこの場合は優先

```

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_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] █
    
```

forward側  
107 bp

93 bp  
reverse側

[11:33午前]

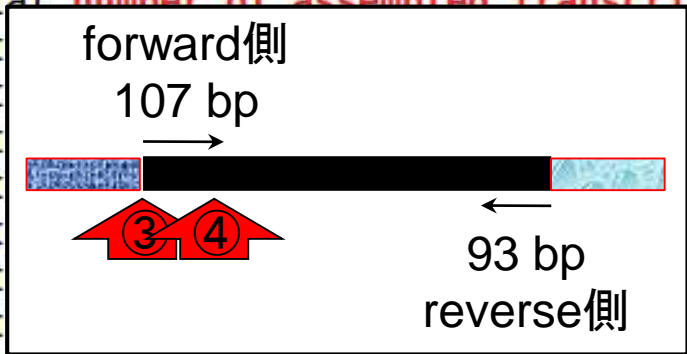
# 結果の全体像を把握

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

```

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]
    
```



- 727
- 723
- 344
- 11
- 1
- 704
- 705
- 372
- 12
- 1



[11:33午前]



# 結果の全体像を把握

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、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…など考える

```

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]
    
```



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
```

```
### -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 ###
```





# 様々なトリム条件3

コピー実行後に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さきほどコピー実行したものと、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っていれば①をやってもよい

```
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
-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]
```

[ 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)



# 結果の全体像を把握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)。実行結果と見比べるとわかりやすいでしょう

- 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なに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
```



```
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
iu@bielinux[20160804] █
```

[ 2:52午後 ]



# 結果の全体像を把握2

①2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]
    
```



# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ト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

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

```
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
① 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 ]
[ 2:52午後 ]
```



# 他の結果もチェック1

① Perfectly 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

```

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]
    
```



[ 2:55午後 ]



# 他の結果もチェック2

① Total 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

```

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]
    
```



[ 3:22午後 ]



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

# 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)上に結果を保存し、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、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いたって記述する場合は「バックslash」です(第4回W5-2やW9-8)。また、たい場合は、「Alt+U」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* トリム条件 配列数
> ~/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のメインタ  
 ーゲット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		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
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*ト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..

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

- 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(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 ###
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_099

### -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%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_099

### -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%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_099

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

コピー実行後の状態。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は...

```

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/tra
nscripts_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]
    
```





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

# 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  
果ファ

```

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/tra
nscripts_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]
    
```





赤枠部分がさきほど行った paired-endのアセンブリ結果

# 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/single-endに比べて劇的に増加していることがわかる。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_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%	183106	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



# 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





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...

# アセンブリ評価系...

## (Rで)塩基配列解析

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

- 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](#) | [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 2016/03/20)
- アセンブル | [トランスクリプトーム\(転写物\)用](#) **①** (last modified 2016/06/03)
- [マッピング](#) | [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\(Gaidatz\)](#)

### What's new?

このウェブページは、最新のソフトウェア(RとLinux)をWindowsにインストールする方法を説明しています。

## アセンブル | ト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](#)
- [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, 2016](#)
- [BinPacker](#): [Liu et al., PLoS Comput Biol., 2016](#)

### Review、ガイドライン、パイプライン系:

- 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, トランスクリプトーム, 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 | [S](#)
- アセンブル | [アセンブル | トランスクリプトーム\(転写物\)用](#) (last modified 2014/06/20)
- アセンブル | [ゲノム用](#) (last modified 2016/03/24)
- アセンブル | [トランスクリプトーム\(転写物\)用](#) (last modified 2016/06/03)
- マッピング | [マッピング | トランスクリプトーム\(転写物\)用](#) (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\(Gaidatz\)](#)

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 | ト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](#)
- [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, パイプライン系:

- 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

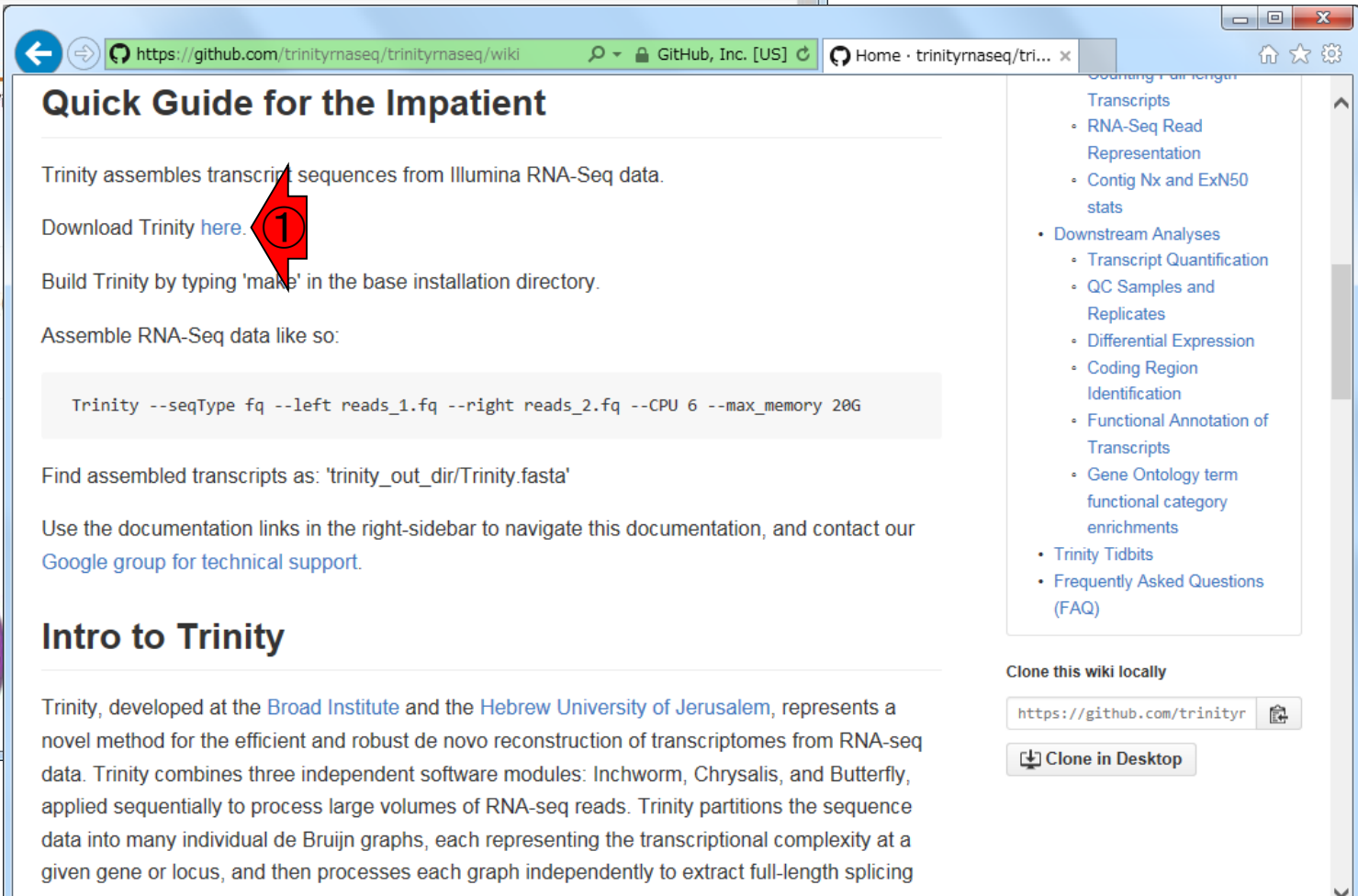
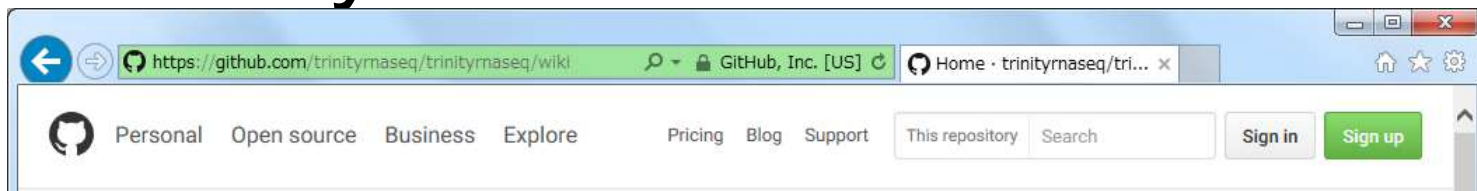
- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*トランスクリプトームアセンブリ
  - 事前準備、FastQC
  - Rockhopper2おさらい、情報抽出
  - 様々なトリム条件でRockhopper2を実行
    - トリミング、fastx-trimmer -f -l、様々なトリム条件
    - 様々な基準でアセンブリ結果を評価、ベストな条件でpaired-endアセンブリを実行
  - Trinity
    - 解凍、インストール、実行方法を調べてパスを通す、色々試しながら実行、apt-get
  - Bridger
    - 解凍してREADME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





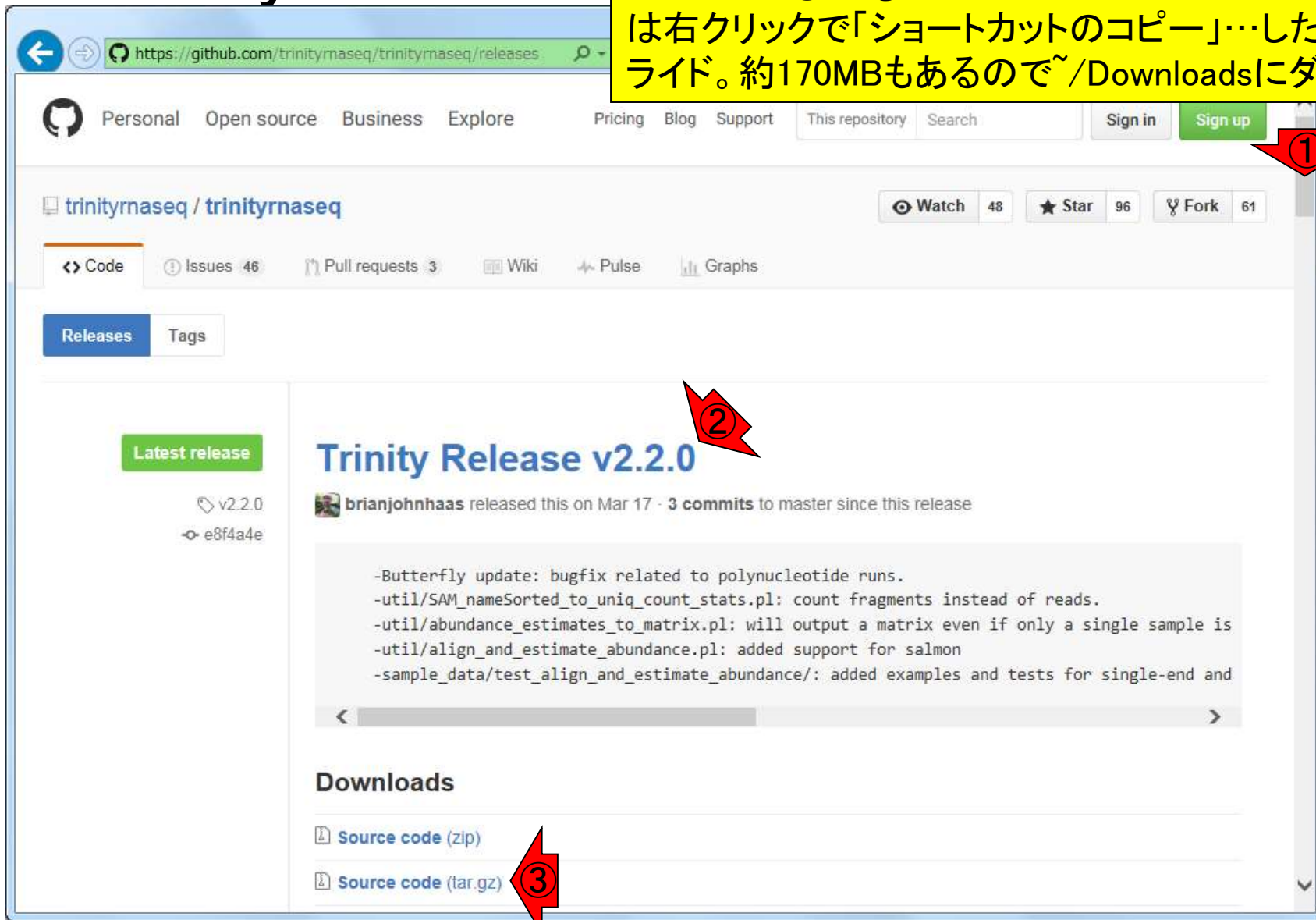
①ここからダウンロードできるようなのでクリック。スライドを見るだけ

# Trinity



# Trinity

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



wgetのところはコメントアウト(#)していません。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
```



- 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

```
File Edit View Search Terminal Help
iu@bielinux[Downloads] pwd [ 2:34午後]
/home/iu/Downloads
iu@bielinux[Downloads] ls -l v2* [ 2:34午後]
-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]
```

# 解凍後

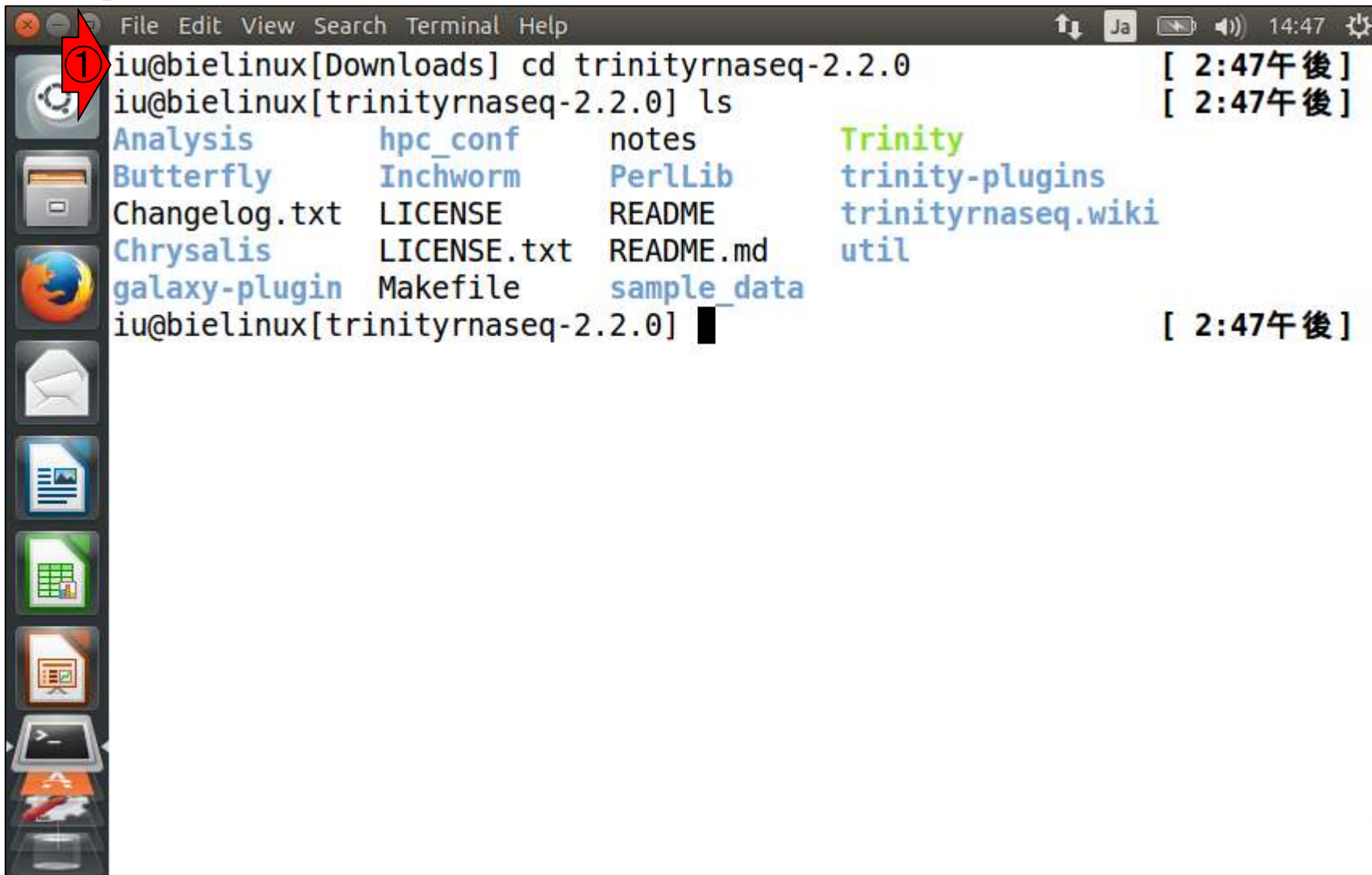
解凍終了後の状態。①赤下線部分の trinityrnaseq-2.2.0 というディレクトリが作成されていると解釈する。エラーは出てないっぽい

```
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



A terminal window showing the following commands and output:

```
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 ]
```

A red arrow with the number '1' points to the first command line.



# README

① READMEファイルがあったので、②moreで表示。赤下線のMakefileを見つけたので、「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

```
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] █ [ 2:50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*ト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



# Install

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

Home

Thisaru Guruge edited this page on Apr 4 - 30 revisions

## RNA-Seq De novo Assembly Using Trinity

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 the GitHub Wiki page for Trinity. The page title is "Installing Trinity" by Brian Haas. The main content area contains the following text:

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:

The page also features a sidebar with a "Pages" section containing a list of links: Trinity Wiki Home, Installing Trinity (with sub-links for Trinity Computing Requirements, Accessing Trinity on Publicly Available Compute Resources, and Running Trinity with sub-links for Genome Guided Trinity Transcriptome Assembly and Genome Annotation), Monitoring Progress of a Trinity Run, and Output of Trinity Assembly.

Red arrows labeled ① and ② point to the "make" command and the prerequisite text, respectively.

# Install

```
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

① iu@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] █

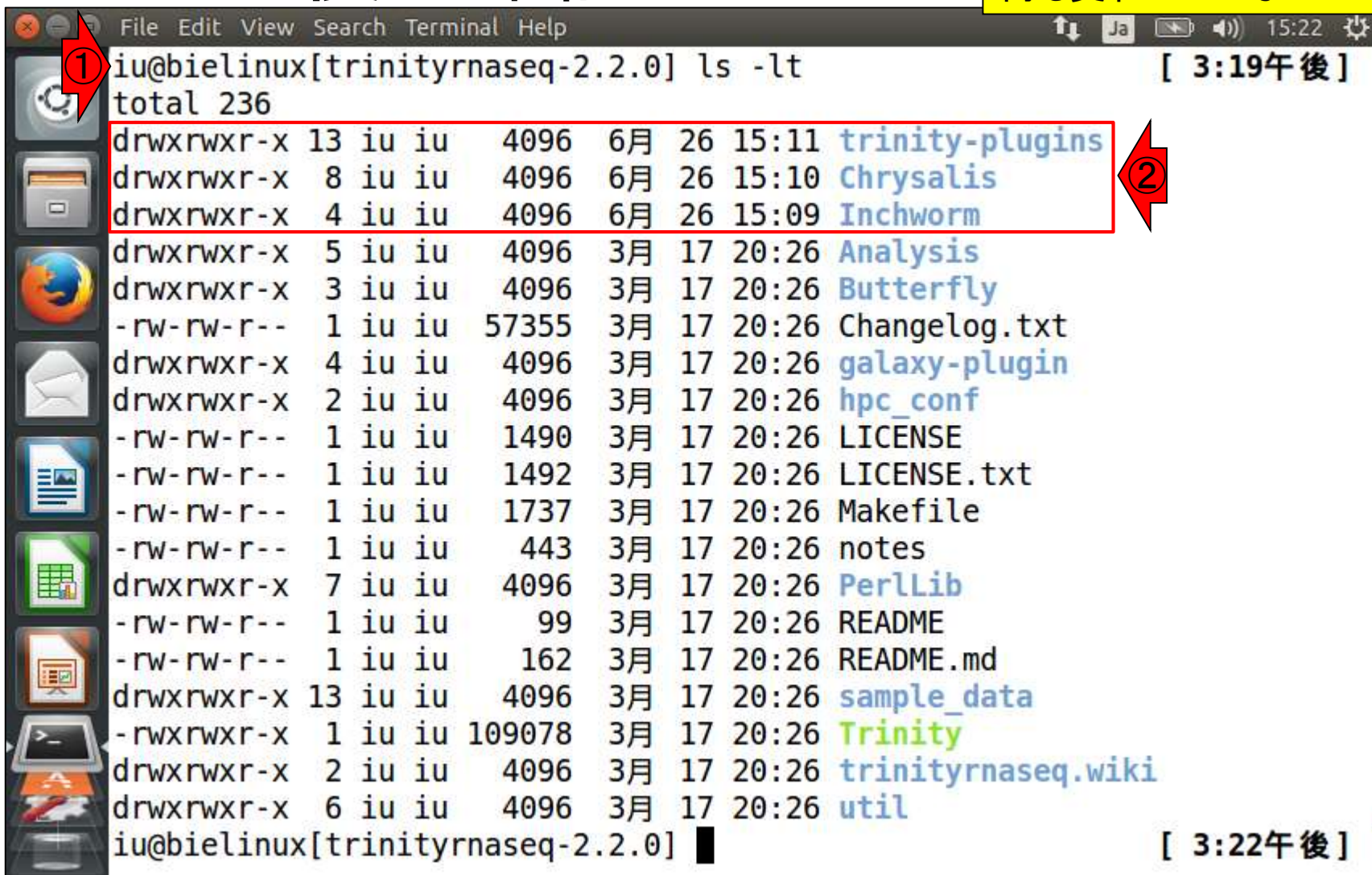
```

[ 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…



```
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]
```

# 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 the GitHub Wiki page for Trinity. The browser address bar shows the URL: `https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing`. The page title is "Installing Trinity".

Annotations include:

- A red arrow labeled "1" points to the `make` command in a code block.
- A red arrow labeled "2" points to the text: "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."
- A red arrow labeled "3" points to the same text block.

The page content includes:

## 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*ト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





# 実行方法を調べる

The screenshot shows a web browser window displaying the GitHub Wiki page for Trinity. The browser's address bar shows the URL <https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing>. The page title is "Installing Trinity" and it was last edited by Brian Haas on Mar 22 with 7 revisions. The main heading is "Installing Trinity". Below the heading, the text reads: "After downloading the software to a Linux server, simply type". A code block contains the command `make`. The text continues: "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." The text concludes: "Afterwards, you may want to build the additional plugin components that provide support for downstream analyses in which case you would then type:". On the right side of the page, there is a "Pages" sidebar with 24 items. A red arrow with the number 1 points to the "Running Trinity" item in the sidebar.

Installing Trinity

Brian Haas edited this page on Mar 22 · 7 revisions

## 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:

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

①下にスクロールして行って、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

The screenshot shows a web browser window displaying the GitHub Wiki page for 'Running Trinity'. The browser's address bar shows the URL 'https://github.com/trinityrnaseq/trinityrnaseq/wiki/Running Trinity'. The page header includes navigation links for 'Personal', 'Open source', 'Business', and 'Explore', along with 'Pricing', 'Blog', and 'Support'. There are buttons for 'Sign in' and 'Sign up', with a red arrow pointing to the 'Sign up' button. The repository name 'trinityrnaseq / trinityrnaseq' is displayed, along with statistics for 'Watch' (48), 'Star' (96), and 'Fork' (61). The 'Wiki' tab is selected, and the page title is 'Running Trinity'. Below the title, it says 'Ruwangika Gunawardana edited this page on May 7 · 10 revisions'. The main content area has the heading 'Running Trinity' and the text 'Trinity is run via the script: 'Trinity' found in the base installation directory. Usage info is as follows:'. Below this text is a code block containing a terminal output showing a progress bar for Trinity assembly. On the right side, there is a 'Pages' section with 24 pages listed, including 'Trinity Wiki Home', 'Installing Trinity', 'Running Trinity', 'Monitoring Progress of a Trinity Run', and 'Output of Trinity Assembly'.

# 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

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:

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

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

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

```

File Edit View Search Terminal Help
iu@bielinux[trinityrnaseq-2.2.0]
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
-rw-rw-r-x 1 iu iu 109078 3月 17 20:26 Trinity
drwxr-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 ]

# 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

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のやり方は間違いです。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)

```
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            Tr
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] where Trinity [ 1:49午後]
Trinity not found ②
iu@bielinux[trinityrnaseq-2.2.0] cp Trinity ~/bin [ 1:49午後]
iu@bielinux[trinityrnaseq-2.2.0] where Trinity [ 1:49午後]
/home/iu/bin/Trinity ③
/home/iu/bin/Trinity
iu@bielinux[trinityrnaseq-2.2.0] [ 1:49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*ト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

赤枠部分をコピー。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 (スライド 109)  
とあえず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
```

# 色々試しながら実行1

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

- 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モジュールがないので、Trinity --seqType fq --max_memory 2G --CPU 2 --left data1.fq.gz --right data2.fq.gzを実行する。
### COMMON.pm問題は解決。Trinity --seqType fq --max_memory 2G --CPU 2 --left data1.fq.gz --right data2.fq.gzを実行する。
### 「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] █
```



# 色々試しながら実行1

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 (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  
Trinity --seqType fq --max  
--left data1.fq.gz
```

```
### COMMON.pm問題は解決。か  
~/Downloads/trinityrnaseq-  
--seqType fq --max  
--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[trinityrnaseq-2.2.0] cd ~/Documents/srp017156/2  
iu@bielinux[20160804] pwd [ 4:38午後]  
/home/iu/Documents/srp017156/20160804  
iu@bielinux[20160804] ls -l data* [ 4:38午後]  
-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  
- ###  
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

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

- 色々試しながら実行1 (スライド 109)  
とくにあえず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
```



```
cd ~/Documents/srp017156/20160804
[ 4:38午後]
160804
a*
[ 4:38午後]
月 22 15:50 data1.fq.gz
月 22 15:51 data2.fq.gz
COMMON.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モジュールの問題は解決。か
```

```
Trinity --seqType fq --max_memory 2G --CPU 2 \  
--left data1.fq.gz --right data2.fq.gz
```

```
### COMMON.pm問題は解決。か
```

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



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



- 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モジュールがインストールされていない
```

```
Trinity --seqType fq --maxMemory 10G
```

```
--left data1.fq.gz
```

```
--right data2.fq.gz
```

```
### COMMON.pm問題は解決。が
```

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

```
--seqType fq --maxMemory 10G
```

```
--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
```

```

rp017156/20160804/trinity_out_dir/chrysalis
-----
----- Trinity Phase 1: Clustering of RNA-Seq Reads -----
-----
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
    
```

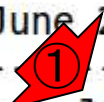
# 色々試しながら実行1

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 (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のTrinityを実行。

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
### COMMON.pmというPerlモジュールの問題
Trinity --seqType fq --maxDepth 1000000000 --left data1.fq.gz --right data2.fq.gz
### COMMON.pm問題は解決。か
~/Downloads/trinityrnaseq-2.2.0/trinityrnaseq-2.2.0/trinityrnaseq-2.2.0/trinityrnaseq-2.2.0
--seqType fq --maxDepth 1000000000 --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
```

```
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 >> 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: 17:12:30 CMD: touch left.fa.ok right.fa.ok
Monday, June 27, 2016: 17:12:30 CMD: cat left.fa right.fa > both.fasta
Monday, June 27, 2016: 17:12:44 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
```





# 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モジュールをインストール
```

```
Trinity --seqType fq --maxMemory 1000000000
```

```
--left data1.fq.gz --right data2.fq.gz
```

```
### COMMON.pm問題は解決。Trinityを実行
```

```
~/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
```

```
### 「CTRL + C」で脱出し、
```

```
ls -lt | 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
^C
Trinity run failed. Must investigate error above.
iu@bielinux[20160804]
```



[ 5:30午後 ]



# 強制終了時の注意

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、その後うまくコマンドを打っても失敗する場合があります(私はこれで何度かハマった経験があります)

- 色々試しながら実行1 (スライド 109)  
とてあえず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モジュールの問題
Trinity --seqType fq --maxReadLength 1000000000 --left data1.fq.gz --right data2.fq.gz
### COMMON.pm問題は解決。か
~/Downloads/trinityrnaseq-2.2.0/trinity-plugins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 273227412 --canonical both.fa
### 「CTRL + C」で脱出し、
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount
```

```
----- 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] ls -lt | head
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]
```

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モジュールの問題
Trinity --seqType fq --maxMemory 1000000000 --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
```

```
File Edit View Search Terminal Help
* 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] 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] [ 6:03午後 ]
```




# 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をコピー実行。約85分



# 途中経過1

①コピー実行から約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

- 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
```

```
~/Downloads/trinityrnaseq-2.2.0
--seqType fq --max_memory 1G
--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
```

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

```
File Edit View Search Terminal Help
-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:ParallelGCTh
reads=2 -jar /home/iu/Downloads/trinityrnaseq-2.2.0/util/support_s
cripts/ExitTester.jar 0
Monday, June 27, 2016: 18:22:34 CMD: java -Xmx64m -XX:ParallelGCTh
reads=2 -jar /home/iu/Downloads/trinityrnaseq-2.2.0/util/support_s
cripts/ExitTester.jar 1
Monday, June 27, 2016: 18:22:34 CMD: mkdir -p /home/iu/Documents/s
rp017156/20160804/trinity_out_dir
Monday, June 27, 2016: 18:22:34 CMD: mkdir -p /home/iu/Documents/s
rp017156/20160804/trinity_out_dir/chrysalis
```

# 途中経過2

①コピー実行から約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

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

```
### 確認 ###
```

```
ls -lt | head
```

```
cd trinity_out_dir
```

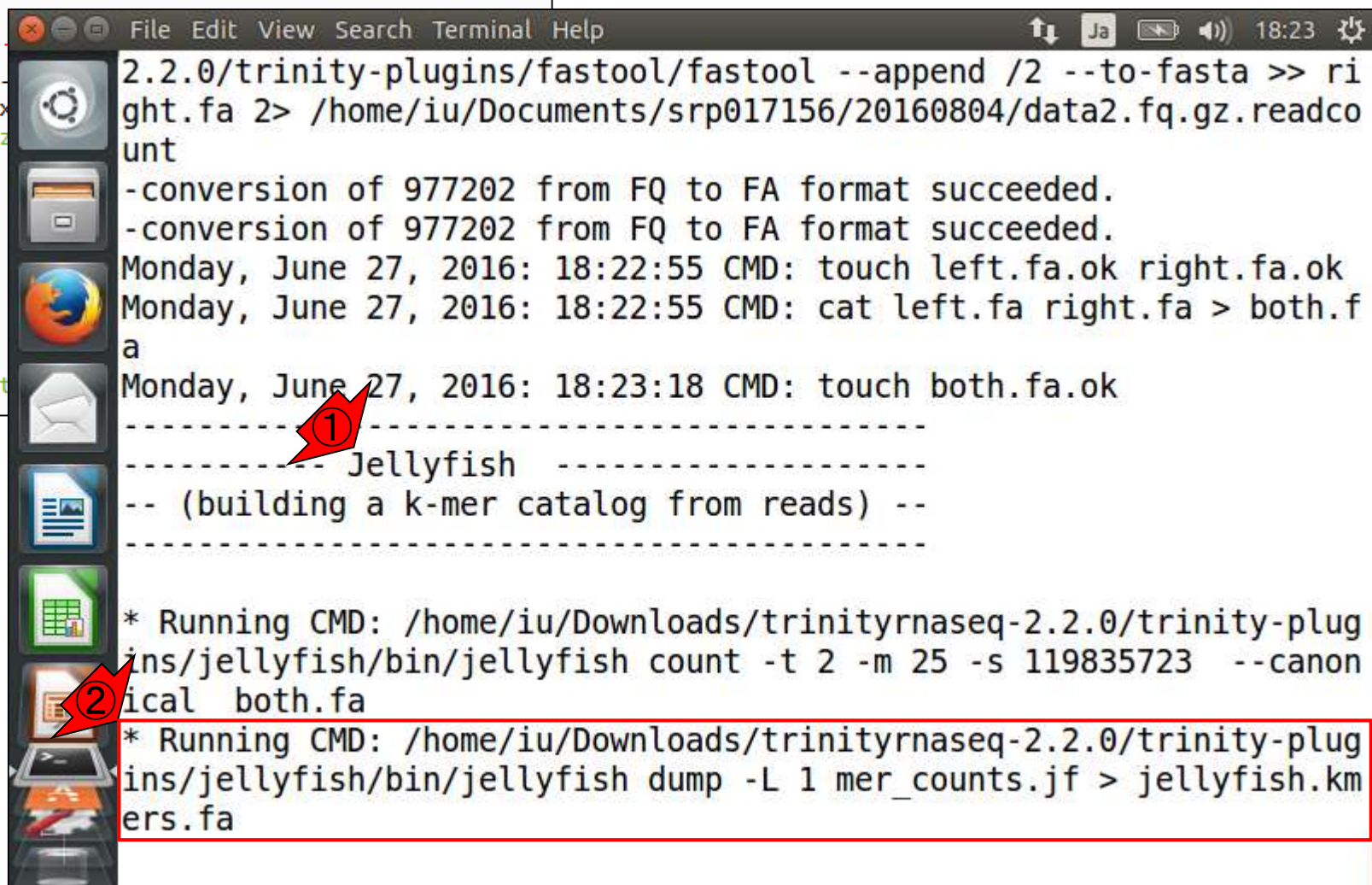
```
pwd
```

```
ls
```

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

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

```
cp Trinity.fasta ../Trinit
```



```
File Edit View Search Terminal Help 18:23
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: 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.f
a
Monday, June 27, 2016: 18:23:18 CMD: touch both.fa.ok
-----
----- 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 119835723 --canon
ical both.fa
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish dump -L 1 mer_counts.jf > jellyfish.km
ers.fa
```



# 途中経過3

確かコピー実行から約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



- 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
```

```
~/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
```

```
iu@bielinux[~/Documents/srp017156/20160804]
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish count -t 2 -m 25 -s 119835723 --canon
ical both.fa
* Running CMD: /home/iu/Downloads/trinityrnaseq-2.2.0/trinity-plug
ins/jellyfish/bin/jellyfish dump -L 1 mer_counts.jf > jellyfish.km
ers.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 -
-monitor 1 --DS --num_threads 2 --PARALLEL_IWORM > /home/iu/D
ocuments/srp017156/20160804/trinity_out_dir/inchworm.K25.L25.DS.fa
.tmp
```



# 途中経過4

①(18:22頃スタートなので)確かコピペ実行から約15分後の状態。②のところが100%になるまで延々と続く。19:27頃に②が30%に達したので、そこまでが1時間強。そこから先は比較的サクサクパーセンテージが上がっていく。③Number of Commandsの数値(2307)はヒトによって異なるらしい

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
```

```
File Edit View Search Terminal Help
-rsds_list_file partitioned_reads.files.list --CPU 1 --max_memory
1G --seqType fa --trinity_complete --full_cleanup > recursive_t
rinity.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/trinityrna
seq-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 ../Trinity
```

```

File Edit View Search Terminal Help
*** 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/supp
ort_scripts/partitioned_trinity_aggregator.pl TRINITY_DN > Trinity
.fasta.tmp

#####
#
Butterfly assemblies are written to /home/iu/Documents/srp017156/2
0160804/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
```

```
Help ↑ ↓ Ja 🔊 19:48 ⚙️
#####
e written to /home/iu/Documents/srp017156/2
/Trinity.fasta
#####
```

```
iu@bielinux[20160804] [ 7:47午後 ]
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] [ 7:48午後 ]
```



# 確認

赤枠内のコードを一気にコピーした結果。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…

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

```
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 ../Trinity1.fasta
```

```
iu@bielinux[20160804] cd trinity_out_dir [ 9:06午後 ]
iu@bielinux[trinity_out_dir] pwd [ 9:06午後 ]
/home/iu/Documents/srp017156/20160804/trinity_out_dir
iu@bielinux[trinity_out_dir] ls [ 9:06午後 ]
both.fa partitioned_reads.files.list
both.fa.ok partitioned_reads.files.list.ok
both.fa.read_count read_partitions
chrysalis recursive_trinity.cmds
inchworm.K25.L25.DS.fa recursive_trinity.cmds.completed
inchworm.K25.L25.DS.fa.finished recursive_trinity.cmds.ok
inchworm.kmer_count right.fa.ok
jellyfish.kmers.fa Trinity.fasta
jellyfish.kmers.fa.histo Trinity.timing
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] [ 9:06午後 ]
```

# 確認

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)

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(2,6

```
cd ~/Documents/srp017156/20160804
pwd
ls -l data*
```

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

```
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
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
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 参考

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実行時に追加する必要があります。ここはコピー実行しないで！

## 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3,081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
### 計算途中じゃなくても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
```





# 色々試しながら実行3 参考

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をやらなないといけないのだと学習します

## 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
### 計算途中じゃなくても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
```



# 色々試しながら実行3 参考

コピー実行結果。①配列数は3,090、②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
```

```
### 計算途中じゃなくてもTrinity
### これをやらずに実行する
```

```
ls -lt | head
rm -rf trinity_out_dir
ls -l *readcount
rm -f *readcount
```

```
### SRR616268の実験情報SRX
### orientation情報も入れ
```

```
~/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 ../Trinity2.fasta
```

```
File Edit View Search Terminal Help
-iw-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 partitioned_reads.files.list
both.fa.ok partitioned_reads.files.list.ok
both.fa.read_count read_partitions
chrysalis recursive_trinity.cmds
inchworm.K25.L25.fa recursive_trinity.cmds.completed
inchworm.K25.L25.fa.finished recursive_trinity.cmds.ok
inchworm.kmer_count right.fa.ok
jellyfish.kmers.fa Trinity.fasta
jellyfish.kmers.fa.histo Trinity.timing
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*ト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

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  
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
```

```
iu@bielinux[trinity_out_dir] cd [ 3:49午後 ]  
iu@bielinux[iu] pwd [ 3:49午後 ]  
/home/iu  
iu@bielinux[iu] where Trinity [ 3:49午後 ]  
/home/iu/bin/Trinity  
/home/iu/bin/Trinity  
iu@bielinux[iu] ~/Downloads/trinityrnaseq-2.2.0/Trinity --ve [ 3:49午後 ]  
rsion  
Trinity version: v2.2.0  
-currently using the latest production release of Trinity.  
iu@bielinux[iu] [ 3:49午後 ]
```

# apt-getでインストール 参

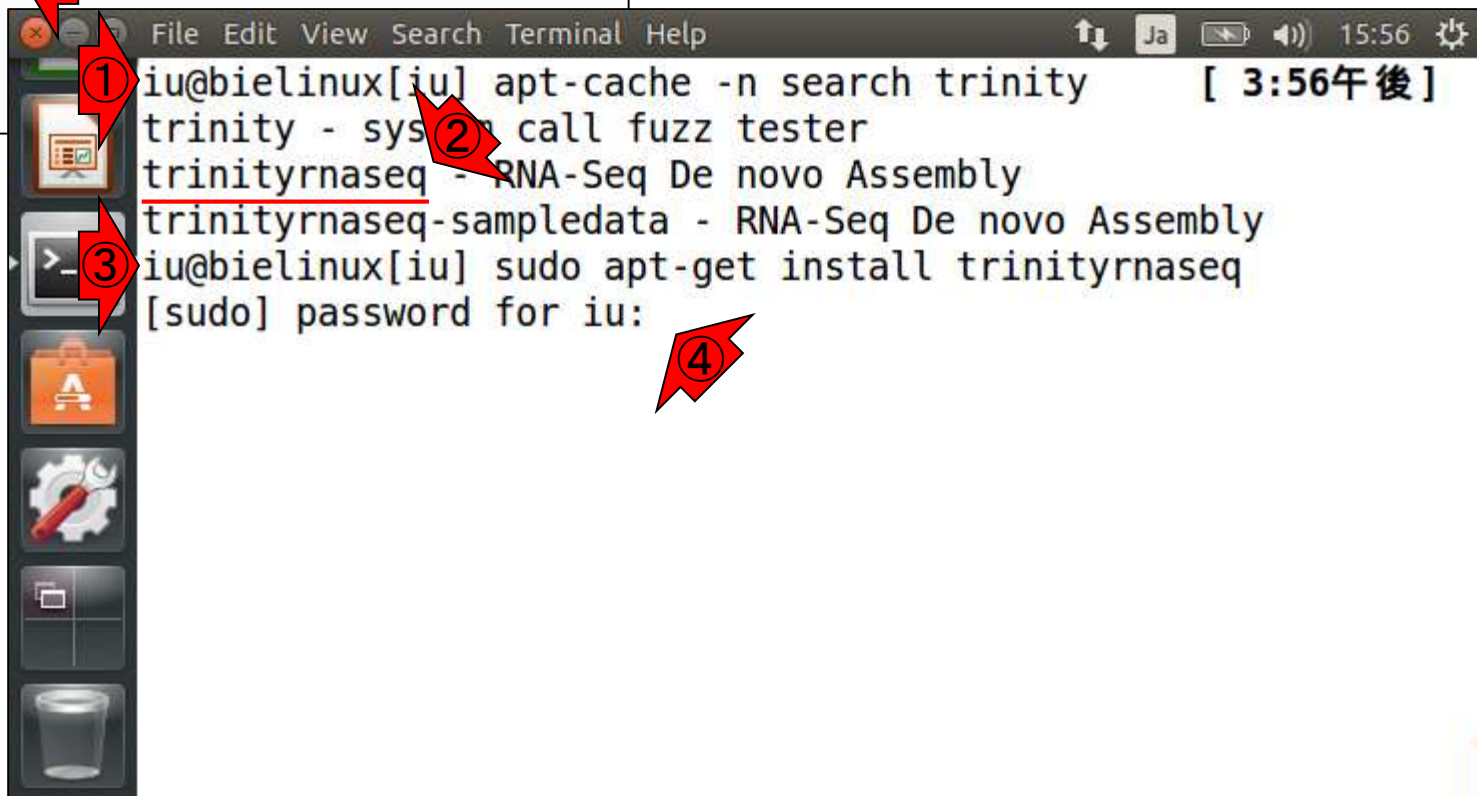
①trinityというキーワードを含むソフトウェア名をリストアップ。②正式名称がtrinityrnaseqであることを確認し、③apt-getでインストールを実行。④rootのパスワードはpass1409。見るだけ

- 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-2.8.0/Trinity --version
```

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

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



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 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  
Recommended packages:  
  med-config  
The following NEW packages will be installed:  
  berkeley-express jaligner libbamtools2.4.0 libcommons-collections4-java  
  libgetopt-java libjai-core-java libjellyfish-2.0-2 libjung-java paraflly rsem  
  transdecoder trimmomatic trinityrnaseq  
The following packages will be upgraded:  
  jellyfish  
1 upgraded, 13 newly installed, 0 to remove and 142 not upgraded.  
Need to get 7,744 kB of archives.  
After this operation, 21.6 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```





# apt-getでインストール

(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 …」などとも出ることもある。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  
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trinityrna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
Setting up libbamtools2.4.0 (2.4.0+dfsg-1biolinux1) ...
Setting up berkeley-express (1.5.1-1biolinux1) ...
Setting up jaligner (1.0+dfsg-1biolinux1) ...
Setting up libjellyfish-2.0-2 (2.2.3-2biolinux2) ...
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 (1.1.4-3ubuntu1) ...
Setting up libjung-java (2.0.1-1biolinux1) ...
Setting up paraflly (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] [ 4:01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
```

```
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 (1.1.4-3ubuntu1) ...
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
/home/iu/bin/Trinity
/home/iu/bin/Trinity
iu@bielinux[iu] [ 4:08午後 ]
```



# apt-getでインストール 参考

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

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
```

```
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/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@bielinux[iu] [ 4:09午後 ]
```



# apt-getでインストール 参考

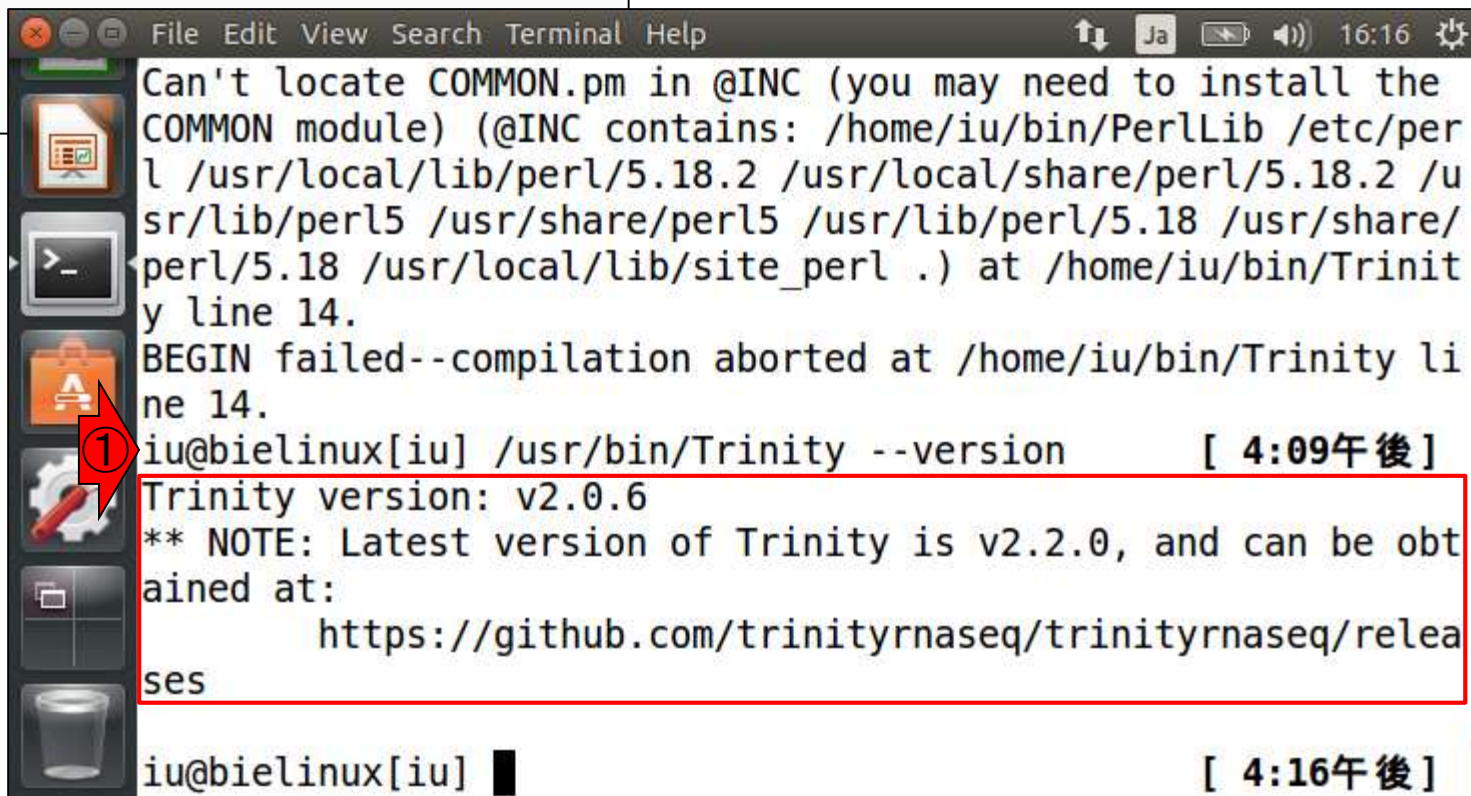
①apt-get経由でインストールしたTrinityは、ver. 2.0.6である。最新版(ver. 2.2.0)がインストールされない現象は、sra-toolkitのときと同じ(第7回W4-3; 2016.08.03スライド153)。一般論としては、最新版だとうまく動かないこともある(2016.08.12追加)

• 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
```



# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ト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, トランスクリプトーム, 正規化, 発現量  
(last modified 2016/06/03, since 2011)

### What's new?

このウェブページは、  
Linux/Windows  
環境で動作する  
ソフトウェアの  
インストール方法  
を解説しています。

- 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 | [トランスクリプトーム\(転写物\)用](#) (last modified 2016/03/24)
- マッピング | [について](#) (last modified 2016/04/07)
- マッピング | [basic aligner](#) (last modified 2014/08/07)
- マッピング | [splice-aware aligner](#) (last modified 2014/08/07)
- 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\(Gaidatz\)](#)

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-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](#)
- EBARDdenovo: [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](#)
- ④ 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](#)
- ① BinPacker: [Liu et al., PLoS Comput Biol., 2016](#)

Review, ガイドライン, パイプライン系:

- 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

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

• [Bridger: Chang et al., Genome Biol., 2015](#)

①  
講義では Bridger\_r2014-12-01.tar.gz (約11MB)を~/Downloadsにダウンロード済  
ショートカットのURLは「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」に変更した。

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

sourceforge

Search Browse Enterprise Blog Deals Help Create

SOLUTION CENTERS Go Parallel Resources Newsletters Cloud Storage Providers Business VoIP Providers

Get Google Chrome

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

Report a problem with ad content

Home / Browse / RNA-Seq Assembly / Files

### RNA-Seq Assembly

Brought to you by: zchang

Summary Files Reviews Support Wiki Code Tickets Discussion

Looking for the latest version? [Download Bridger\\_r2014-12-01.tar.gz \(11.1 MB\)](#)

Home

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

Free Project Management

Most Free PM Solution Aren't Free. Bitrix24 Free PM Is 100% Free.

Report a problem with ad content

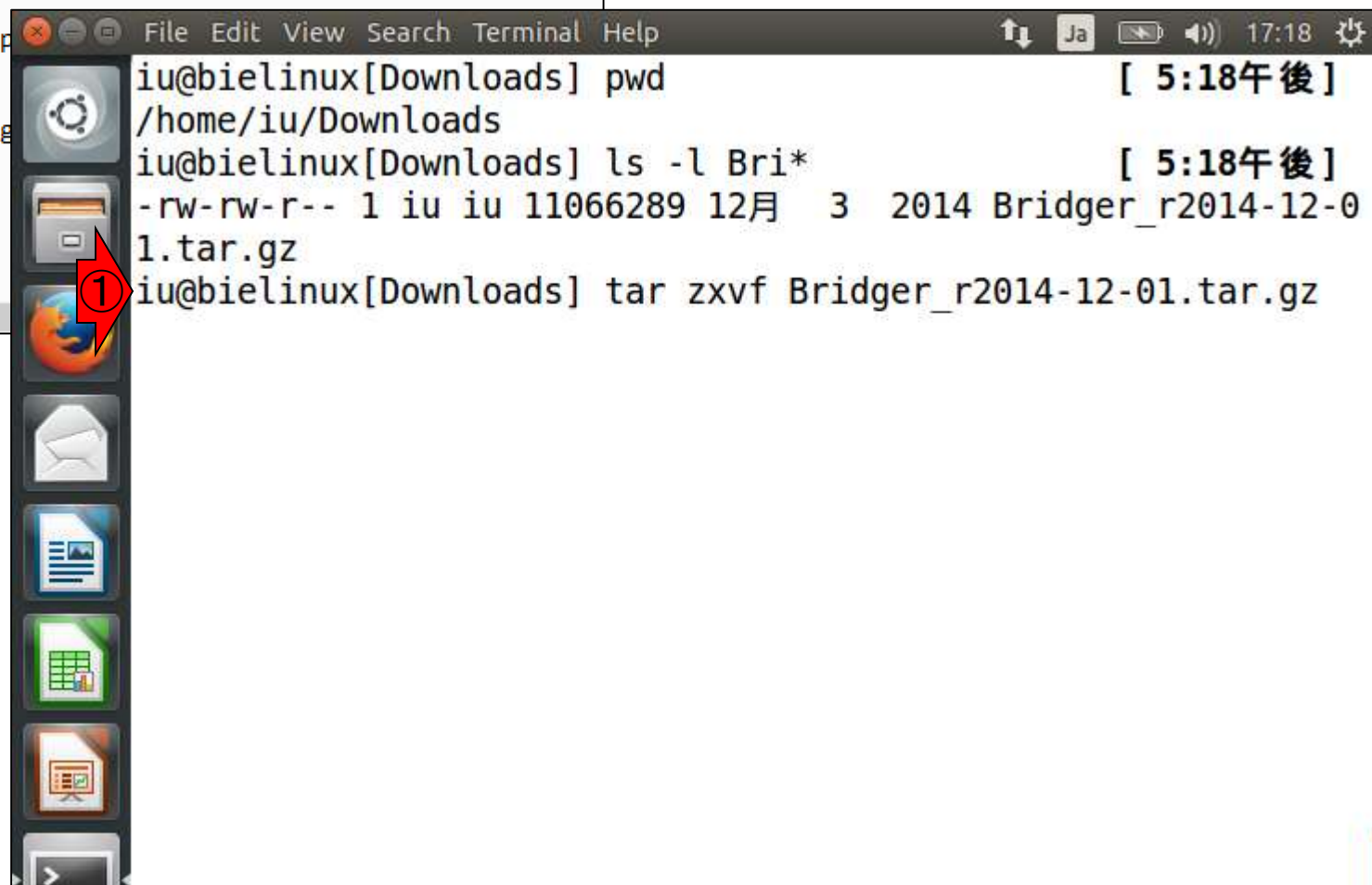
Recommended Projects

# 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/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
```



```
File Edit View Search Terminal Help
iu@bielinux[Downloads] pwd
[ 5:18午後 ]
/home/iu/Downloads
iu@bielinux[Downloads] ls -l Bri*
[ 5:18午後 ]
-rw-rw-r-- 1 iu iu 11066289 12月 3 2014 Bridger_r2014-12-0
1.tar.gz
iu@bielinux[Downloads] tar zxvf Bridger_r2014-12-01.tar.gz
```

# Bridger

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

• [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
```

```
File Edit View Search Terminal Help
Bridger_r2014-12-01/Makefile.am
Bridger_r2014-12-01/configure.ac
Bridger_r2014-12-01/boost.m4
Bridger_r2014-12-01/Makefile
Bridger_r2014-12-01/libtool
Bridger_r2014-12-01/stamp-h1
Bridger_r2014-12-01/ChangeLog
Bridger_r2014-12-01/README
Bridger_r2014-12-01/Bridger.pl
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 Makefile.in
AUTHORS config.h INSTALL perllib
autom4te.cache config.h.in libtool plugins
ax_boost_base.m4 config.log LICENCE README
boost.m4 config.status m4 sample_test
Bridger.pl configure Makefile src
build-aux configure.ac Makefile.am stamp-h1
iu@bielinux[Bridger_r2014-12-01] more INSTALL [ 5:36午後]
```





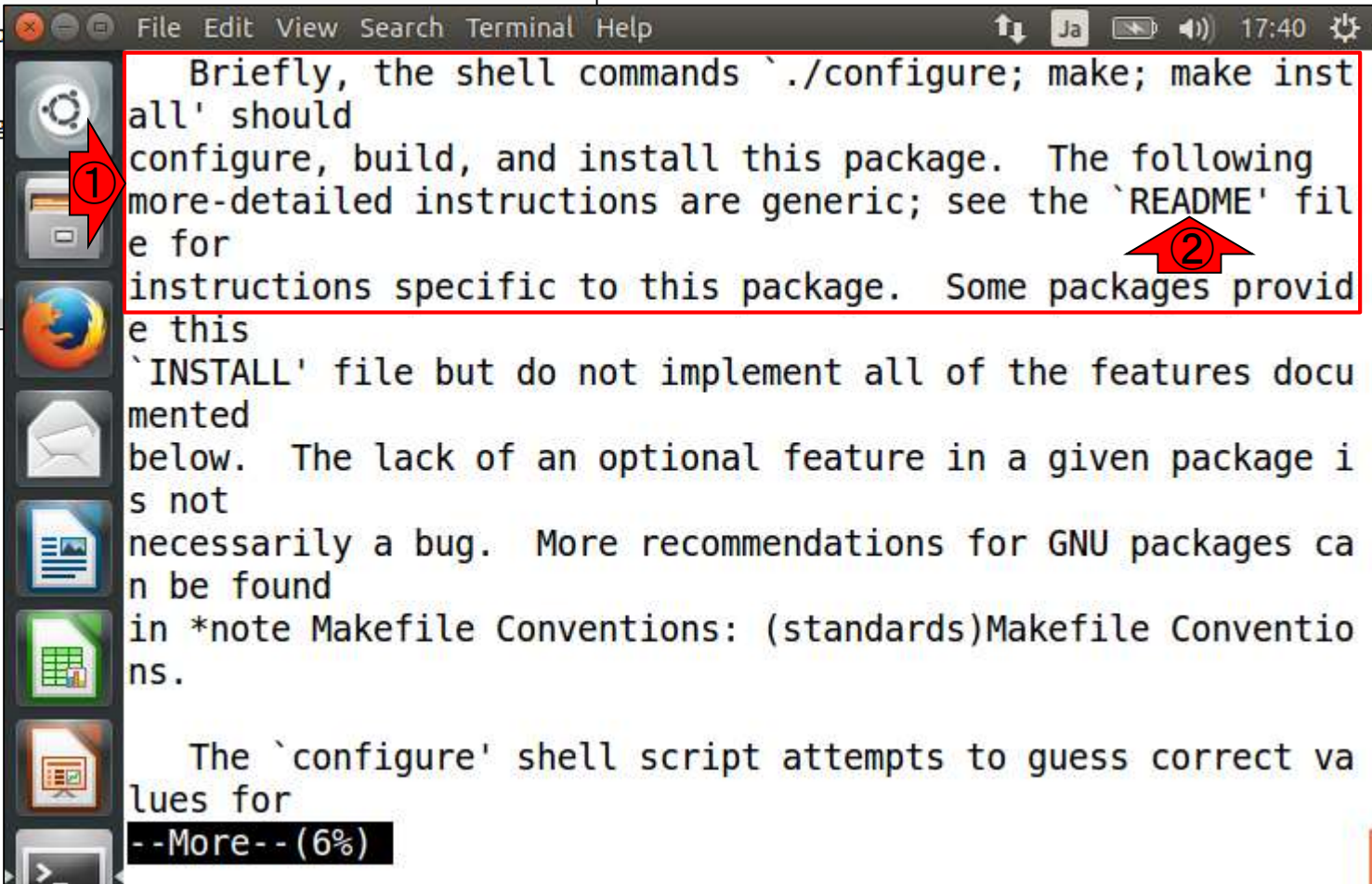
# more INSTALL

①のあたりを眺めて、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

• [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
```



# less README

- [Bridger: Chang et al., Genome Biol., 2015](#)

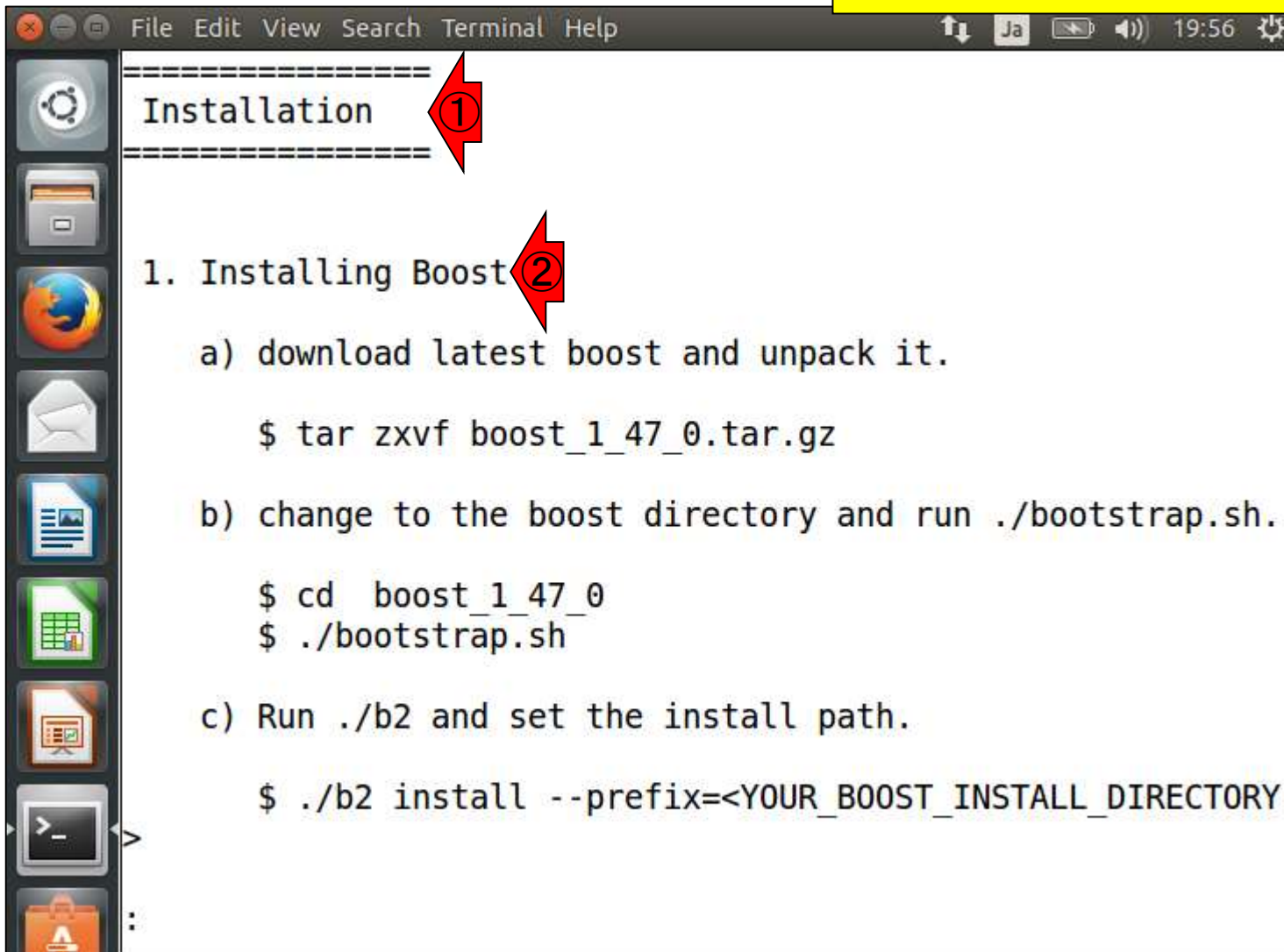
講習会ではBridger\_r2014-12-01.tar.gz (約11MB)を~/Downloadsにダウンロード済み。wget時のデフォルトのURLは「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_r2014-12-01.tar.gz/download](https://sourceforge.net/projects/maseqassembly/files/Bridger_r2014-12-01.tar.gz/download)」でしたが、得られるファイル名がdownloadとなって気持ち悪かったので、「[https://sourceforge.net/projects/maseqassembly/files/Bridger\\_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

```
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 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

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



# Boost

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

Navigation menu items:

- WELCOME
- Getting Started
- Download
- Libraries
- Mailing Lists
- Reporting and Fixing Bugs
- Wiki
- INTRODUCTION
- COMMUNITY
- DEVELOPMENT
- DOCUMENTATION

# Boost

The screenshot shows the Boost Downloads page in a browser window. The address bar displays 'http://www.boost.org/users/download/'. The page header features the Boost logo and 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 is titled "BOOST DOWNLOADS" and includes a navigation menu with "Current Release", "Old Boost Releases", and "Git Repositories". A red arrow labeled "1" points to the "Current Release" section, which displays "VERSION 1.61.0" and "May 13th, 2016 02:58". Below this, it lists "New Libraries" and "Updated Libraries". A second red arrow labeled "2" points to the "Download" link in the navigation menu.

The right sidebar contains a "GET BOOST" button, a "Google Custom Search" box, a "Donate" button with payment logos, and a navigation menu with links for "WELCOME", "INTRODUCTION", "Download", "News", "Version History", "License", "People", "Bibliography", "Who's Using Boost", "FAQ", "How Boost Started", "Index", "COMMUNITY", "DEVELOPMENT", and "DOCUMENTATION".



①講習会では~/Downloadsにダウンロード済み。②約85MB

# Boost

The screenshot shows the SourceForge page for Boost C++ Libraries. The file list is as follows:

Name	Modified	Size	Downloads / Week
↑		Parent folder	
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

Red arrows labeled ① and ② point to the 'boost\_1\_61\_0.tar.bz2' file and the 'Parent folder' header, respectively.

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
pwd
ls -l boo*
bzip2 -dc boost_1_61_0.tar.bz2 | t
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/Dow

### includeとlibの確認 ###
```

```
File Edit View Search Terminal Help
boost_1_61_0/libs/iostreams/test/finite_state_filter_test.cp
p
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
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 ]
```



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

# less README

```
File Edit View Search Terminal Help 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

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

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



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

# ./bootstrap.sh

```
File Edit View Search Terminal Help
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 ./b2 and install
$ ./b2 install
```

```
File Edit View Search Terminal Help
p
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
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] ./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 Boost

```
$ tar zxvf boost_1_61_0.tar.gz
```

b) change to the Boost directory

```
$ cd boost_1_61_0
$ ./bootstrap.sh
```

→ c) Run ./b2 and see the output

```
$ ./b2 install
```

Bootstrapping is done. To build, run:

```
./b2
```

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](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 ]



# 続: less README

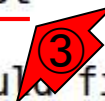
①Bridgerの「less README」の./b2部分を眺める。②の赤下線部分を自分で指定しないといけないらしい。例えば③のように指定するらしい

```
File Edit View Search Terminal Help 09:50
-> 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

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, beacuse you need to tell the Bridger inst
aller 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に対応するのはよくわからないので、もうちょっとマニュアルを見る

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

If the boost i
d two sub-directories
/home/czheng/l
/home/czheng/l

Note: The defau
/local. Take note of
directory, bea
aller where to find b
:
```

```
File Edit View Search Terminal Help
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
iu@bielinux[boost_1_61_0]
```



# 自分に置き換える

①find?!。どっちにしる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、と書いているので、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

```
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 commnd 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, beacuse you need to tell the Bridger inst
aller where to find boost later on.
```

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, beacuse you need to tell the Bridger installer where to find boost later on.

```
-rw-rw-r-- 1 iu iu 2477 5月 6 06:10 b2
-rw-rw-r-- 1 iu iu 918 6月 28 20:55 bjam
-rwxrwxr-x 1 iu iu 10631 5月 6 06:10 bootstrap.sh
iu@bielinux[boost_1_61_0]
```

```
Ja 13:37
[ 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 06:10 bootstrap.log
6 06:10 bootstrap.sh
[ 1:37午後]
```



①boostディレクトリに移動したので、②今はココ。③ include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 commnd is :
$ ./b2 install --prefix=/home/czheng/local/boost

If the boost i
d two sub-directories
/home/czheng/l
/home/czheng/l

Note: The defau
/local. Take note of
directory, bea
aller where to find b
:
```

```
File Edit View Search Terminal Help
iu@bielinux[boost_1_61_0] cd boost [ 2:01午後]
iu@bielinux[boost] pwd [ 2:01午後]
/home/iu/Downloads/boost_1_61_0/boost
iu@bielinux[boost] ls -ld include lib [ 2:01午後]
ls: cannot access include: No such file or directory
ls: cannot access lib: No such file or directory
iu@bielinux[boost] █ [ 2:01午後]
```



# ./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
-> 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

If the boost i
d two sub-directories
/home/czheng/l
/home/czheng/l

Note: The defau
/local. Take note of
directory, bea
aller where to find b
:
```

```
File Edit View Search Terminal Help
iu@bielinux[boost] cd .. [ 2:07午後]
iu@bielinux[boost_1_61_0] pwd [ 2:08午後]
/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
```



(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 commnd is :
$ ./b2 install --prefix=/home/czheng/local/boost

If the boost i
d two sub-directories
/home/czheng/l
/home/czheng/l

Note: The defau
/local. Take note of
directory, bea
aller where to find b
:
```

```
File Edit View Search Terminal Help
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-stat
ic/threading-multi/libboost_unit_test_framework.a
common.copy /home/iu/Downloads/boost_1_61_0/boost/lib/libboo
st_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-stat
ic/threading-multi/libboost_test_exec_monitor.a
common.copy /home/iu/Downloads/boost_1_61_0/boost/lib/libboo
st_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ができています

# includeとlibの確認

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

If the boost i
d two sub-directories
/home/czheng/l
/home/czheng/l

Note: The defau
/local. Take note of
directory, bea
aller where to find b
:
```

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



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

# 続: 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
```

# 続: less README

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

```
File Edit View Search Terminal Help
-> d) Set the LD_LIBRARY_PATH enviroment 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
:
```





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

# 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 lines to the ~/.bash_profile or ~/.profile file:
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/boost_1_61_0/boost/lib

Save and close the file.

OR

just type the following lines in the terminal:
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/boost_1_61_0/boost/lib
```

```
File Edit View Search Terminal Help
iu@bielinux[boost_1_61_0] tail -n 5 ~/.zshrc [ 4:06午後]
# f1
export PATH=$PATH:/home/iu/Downloads/FastQC:/home/iu/bin
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
iu@bielinux[boost_1_61_0] echo 'export LD_LIBRARY_PATH=/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH' >> ~/.zshrc
iu@bielinux[boost_1_61_0] tail -n 5 ~/.zshrc [ 4:06午後]
export PATH=$PATH:/home/iu/Downloads/FastQC:/home/iu/bin
export CLASSPATH=/home/iu/Downloads/Rockhopper.jar
export LD_LIBRARY_PATH=/home/iu/Downloads/boost_1_61_0/boost/lib:$LD_LIBRARY_PATH
iu@bielinux[boost_1_61_0] [ 4:06午後]
```

# LD\_LIBRARY\_PATH

```
File Edit View Search Terminal Help 15:09
-> d) Set the LD_LIBRARY_PATH enviroment variable:

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

Append the fol
/.profile file:
$ export LD_LI
:$LD_LIBRARY_PATH

Save and close

OR

just type the
$ export LD_LI
:$LD_LIBRARY_PATH
```

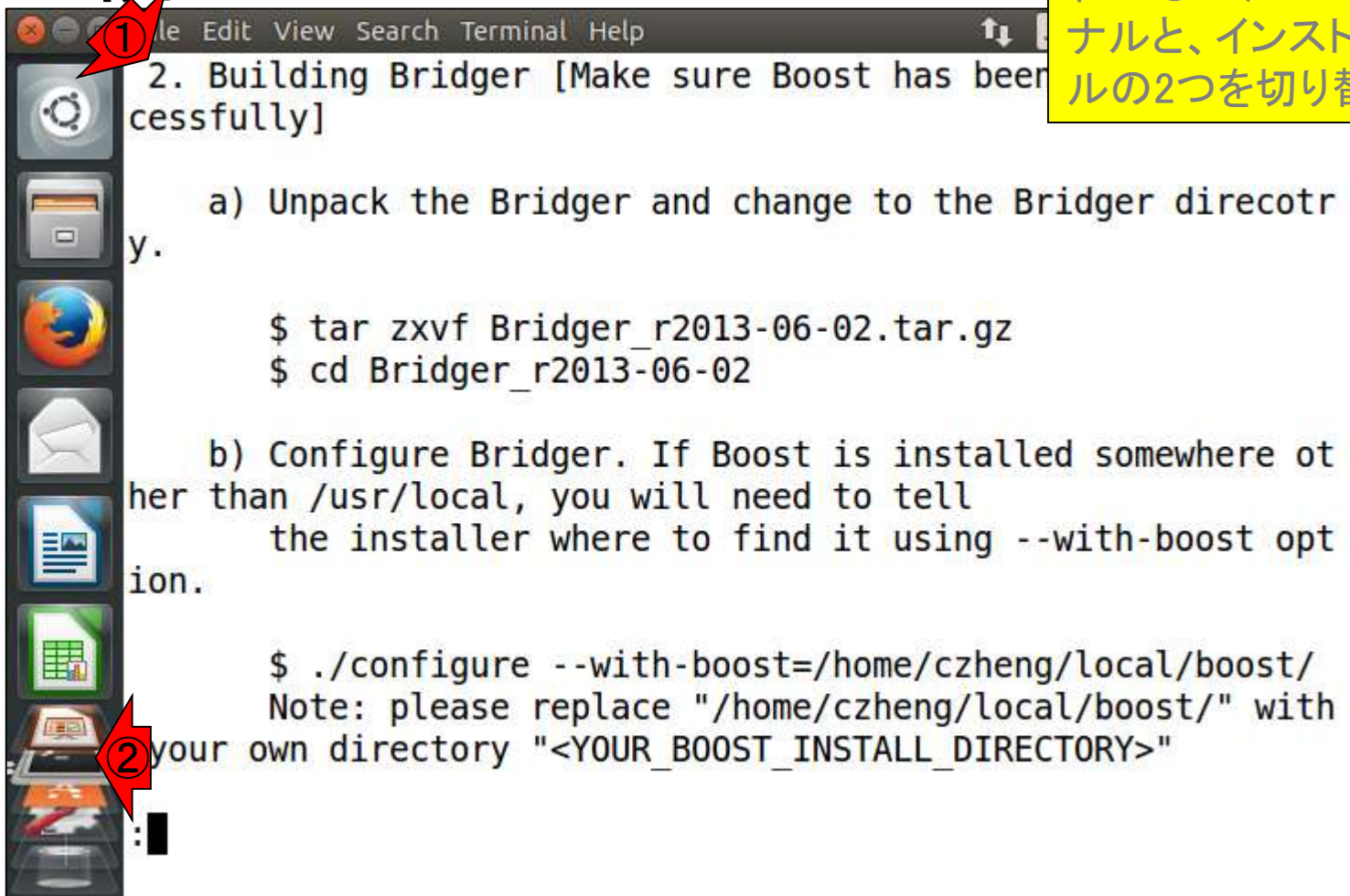
```
File Edit View Search Terminal Help 16:14
iu@bielinux[boost_1_61_0] echo $LD_LIBRARY_PATH [ 4:14午後]
iu@bielinux[boost_1_61_0] source ~/.zshrc [ 4:14午後]
iu@bielinux[boost_1_61_0] echo $LD_LIBRARY_PATH [ 4:14午後]
/home/iu/Downloads/boost_1_61_0/boost/lib:
iu@bielinux[boost_1_61_0] █ [ 4:14午後]
```





# 続: less README

Bridgerの「less README」で次の項目(①2. Building Bridger)に移行。②の部分を見ればわかるが、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



```
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>"
:
```



# 続: less README

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

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

① a) Unpack the Bridger and change to the Bridger directory.

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

③ 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 Bridgerのディレクトリに移動して、② ls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 suc
cessfully]

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

$ tar zxvf Bri
$ cd Bridger_r

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

$ ./configure
Note: please r
your own directory "
:
    
```

```

File Edit View Search Terminal Help
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           Makefile.in
AUTHORS            config.h           INSTALL          perl-lib
autom4te.cache     config.h.in       libtool          plugins
ax_boost_base.m4  config.log        LICENCE          README
boost.m4           config.status     m4               sample_test
Bridger.pl         configure         Makefile         src
build-aux          configure.ac      Makefile.am      stamp-h1
iu@bielinux[Bridger_r2014-12-01] ./configure --with-boost=/h
ome/iu/Downloads/boost_1_61_0/boost
    
```



# configure実行後

```

File Edit View Search Terminal Help
2. Building Bridger [Make sure Boost has been installed suc
cessfully]

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

$ tar zxvf Brid
$ cd Bridger_r

b) Configure Bridger
If you install Bridger in a directory other than /usr/local,
the installer will not be able to find the Boost headers.
Please specify the Boost headers location.

$ ./configure
Note: please specify your own directory if you install Bridger
in a directory other than /usr/local.

:

```

```

File Edit View Search Terminal Help
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/h
ome/niu/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 <changzmaths@gmail.com> with questions and bug repor
ts.

niu@bielinux[Bridger_r2014-12-01] █

```

[ 5:06午後 ]



# 続: less README

Bridgerの「less README」で次の項目(① make)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

```
File Edit View Search Terminal Help
c) Make Bridger.

$ make

note: If you build boost successfully 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 direct
ory)
    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:
=====
:█
```

# make

```
File Edit View Search Terminal Help
c) Make Bridger.
$ make
```

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

3. Test the installa  
are distribution in t  
\$ cd src  
\$ ./Assemble -h  
you would see:  
=====

```
File Edit View Search Terminal Help
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/h
ome/iu/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 <changzmaths@gmail.com> with questions and bug repor
ts.

iu@bielinux[Bridger_r2014-12-01] make
```

[ 5:17午後 ]



# make実行後

```

File Edit View Search Terminal Help
c) Make Bridger.

$ make

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

3. Test the installa
are distribution in t
$ cd src
$ ./Assemble -h
you would see:
=====
:

```

```

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 ]

```



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な

# 続: less README

```
3. Test the installation. Test data are provided with sofew  
are 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 co  
ntaining reads  
  
** Optional :  
  --kmer_length/-k <int>       : length of kmer, default  
: 25.  
  --double_stranded_mode       : set it true if double s  
:  
:
```



# ./Assemble -h

```

3. Test the installation. Test data are provided with sofew
are distribution in the sample_test directory.

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

Usage: Assemble
=====

**Required :
--reads/-i <stri
ntaining reads

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

```

iu@bielinux[Bridger_r2014-12-01] clear [ 5:26午後 ]

iu@bielinux[Bridger_r2014-12-01] pwd [ 5:26午後 ]
/home/iu/Downloads/Bridger_r2014-12-01

iu@bielinux[Bridger_r2014-12-01] ls [ 5:26午後 ]
aclocal.m4 ChangeLog COPYING Makefile.in
AUTHORS config.h INSTALL perllib
autom4te.cache config.h.in Libtool plugins
ax_boost_base.m4 config.log LICENCE README
boost.m4 config.status m4 sample_test
Bridger.pl configure Makefile src
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



```

File Edit View Search Terminal Help
3. Test the installation. Test data are provided with sofew
are distribution in the sample_test directory.
$ cd src
$ ./Assemble -h
you would see:
=====
Usage: Assemble
=====
**Required :
--reads/-i <string> : path to the directory containing reads
--kmer_length/-k <int> : kmer length, default: 25.
--double_stranded :

```



```

File Edit View Search Terminal Help
--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 alte
rnative extension that is
                                not an error, default
                                t: 0.05.
--pair_gap_length <int> : gap length of paired reads,
default: 200.
--out_dir/-o <string> : name of directory for output
t, default : ./RawGraphs
--help/-h : display the help information.
=====
=====
iu@bielinux[src] █ [ 5:41午後 ]

```



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

# 続: less README

```
File Edit View Search Terminal Help 17:57
=====
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
```

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

# 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]
① iu@bielinux[src] cd ~/Downloads/Bridger_r2014-12-01
iu@bielinux[Bridger_r2014-12-01] pwd [ 7:15午後]
/home/iu/Downloads/Bridger_r2014-12-01
② iu@bielinux[Bridger_r2014-12-01] ls [ 7:15午後]
aclocal.m4 ChangeLog COPYING Makefile.in
AUTHORS config.h INSTALL perllib
autom4te.cache config.h.in libtool plugins
ax_boost_base.m4 config.log LICENCE README
boost.m4 config.status m4 sample_test
Bridger.pl configure Makefile src
build-aux configure.ac Makefile.am stamp-h1
iu@bielinux[Bridger_r2014-12-01] [ 7:15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

The image shows a terminal window with a dark background and a sidebar on the left containing application icons. The terminal output is as follows:

```
=====  
=====  
Note : If you s  
file loading shared li  
.0:  
cannot  
or directory", please  
mmand:  
"export  
st_1_47_0/lib:$LD_LIB  
Test Bridger wit  
$ cd sample_test  
$ ./run_Me.sh  
:  
iu@bielinux[~/Downloads/Bridger_r2014-12-01/sample_test] 19:23  
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.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  
##### Done #####  
iu@bielinux[sample_test] [ 7:22午後]
```

Annotations in the image:

- ①: Points to the terminal prompt `iu@bielinux[~/Downloads/Bridger_r2014-12-01/sample_test]`.
- ②: Points to the `cd sample_test/` command.
- ③: Points to the `run_Me.sh` file listed in the `ls` output.
- ④: Points to the `more run_Me.sh` command.
- ⑤: Points to the first line of the script, `#!/bin/bash -ve`.



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

# ./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 wit
$ cd sample_test
$ ./run_Me.sh
```



```
File Edit View Search Terminal Help
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 ###
./Bridger.pl --seqType fq --left reads.left.fq.gz --right r
eads.right.fq.gz --SS_lib_type RF --CPU 2
##### Done #####
iu@bielinux[sample_test] ./run_Me.sh [ 7:22午後]
```



# Errorで終了



```
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

:
```

```
File Edit View Search Terminal Help
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>Assemb
le.log died with ret 256 !
iu@bielinux[sample_test] [ 7:53午後]
```





# 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ていようなので…

The image shows a Linux terminal window with a dark theme. The window title is "File Edit View Search Terminal Help". The terminal output is as follows:

```
=====  
=====  
Note : If you s  
file loading shared li  
.0:  
cannot  
or directory", please  
mmand:  
"export  
st_1_47_0/lib:$LD_LIB  
Test Bridger wit  
$ cd sample_test  
$ ./run_Me.sh  
:  
### 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>Assemb  
le.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 ②  
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estima  
tion.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 ]
```

Red arrows point to the error message (①) and the newly created directory 'bridger\_out\_dir' (②) in the terminal output.



# 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ていようなので、③中身を確認。④Assemble.logというログファイルが確かにある

```
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 wit
$ cd sample_test
$ ./run_Me.sh
:
```

```
File Edit View Search Terminal Help
Error, cmd: /home/iu/Downloads/Bridger_r2014-12-01/src/Assem
ble --reads both.fa -k 25 --pair_end --fr_strand 1 2>Assemb
le.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_estima
tion.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]
```



# 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…。

```
File Edit View Search Terminal Help [ 8:32午後 ]
=====
Note : If you s
ile loading shared li
.0:
cannot
or directory", please
mmand:
"export
st_1_47_0/lib:$LD_LIB
Test Bridger wit
$ cd sample_test
$ ./run_Me.sh
:

File Edit View Search Terminal Help [ 8:32午後 ]
① iu@bielinux[sample_test] ls -lt | head
total 428
drwxrwxr-x 3 iu iu 4096 6月 29 19:53 bridger_out_dir ②
-rwxr-xr-x 1 iu iu 497 6月 2 2013 run_abundance_estima
tion.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
③ 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 ④
-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
⑤ iu@bielinux[sample test] more bridger out dir/Assemble.log
[Error] Cannot create directory ./RawGraphs/ !
iu@bielinux[sample_test] [ 8:32午後 ]
```



# 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 s
file loading shared li
.0:
cannot
or directory", please
mmand:
"export
st_1_47_0/lib:$LD_LIB
Test Bridger wit
$ cd sample_test
$ ./run_Me.sh
:
```

```
File Edit View Search Terminal Help
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_estima
tion.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 ①
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

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

```
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
:
```

```
File Edit View Search Terminal Help
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_estima
tion.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]
```

# Contents

- 乳酸菌RNA-seqデータ解析のおさらいと問題設定
- *de novo*ト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



ト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

# 発現量推定

## (Rで)塩基配列解析

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

- 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\(トランスクリプトーム配列を利用\)](#) **1** (last modified 2016/06/30)
- 解析 | クラスタリング | [|](#)について (last modified 2016/05/05)
- 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) | [MBCcluster.Seq\(Si 2014\)](#) (last modified 2015/11/15)

### What's new?

- このウェブページ  
リーソフト Rと必要  
[法\(Windows2015\)](#)  
本頁です (2015/11/15)

### 解析 | 発現量推定(ト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ジ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-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。間違いないいくつか含まれているとは思いますが、2016年6月に調べた結果をリストアップします:

#### R用:

- AllelicImbalance: [Gádin et al., BMC Bioinformatics, 2015](#)
- kimport: [Soneson et al., F1000Res., 2015](#)
- RNAontheBENCH (github上にある): [Germain et al., Nucleic Acids Res., 2016](#)
- SARTools (github上にある): [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-Skim: [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, バイブライン系:

- 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](#)
- バイブライン(Ion Proton用): [Yuan et al., BMC Genomics, 2016](#)
- バイブライン(single-cell用): [Ntranos et al., Genome Biol., 2016](#)



# TIGAR2

① TIGAR2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, Naqasaki 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.

PMID: [25560536](#) PMCID: [PMC4304212](#) DOI: [10.1186/1471-2164-15-S10-S5](#)

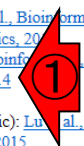
### 解析 | 発現量推定(トランスクリプトーム配列を利用) NEW

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、5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、数箇所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を用いたalignment-freeで行う発現量推定を行うSailfishやRNA-Skimは従来法に比べて劇的に高速化が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が、2016年6月に調べた結果をリストします:

- [AllelicImbalance: Gádin et al., BMC Bioinformatics, 2015](#)
- [ximport: Soneson et al., F1000Res., 2015](#)
- [RNAontheBENCH \(github上にある\): Germain et al., Nucleic Acids Res., 2016](#)
- [SARTools \(github上にある\): Varet et al., PLoS One, 2016](#)

本:

- [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-Skim: Zhang and Wang, Bioinformatics, 2014](#)
- [QuASAR \(allele-specific\): Harvey et al., Bioinformatics, 2015](#)
- [TIGER2: Nariai et al., BMC Genomics, 2014](#)
- [SUPPA: Alamanos 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](#)
- [NLDMSeg: 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](#)



www. 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 \(transcript-based approach\): Zhao et al., BMC Genomics, 2016](#)
- [パイプライン \(union exon-based approach\): Zhao et al., Genome Biol., 2016](#)

# TIGAR2

- [TIGAR2: Nariai et al., BMC Genomics, 2014](#)

①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*

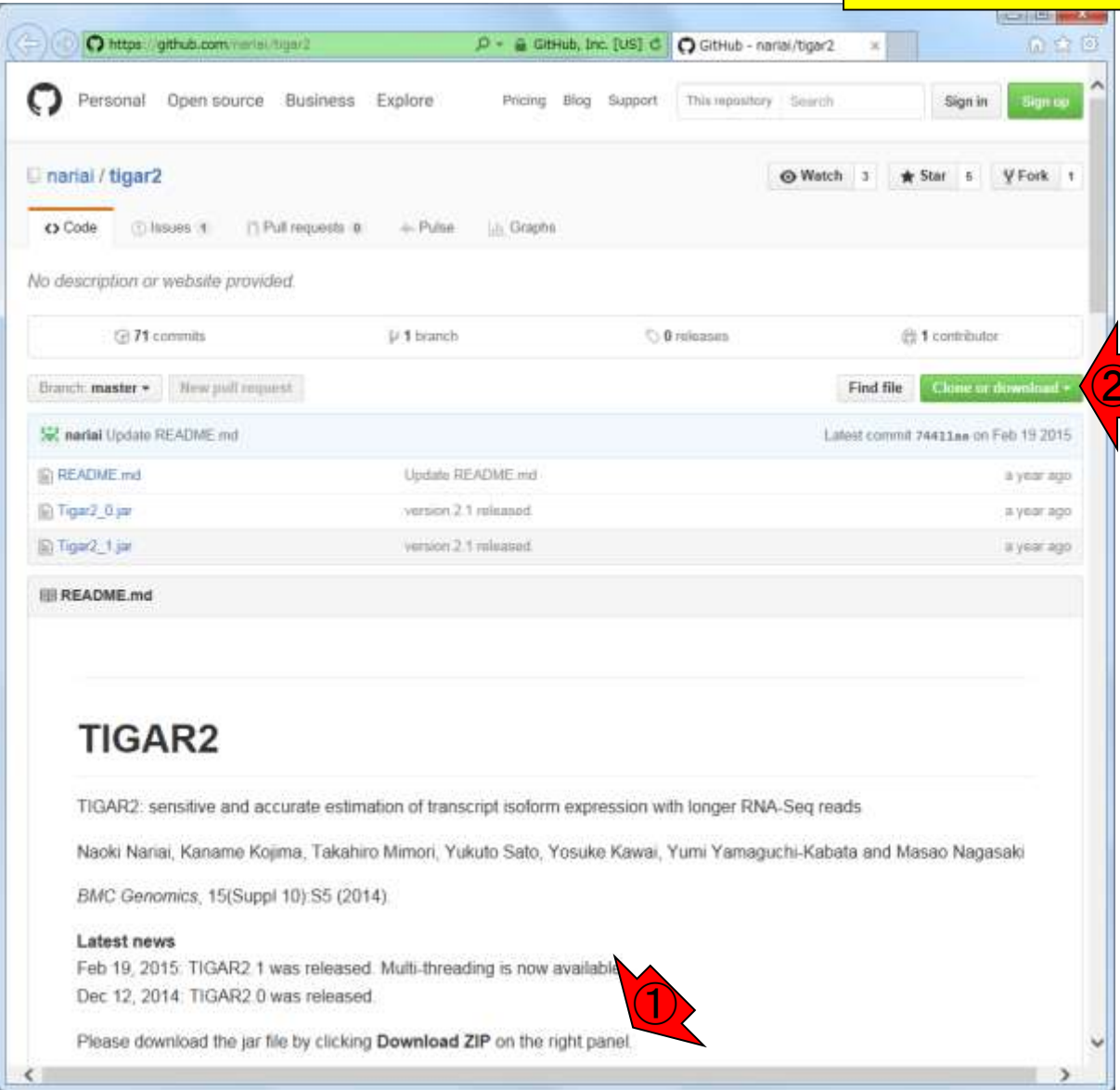
### 解凍 ###
unzip master.zip

### Githubと照合 ###
pwd
head tigar2-master/README.md

### 動作確認 ###
java -jar ~/Downloads/tigar2-master/Tigar2_1.jar
```

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…

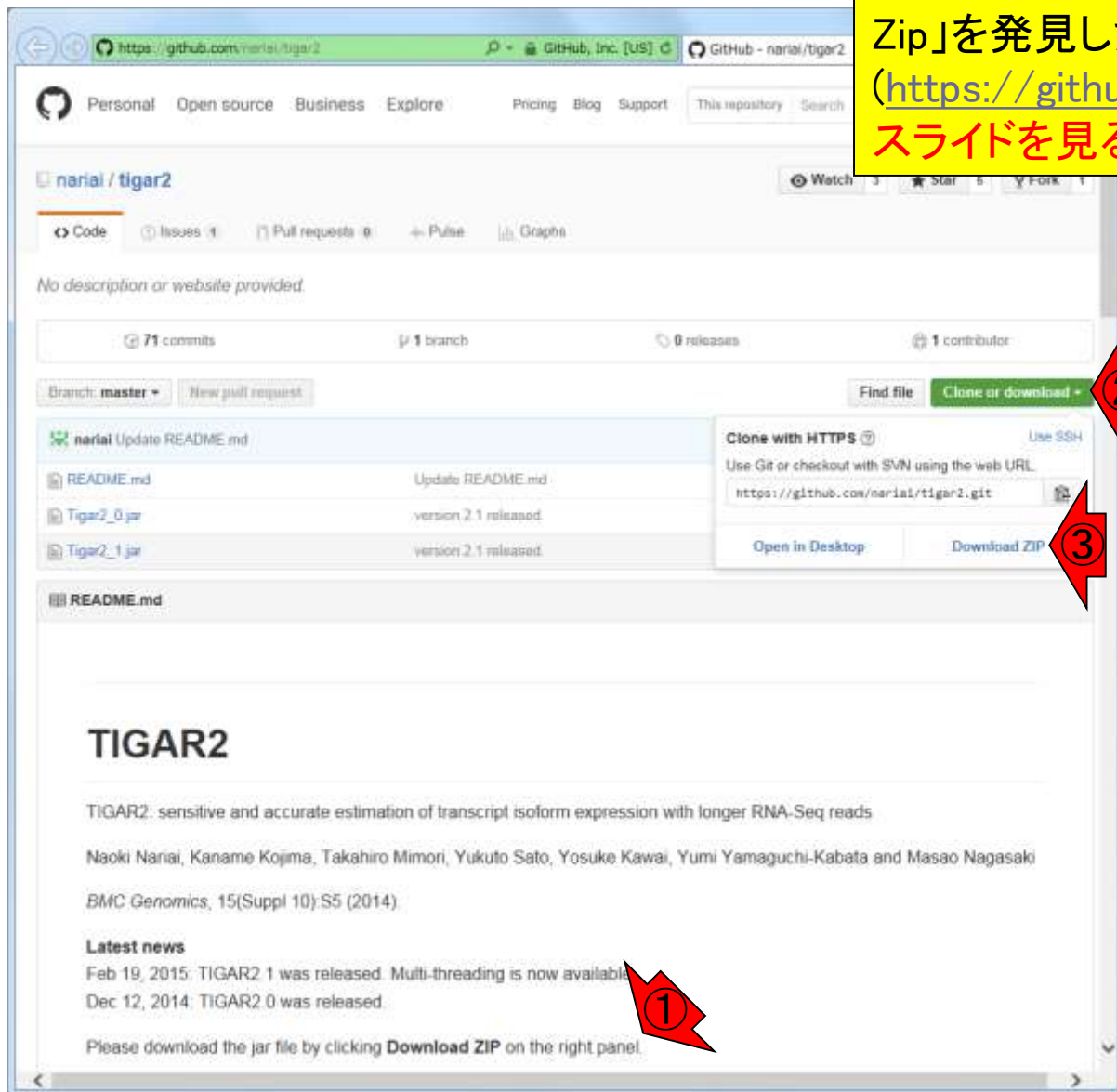
# TIGAR2





# 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



# 解凍

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

• [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

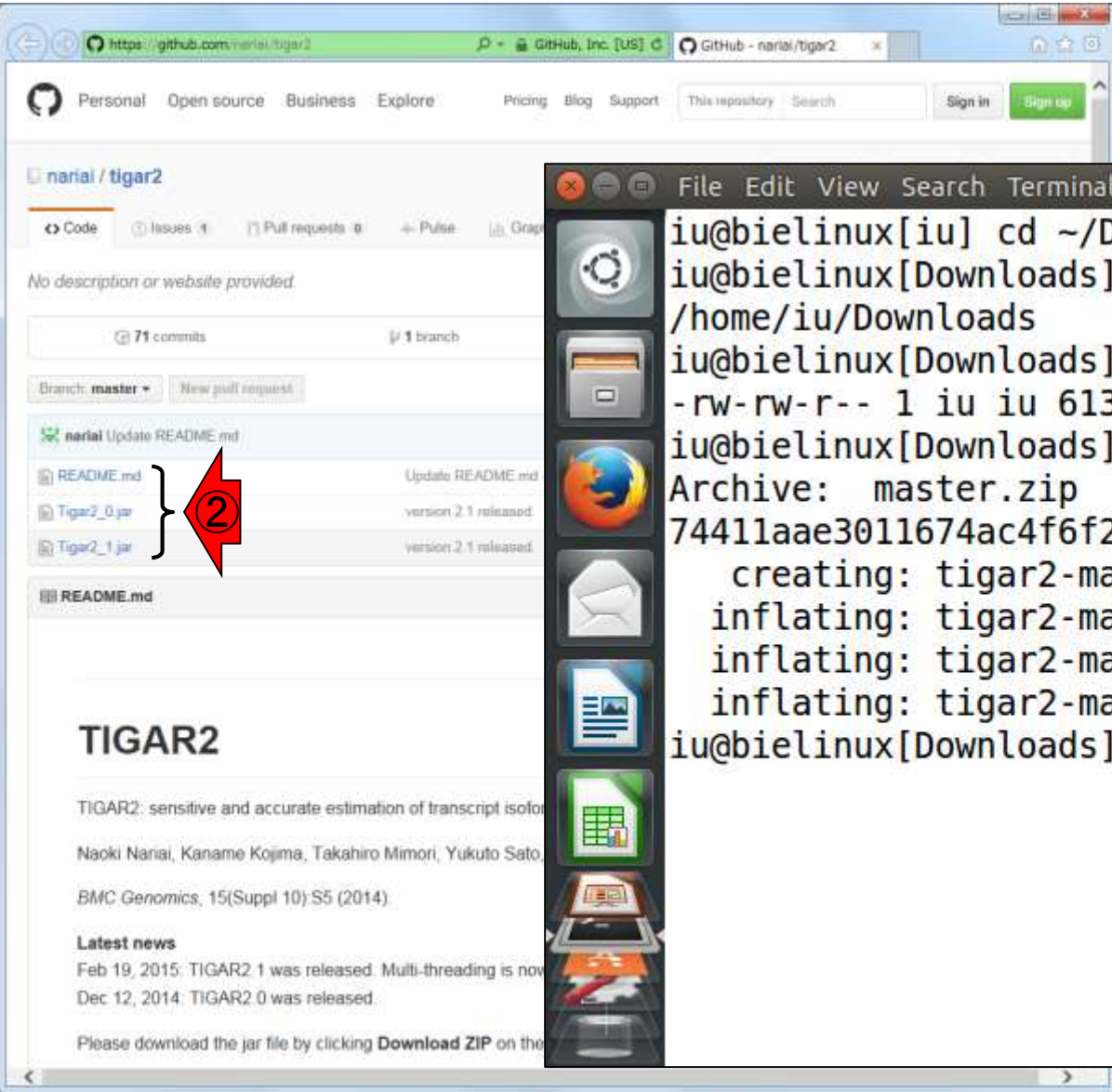
### 動作確認 ###
java -jar ~/Downloads/tigar2-master
```

```

iu@bielinux[iu] cd ~/Downloads [ 2:20午後 ]
iu@bielinux[Downloads] pwd [ 2:20午後 ]
/home/iu/Downloads
iu@bielinux[Downloads] ls -l mas* [ 2:20午後 ]
-rw-rw-r-- 1 iu iu 6135780 6月 20 19:40 master.zip
iu@bielinux[Downloads] unzip master.zip [ 2:20午後 ]
Archive:  master.zip
74411aae3011674ac4f6f237fe3e158fedefe047
  creating:  tigar2-master/ ②
  inflating: tigar2-master/README.md
  inflating: tigar2-master/Tigar2_0.jar } ③
  inflating: tigar2-master/Tigar2_1.jar }
iu@bielinux[Downloads] [ 2:20午後 ]

```

# Githubと照合



```

File Edit View Search Terminal Help
iu@bielinux[iu] cd ~/Downloads [ 2:20午後 ]
iu@bielinux[Downloads] pwd [ 2:20午後 ]
/home/iu/Downloads
iu@bielinux[Downloads] ls -l mas* [ 2:20午後 ]
-rw-rw-r-- 1 iu iu 6135780 6月 20 19:40 master.zip
iu@bielinux[Downloads] unzip master.zip [ 2:20午後 ]
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] [ 2:20午後 ]

```

A red arrow labeled '1' points to the output of the unzip command, specifically the files being inflated: README.md, Tigar2\_0.jar, and Tigar2\_1.jar.



# Githubと照合

①README.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…

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

narial / tigar2 Watch 1 Star 5 Fork 1

Code Issues 1 Pull requests 0 Pulse Graphs

No description or website provided.

71 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Find file Clone or download

File	Commit
narial (Update README.md)	Update README.md
README.md	Update README.md
Tigar2_0.jar	version 2.1 released
Tigar2_1.jar	version 2.1 released

Clone with HTTPS Use SSH  
Use Git or checkout with SVN using the web URL.  
https://github.com/narial/tigar2.git  
Open in Desktop Download ZIP

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)

### 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

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

nariai / tigar2

Code Issues Pull requests Pulse Graphs

No description or website provided.

71 commits 1 branch

Branch: master + New pull request

README.md	Update README.md
Tigar2_0.jar	version 2.1 released.
Tigar2_1.jar	version 2.1 released.

README.md

**TIGAR2**

TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads

Naoki Nariiai, 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.

```

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 Nariiai, Kaname Kojima, Takahiro Mimori, Yukuto Sato, Y
osuke Kawai, Yumi Yamaguchi-Kabata and Masao Nagasaki

*BMC Genomics*, 15(Suppl 10):S5 (2014).

iu@bielinux[Downloads] █
  
```

# README.md

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

## 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)

### 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.

Usage: java -jar Tigar2\_1.jar FASTA SAM OUT

FASTA : reference FASTA file  
SAM : target SAM/BAI 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/ref/rna.fa.gz>  
  
e.g.) mouse  
<http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/ref/rna.fa.gz>

### 2. Build bowtie2 index



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-master/
```

```
File Edit View Search Terminal Help
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 sam/bam file was generated from paired-end reads.
--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*ト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



①TIGAR2の推奨パイプラインに従って行う。② Step1は、トランスクリ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

# 推奨パイプライン1

Recommended pipeline to run TIGAR2

1. Prepare cDNA reference sequences in FASTA format.  
e.g.) human  
`http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMna.fa.gz`  
e.g.) mouse  
`http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMna.fa.gz`
2. Build bowtie2 index  

```
mkdir ref
bowtie2-build refMna.fa ./ref/refMna
```
3. Run bowtie2  
For single-end data  

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna sample.fastq > sample.sam
```

  
For paired-end data  

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna -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 refMna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```

  
For paired-end data  

```
java -jar Tigar2_1.jar --thread_num 8 refMna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format



# Step 1

①実際の作業はこちら。②赤枠内をコピー。発現量推定を行いたいトランスクリプトーム配列ファイル(Trinity1.fasta)の場所へ移動し、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

- 推奨パイプラインでTIGAR2を実行 (スライド201) **①**
- TIGAR2の、「Recommended pipeline to run TIGAR2」の部分

```
### 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でとりにあえずやる ###
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

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.a
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-sensiti
-1 data1.fq.gz -2 data2.fq.gz
```

```
pwd
ls -l test.sam
```

```
File Edit View Search Terminal Help
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 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

# 推奨パイプライン2

**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



# Step2

①実際の作業はこちら。②赤枠内をコピー。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

- 推奨パイプラインでTIGAR2を実行 (スライド201) **①**  
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/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。

### 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  
### トリム前のpaired-endでとりあえ

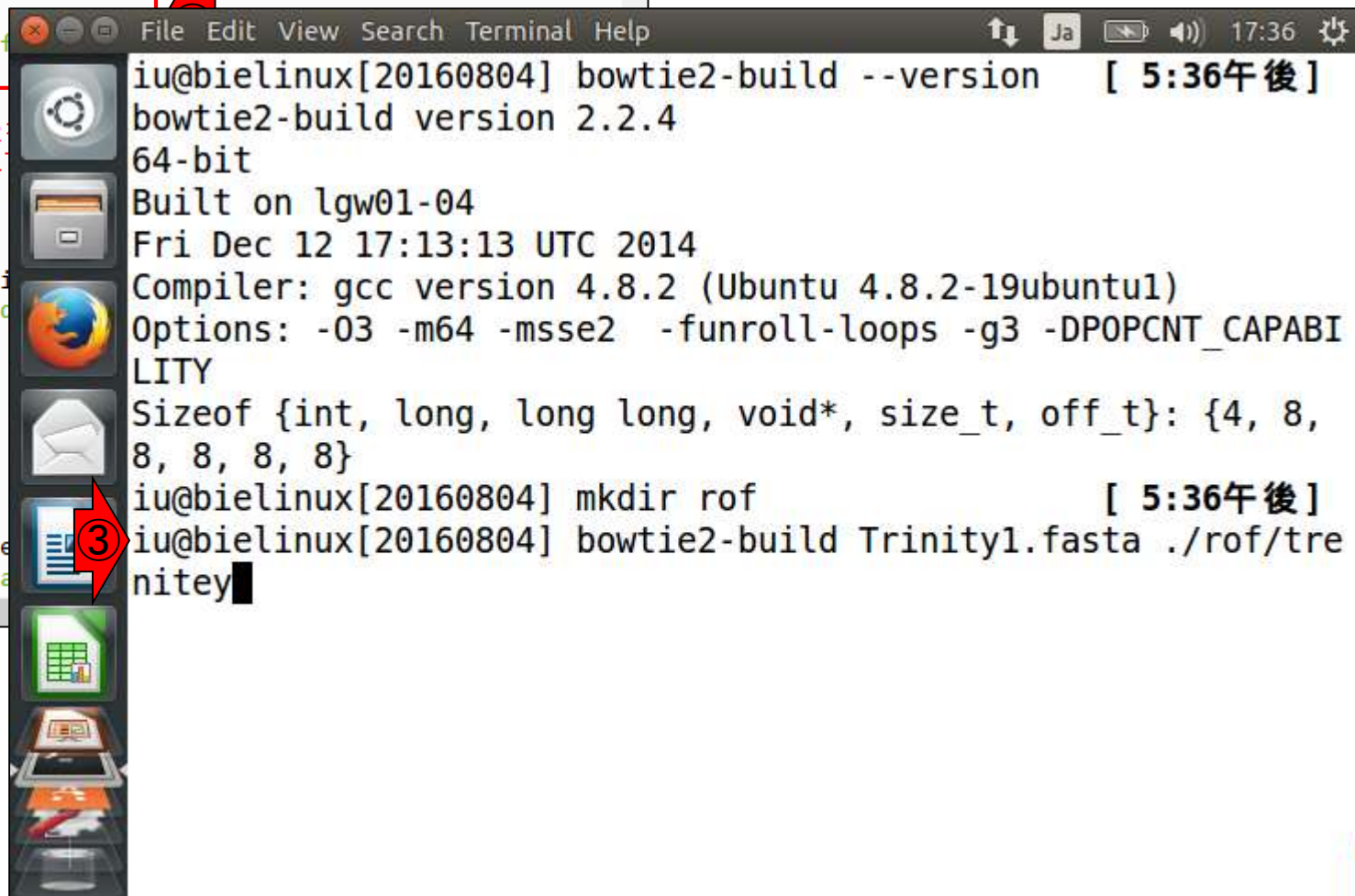
```
bowtie2 --version
bowtie2 -h
```

```
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-master
--thread_num 2 Trinity1.fasta
```



```
iu@bielinux[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_CAPABI
LITY
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8,
8, 8, 8, 8}
iu@bielinux[20160804] mkdir rof [ 5:36午後 ]
iu@bielinux[20160804] bowtie2-build Trinity1.fasta ./rof/tru
nitye
```

# 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のでしょうか

- 推奨パイプラインで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  
### トリム前のpaired-endでとりあ

```
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-master
--thread_num 2 Trinity1.fasta
```

```
File Edit View Search Terminal Help
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 [ 5:42午後 ]
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] [ 5:43午後 ]
```



# 推奨パイプライン3

## Recommended pipeline to run TIGAR2

### 1. Prepare cDNA reference sequences in FASTA format.

```
e.g.) human
http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/refMna.fa.gz

e.g.) mouse
http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMna.fa.gz
```

### 2. Build bowtie2 index

```
mkdir ref
bowtie2-build refMna.fa ./ref/refMna
```

### 3. Run bowtie2

For single-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna sample.fastq > sample.sam
```

For paired-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna -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 refMna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```

For paired-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format



# Step3

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

- 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でとりにあわずやる ###
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

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

- 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pair-endでとりあえずやる ###
bowtie2 --version
bowtie2 -h

bowtie2 -x test -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-master.jar
--thread_num 2 Trinity1.fasta test.sam
--is_paired --alpha_zero 0.1

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n
```

```
File Edit View Search Terminal Help
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}
iu@bielinux[20160804] bowtie2 -h [11:14午前]
```



# 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 ###
```

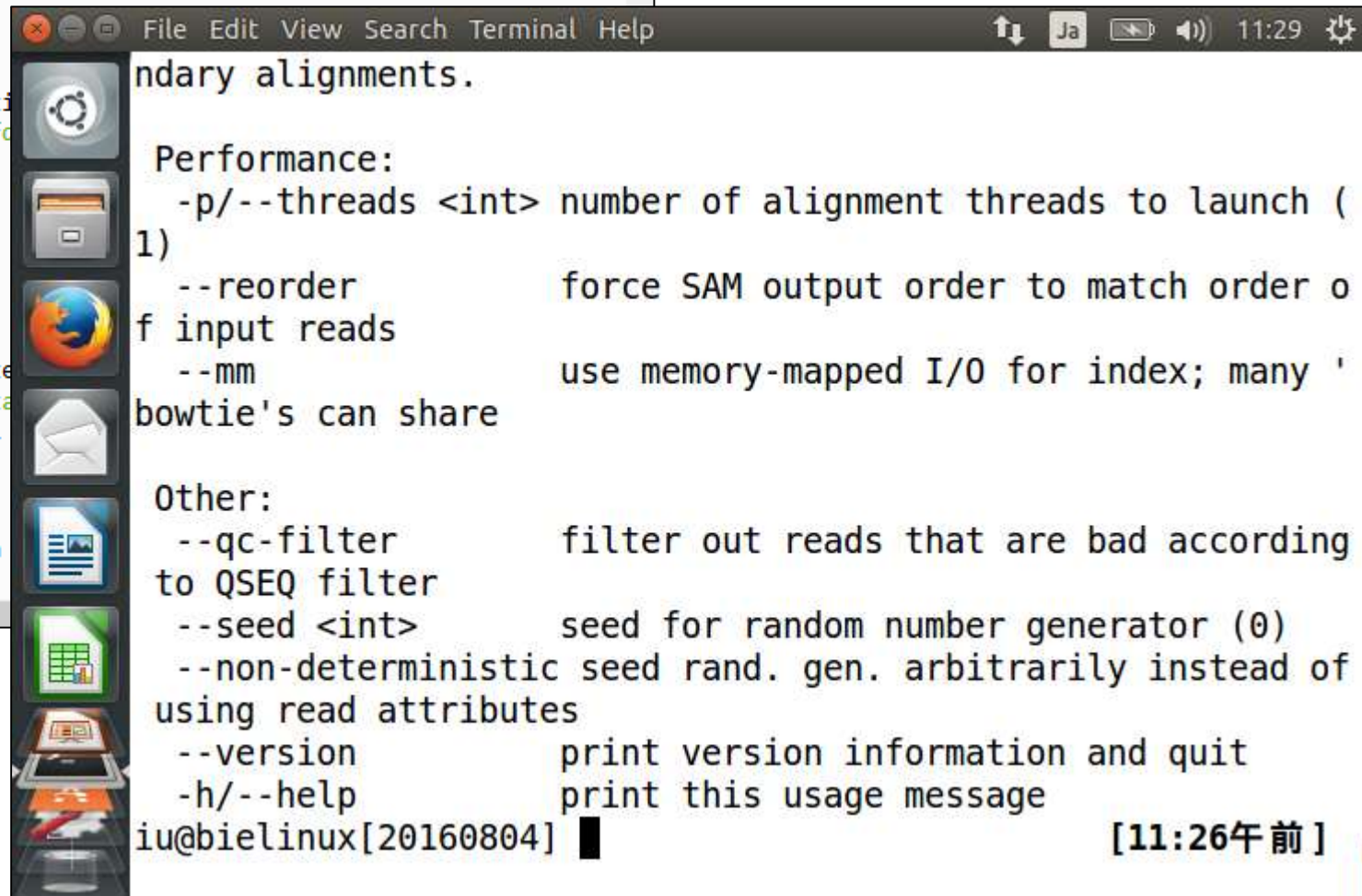
```
bowtie2 --version
bowtie2 -h
bowtie2 -x index -k 100 --very-sensitive
          -1 data1.fq.gz -2 data2.fq.gz
```

```
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-master.jar
      --thread_num 2 Trinity1.fasta test.sam
      --is_paired --alpha_zero 0.1
```

```
ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```



```
File Edit View Search Terminal Help
secondary alignments.

Performance:
  -p/--threads <int> number of alignment threads to launch (
1)
  --reorder                force SAM output order to match order o
f 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]
```

# 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。

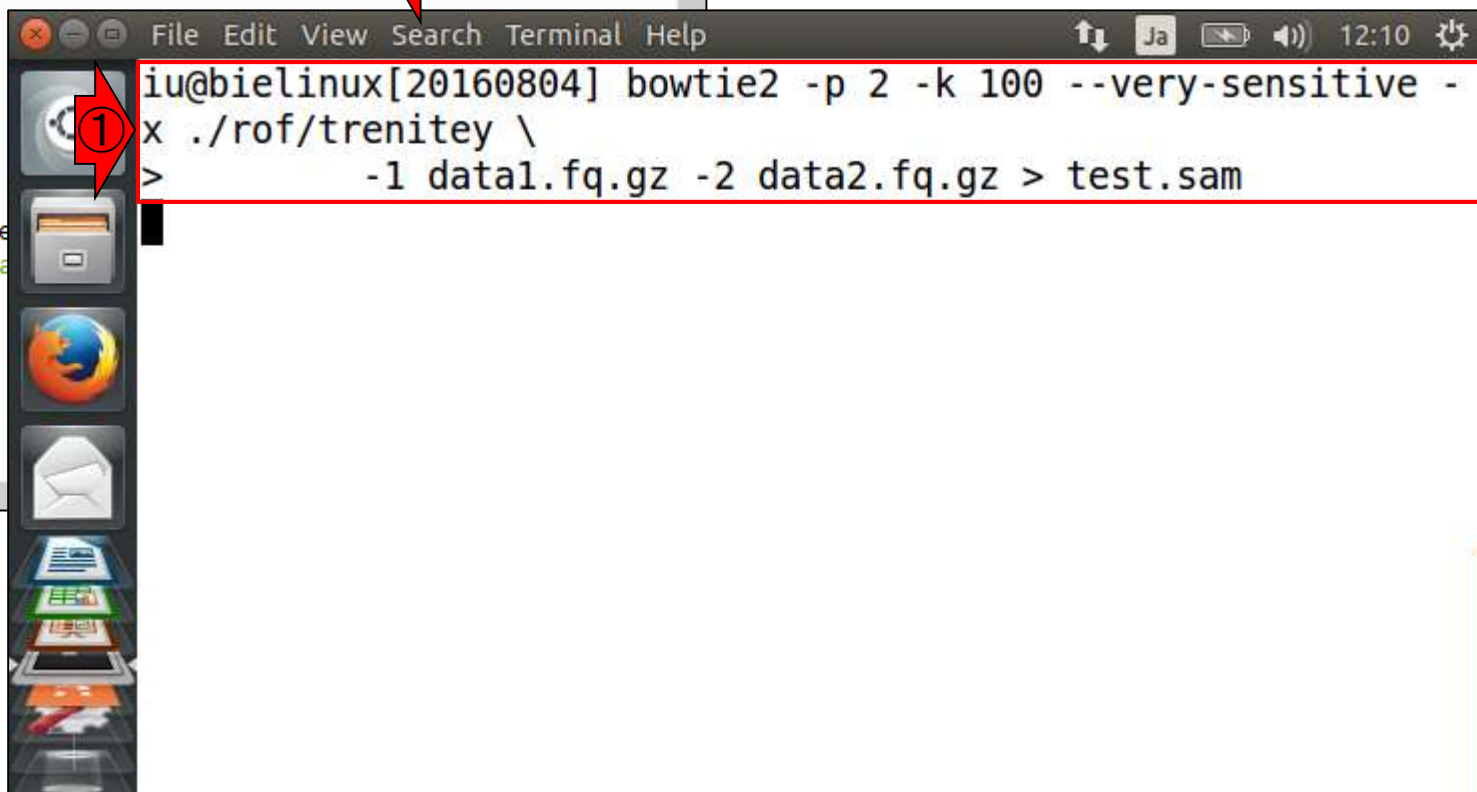
```
### 3. マッピングプログラムbowtie2を実行 ###
### トリム前のpaired-endでとりあえずやる ###
bowtie2 --version
bowtie2 -h
```

```
bowtie2 -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
--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
```

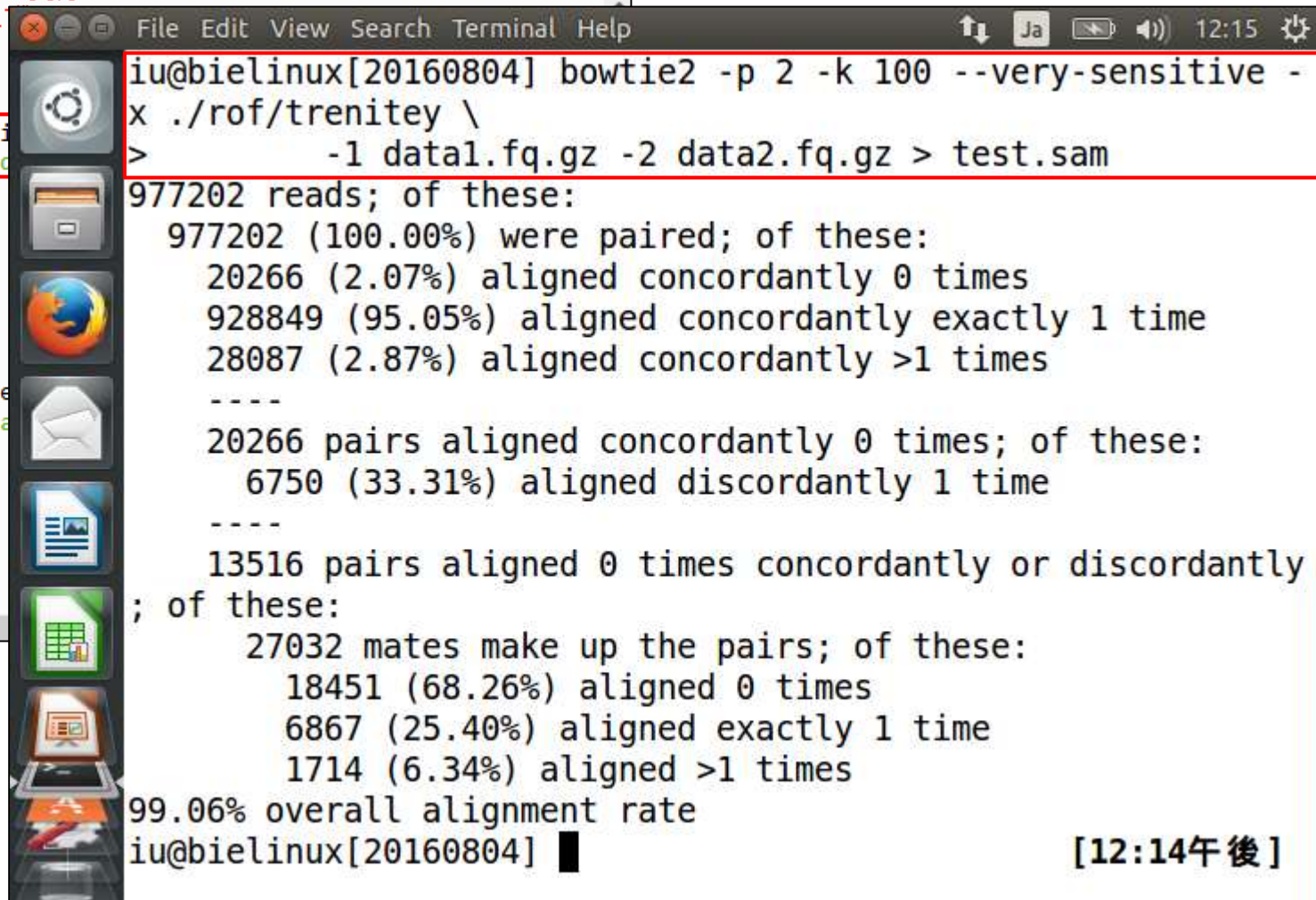
```
bowtie2 -p 2 -k 100 --very-sensitive
      -1 data1.fq.gz -2 data2.fq.gz
```

```
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-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] 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

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対し、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…

- 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 > 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
--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

ちなみに、①この928,849リード(95.05%)は1回だけマップされたリード(1 time)ということなので、おそらく uniquely mapped readsということなのだろう。②28,087リード(2.87%)がmulti-mappersと判断した。プログラムによって表現方法が異なりますが、慣れです

- 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 -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 test.sam --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]



①lsでファイルサイズを確認。  
②656MB。結構デカいですね

# 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-master.jar
      --thread_num 2 Trinity1.fasta test.sam
      --is_paired --alpha_zero 0.1

ls -l test*
head -n 5 test_out.txt
grep ">" Trinity1.fasta | head -n 5
```

```
iu@bielinux[20160804] pwd [ 1:34午後 ]
/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 ]
```





# 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/refMna.fa.gz

e.g.) mouse
http://hgdownload.soe.ucsc.edu/goldenPath/mm9/bigZips/refMna.fa.gz
```

### 2. Build bowtie2 index

```
mkdir ref
bowtie2-build refMna.fa ./ref/refMna
```

### 3. Run bowtie2

For single-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna sample.fastq > sample.sam
```

For paired-end data

```
bowtie2 -p 8 -k 100 --very-sensitive -x ./ref/refMna -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 refMna.fa sample.sam --alpha_zero 0.1 sample_out.txt
```

For paired-end data

```
java -jar Tigar2_1.jar --thread_num 8 refMna.fa sample.sam --is_paired --alpha_zero 0.1 sample_out.txt
```

Output format



# 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/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
```

# Step4

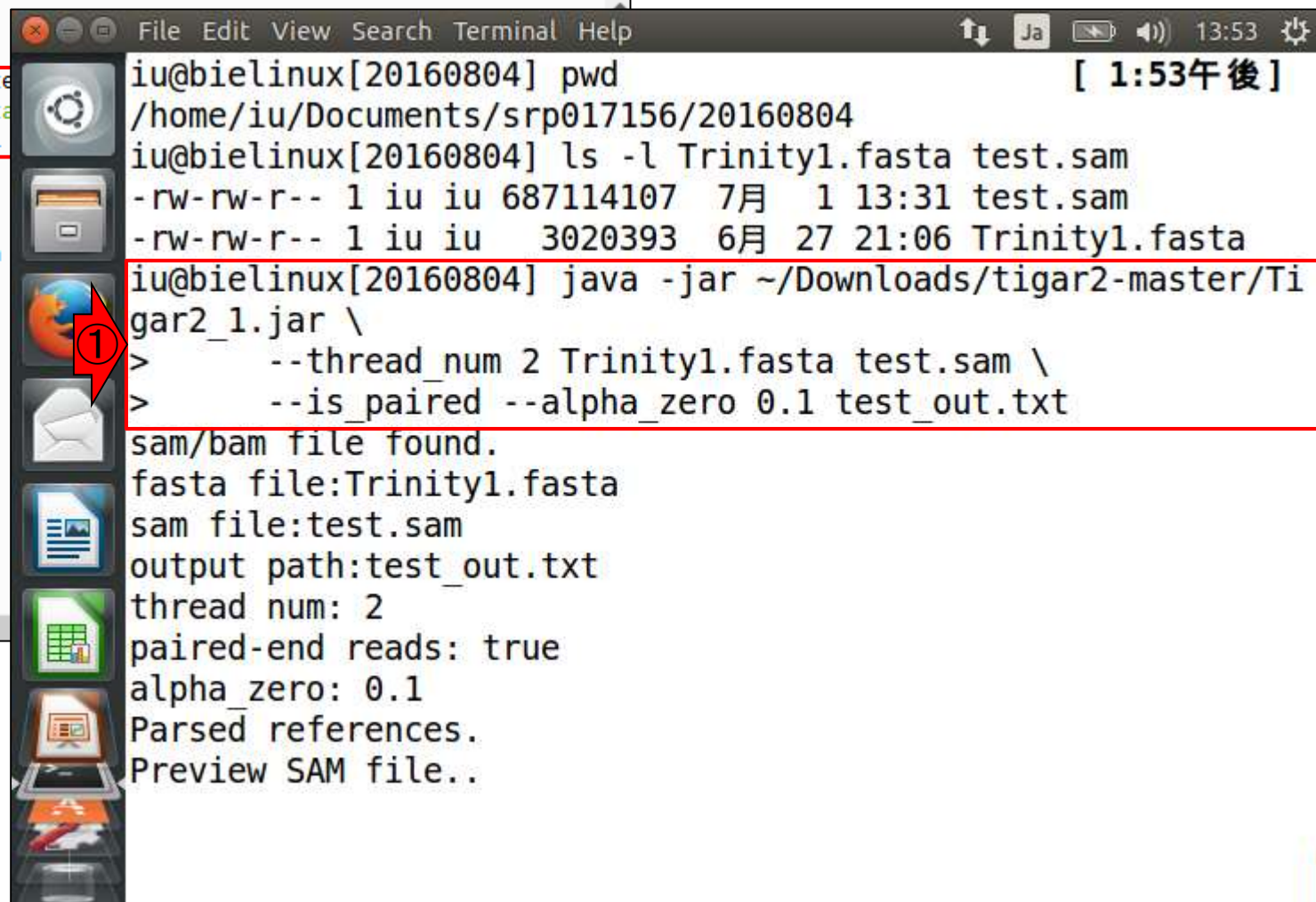
- 推奨パイプラインでTIGAR2を実行 (スライド201)

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/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] 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/Tigar2_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
```

```
File Edit View Search Terminal Help
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*ト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

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、事実上使用ことはない

Output format

ID: transcript (sRNA) 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:

```
santools sort sample_out.txt.opt.bam sample_opt_sorted
santools index sample_opt_sorted
```

Please start GV\_2.3.14 or later to see the optimized alignment of read.

Please note that the current cases, please specify

```
e.g.) java -Xmx16g -Xms16g -jar santools.jar -i sample_opt_sorted -o sample_opt_sorted
e.g.) java -Xmx32g -Xms32g -jar santools.jar -i sample_opt_sorted -o sample_opt_sorted
e.g.) java -Xmx64g -Xms64g -jar santools.jar -i sample_opt_sorted -o sample_opt_sorted
```

You can also choose

\* Build FM-index for

ID: transcript (mRNA) ID that the program predicted

LENGTH: transcript length

Z: the number of expected fragments that the program assigned to the transcript

① FPKM: normalized expression level (Fragments Per Kilobase of exon per Million mapped fragments)

③ THETA: estimated parameter (transcript abundance), essentially Z divided by total fragments.



①出力ファイル(test\_out.txt)の最初の5行分を表示

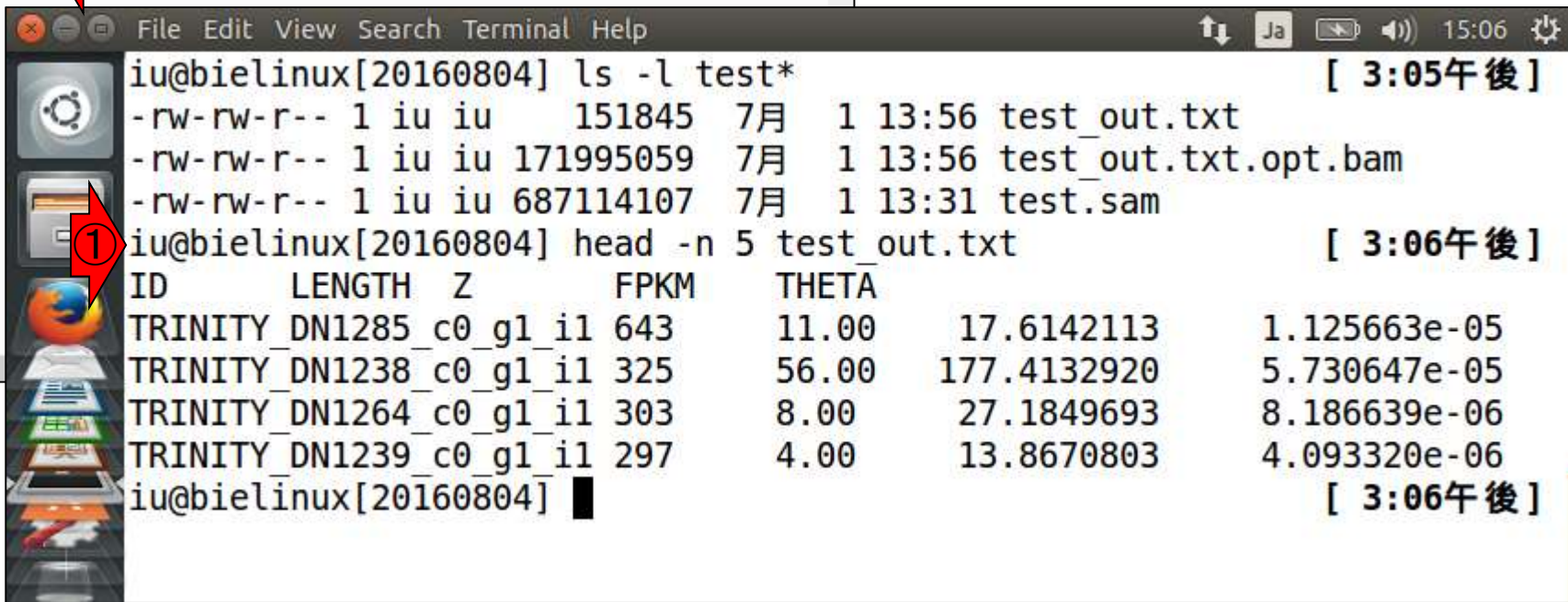
# 出力ファイル確認

- 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
```



# 出力ファイル確認

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

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
```

Terminal output showing file listing and the first 5 lines of test\_out.txt. Red arrows point to specific columns in the table 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

Red arrows indicate: ④ points to the ID column, ② points to the Z column, ① points to the FPKM column, and ③ points to the THETA column.



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] [ 3:18午後 ]
```



# FPKM値を手計算 参考

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

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] 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]
```

↑③      ↑②      ↑①

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 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/test_out.txt
#cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
head -n 5 test_out.txt
```

```
### Rを起動してFPKM値を手計算 ###
R -q
11 * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222)
56 * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222)
8 * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222) * (1000000/971222)
q(save="no")
```

Terminal screenshot showing the execution of `ls -l test*` and `head -n 5 test_out.txt`. The output shows a table with columns: ID, LENGTH, Z, FPKM, and THETA. A red arrow points to the Z column.

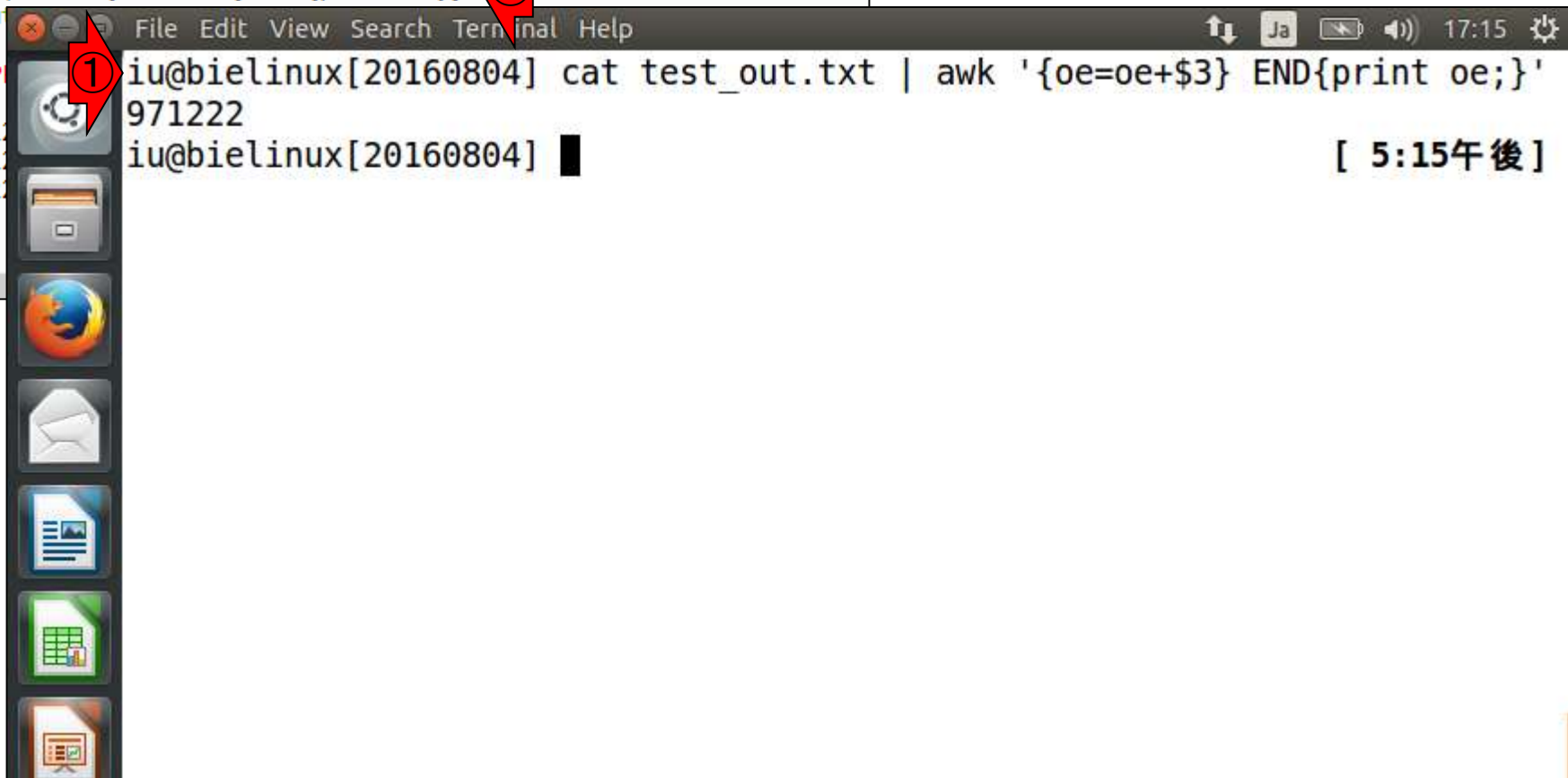
ID	LENGTH	Z	FPKM	THETA
TRINITY_DN1285_c0_g1_i1	643	17.6142113	11.00	1.125663e-05
TRINITY_DN1238_c0_g1_i1	325	177.4132920	56.00	5.730647e-05
TRINITY_DN1264_c0_g1_i1	303	27.1849693	8.00	8.186639e-06
TRINITY_DN1239_c0_g1_i1	297	13.8670803	4.00	4.093320e-06

# 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、971,222。この計算結果を確かめたければ、自分で②test\_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

• FPKM値を手計算 (スライド22) **②**  
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_nginx_sokushu_2016/3/test_out.txt  
#cp ~/Desktop/backup/test_out.txt .  
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}' ①  
head -n 5 test_out.txt
```





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 参考

教科書p132-137



• 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.txt
```

Terminal output showing the calculation of FPKM values:

```
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
1 TRINITY_DN1285_c0_g1_i1 643    11.00  17.6142113  1.125663e-05
2 TRINITY_DN1238_c0_g1_i1 325    56.00  177.4132920  5.730647e-05
3 TRINITY_DN1264_c0_g1_i1 303    8.00   27.1849693  8.186639e-06
4 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
```

Red arrows in the image point to:

- ①: The first column of the table (ID).
- ②: The FPKM value (17.6142113) in the first row of the table.
- ③: The calculated RPKM value (17.61421) in the terminal output.

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.txt
```

```
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
```

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 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.txt
```

Terminal output showing the calculation of FPKM for the first row of the table:

```
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
```

The table of results is as follows:

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

Red arrows in the image point to the value 177.4132920 in the THETA column of the second row, and to the value 17.61421 in the R console output, which is the calculated FPKM value for the first row.



# FPKM値を手計算 参考

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

• 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、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。\$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/test_out.txt
#cp ~/Desktop/backup/test_out.txt .
cat test_out.txt | awk '{oe=oe+$3} END{print oe;}'
head -n 5 test_out.txt
```

Terminal output showing the calculation of FPKM values for TRINITY transcripts. The calculation is:  $11 * (1000000 / 971222) * (1000 / 643)$ . Red arrows indicate the components: ① points to the first '11' (number of fragments), ② points to the result of the first multiplication (17.61421), and ③ points to the final result (177.413292).

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

# 発現量推定

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

## (Rで)塩基配列解析

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

- 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 2015/11/11)
- 解析 | 基礎 | 平均-分散プロット | [Biological replicates](#) (last modified 2015/11/11)
- 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\(トランスクリプトーム配列を利用\)](#) (last modified 2016/05/25)
- 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 2015/02/26)
- 解析 | [クラスタリング | 遺伝子間\(基礎\) | MBCluster.Seq\(Si 2014\)](#) (last modified 2015/02/26)

### What's new?

- このウェブページにRと必要ライブラリをインストールする方法(Windows2015)を載せています。(2015/11/11)

## 解析 | 発現量推定(ト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ジ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-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が、2016年6月に調べた結果をリストアップします:

### R用:

- AllelicImbalance: [Gadin et al., BMC Bioinformatics, 2015](#)
- tximport: [Soneson et al., F1000Res., 2015](#)
- RNAontheBENCH (github上にある): [Germain et al., Nucleic Acids Res., 2016](#)
- SARTools (github上にある): [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-Skim: [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, バイブライシ系:

- 手法比較: [Kanitz et al., Genome Biol., 2015](#)
- 手法比較(transcript-based approach vs. 'union-based approach'): [Zhao et al., PLoS One, 2015](#)
- バイブライシ(Ion Proton用): [Yuan et al., BMC Genomics, 2016](#)
- バイブライシ(single-cell用): [Ntranos et al., Genome Biol., 2016](#)



# おまけ

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

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 Search 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 ]
```

