

3. 欲しいデータを取得する (DRA経由) (7/14)

<http://trace.ddbj.nig.ac.jp/dra/index.html>



DRA Search ▶ Sea

Accession :

Organism : StudyType :

CenterName : Platform :

Keyword :

Show records Sort by

D

3. 欲しいデータを取得する (DRAから) (8/14)

danio rerio

Transcriptome Analysis

ILLUMINA

DRASearch

Accession :

Organism : StudyType : CenterName : Platform : Keyword :

Show 20 records Sort by Study Search Clear

Search Results (262 studies)

#	STUDY	SUBMISSION	STUDY_TITLE	STUDY_TYPE	ORGANISM
1	ERP000016	ERA000092	Sequencing the Zebrafish transcriptome form a range of tissues and developmental stages using the Illumina Genome Analyzer	Transcriptome Analysis	Danio rerio

DRASearch

ERP000016

Study Detail

Title: Sequencing the Zebrafish transcriptome form a range of tissues and developmental stages using the Illumina Genome Analyzer

Study Type: Transcriptome Analysis

Abstract: Paired-end sequence data has been generated using polyA selected RNA from a range of zebrafish tissues and developmental stages using the Illumina Genome Analyzer. These data have been used to improve the gene annotation of the zebrafish genome

Description: Zebrafish total RNA was extracted from embryonic and adult tissue, then polyA selected. After fragmentation and reverse transcription Illumina sequencing libraries were prepared. Paired-end sequence runs were performed with 36, 37, 54 and 76 base reads on the Illumina Genome Analyzer. This data is .. [more]

Center Name: SC

Navigation

- Submission: ERA000092 FTP
- Experiment: ERX000398 FASTQ SRA
- ERX000399 FASTQ SRA
- ERX000400 FASTQ SRA
- ERX000401 FASTQ SRA
- ERX000402 FASTQ SRA
- ERX000403 FASTQ SRA
- ERX000404 FASTQ SRA
- ERX000405 FASTQ SRA
- ERX000406 FASTQ SRA
- ERX000407 FASTQ SRA
- ERX000408 FASTQ SRA
- ERX000409 FASTQ SRA

ftp://ftp.ddbj.nig.ac.jp/ddbj...0092/ERX000398/^へ接続中...

サーバで"ftp.ddbj.nig.ac.jp"用の名前とパスワードを入力してください。

ユーザの種類: ☒ ゲスト ☐ 登録ユーザ

キャンセル 接続

ERX000398

ERR003990_1.fastq.bz2 ERR003990_2.fastq.bz2 ERR003990.fastq.bz2

ERR003990.sra

3. 欲しいデータを取得する (DRAから) (9/14)

/ddbj_database/dra/fastq/ERA/

 [親ディレクトリ]

名前	サイズ	更新日
 ERR003990.fastq.bz2	63.4 kB	2015/10/12 9:00:00
 ERR003990_1.fastq.bz2		2015/10/12 9:00:00
 ERR003990_2.fastq.bz2		2015/10/12 9:00:00
 ERR003991.fastq.bz2		2015/10/12 9:00:00
 ERR003991_1.fastq.bz2		2015/10/12 9:00:00
 ERR003991_2.fastq.bz2		2015/10/12 9:00:00
 ERR003992.fastq.bz2	1.5 kB	2015/10/12 9:00:00
 ERR003992_1.fastq.bz2	162 MB	2015/10/12 9:00:00
 ERR003992_2.fastq.bz2	181 MB	2015/10/12 9:00:00

Control key を押しながらクリック

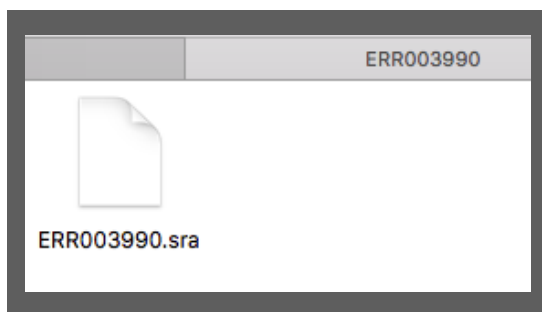
ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/ERA000/ERA000092/ERX000398/ERR003990_1.fastq.bz2

3. 欲しいデータを取得する (DRAから) (10/14)

後の作業のために下記4つのファイルをダウンロードしておく

```
[dstep@nt094 dstep20180126]$ wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/ERA000/ERA000092/ERX000398/ERR003991_1.fastq.bz2  
[dstep@nt094 dstep20180126]$ wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/ERA000/ERA000092/ERX000398/ERR003991_2.fastq.bz2  
[dstep@nt094 dstep20180126]$ wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/ERA000/ERA000092/ERX000398/ERR003992_1.fastq.bz2  
[dstep@nt094 dstep20180126]$ wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/ERA000/ERA000092/ERX000398/ERR003992_2.fastq.bz2
```

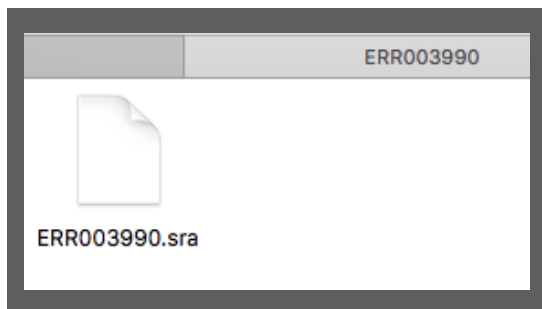
```
[dstep@nt094 dstep20180126]$ bunzip2 ERR00399*.bz2 # <- 時間がかかる
```



比較のために、ERR003990.sraファイルもダウンロードしておく

```
[dstep@nt094 DSTEP20180126]$ wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/sralite/ByExp/litesra/ERX/ERX000/ERX000398/ERR003990/ERR003990.sra
```

3. 欲しいデータを取得する (DRAから) (11/14)



```
# fastq-dump ERR003990.sra
```

```
/usr/local/pkg/sra-tools/2.5.7/bin/fastq-dump.2.5.7 --split-files ERR003990.sra <- やってはいけない
```

```
# qsub コマンドを利用する。必ず！！
```

```
[dstep@nt094 dstep20180126]$ cat sra2fastq.qsub
```

<- qsub用のスクリプトを作成して、

```
#!/bin/sh
#$ -S /bin/bash
#$ -cwd
#$ -l short
#$ -l s_vmem=1G,mem_req=1G
#$ -o ./
#$ -e ./

source ~/.bashrc

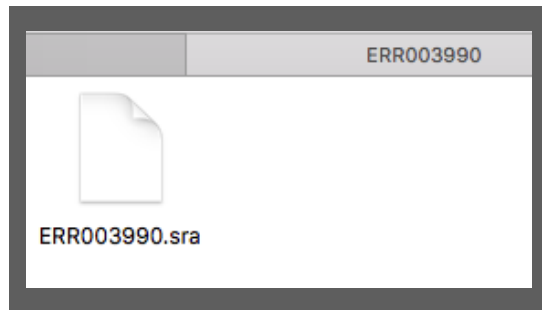
FASTQDUMP="/usr/local/pkg/sra-tools/2.5.7/bin/fastq-dump.2.5.7"
Dbdir="/home/dstep/tanomare/dstep20180126"

$FASTQDUMP --split-files ERR003990.sra
```

```
[dstep@nt094 dstep20180126]$ qsub sra2fastq.qsub
```

<- qsubコマンドを利用して計算機に投入

3. 欲しいデータを取得する (DRAから) (12/14)



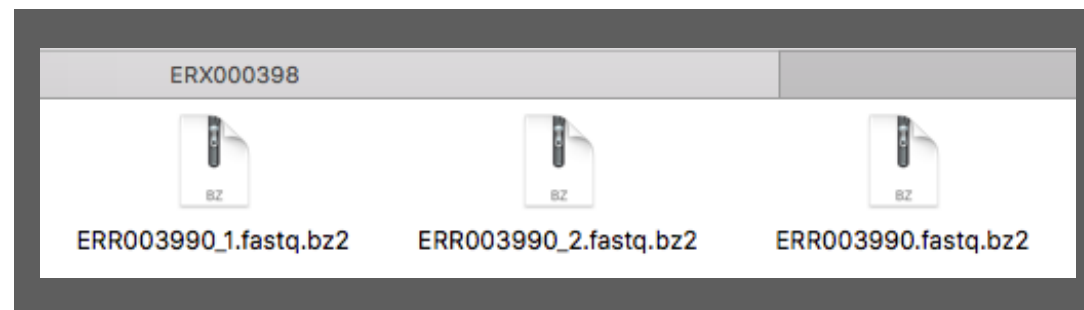
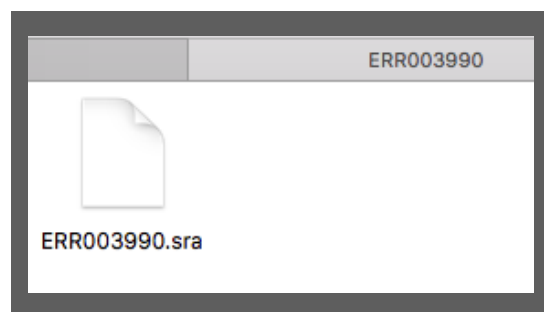
```
[dstep@nt094 dstep20180126]$ qsub sra2fastq.qsub
```

<- qsubコマンドを利用して計算機に投入

```
[dstep@nt094 dstep20180126]$ ls ERR*  
ERR003990.sra  ERR003990_1.fastq  ERR003990_2.fastq
```

SRAファイルから、ペアエンドのfastqファイルが取得できた

3. 欲しいデータを取得する (DRAから) (13/14)



```
[dstep@nt094 dstep20180126]$ ls ERR*  
ERR003990.sra  ERR003990_1.fastq  ERR003990_2.fastq
```

もともとdraに登録されていた
fastqファイル

*.sraファイルから自分でfastq-dump
を利用してfastqを作成した

両者は少し違う！

*.sraから作成したfastqファイルは
クオリティー等でtrimmingされていない(ことが多い)

NGSデータを利用する場合は、
低クオリティー部分やアダプター等の除去してから使いましょう
(この後fastQCやTrimmomaticの利用を体験しましょう。)

3. 欲しいデータを取得する (14/14)

以上で、

3-1. NIGのスパコンにすでに整備してあるデータベースを利用する

3-2. その他のゲノムの基礎データの登録サイト (いくつか)

3-3. DRAから取得

これらの典型的なデータの取得方法を学びました。

Webブラウザからクリックしてダウンロードする場合、

wgetやcurlコマンドを利用する場合、

ftpコマンドを利用する場合、などがありました。

またDRAやSRAに特徴的なこととしてfastq-dumpの利用方法も学びました。

なお、DRAから *.sraファイルを取得すると、cfq フォーマット(solidのcolor space)の配列という場合があります。cfqからfqへの変換方法を知っておきたい方ははありますか？

4. 自分の環境を使いやすくする

UNIX系の計算機環境では、自分の利用している環境を使いやすくするために、環境変数や各種設定ファイルの編集方法を学んでいなくてはなりません。本来最初に学ぶべきこれらの事柄をここまでは避けて説明してきました。

NIGの大型計算機には主要なゲノム解析様のプログラムがかなりインストールされていますが、これらを利用するためにも環境変数について学ばなければなりません。

また利用者が利用したいプログラムがインストールされていない場合は、利用者ご自身でご自身の計算機環境にそのプログラムをインストールする必要があるでしょう。

4-1. 環境変数について学びましょう

4-2. ローカルに好みのプログラムをインストールすることを学びましょう。

4. 1-4. 環境変数 (6/8)のおさらい (1/6)

```
# qloginしたターミナルにて
```

```
echo "TEST"
```

```
printenv
```

```
printenv | wc
```

```
printenv | sed -e "s/=.*/"
```

```
echo $HOME
```

```
echo $SHELL
```

```
echo $PWD
```

```
echo $USER
```

```
echo $PATH
```

```
time
```

```
which time
```

```
[dstep@nt098 ~]$ which time  
/usr/bin/time
```

time コマンドは /usr/binの下においてある

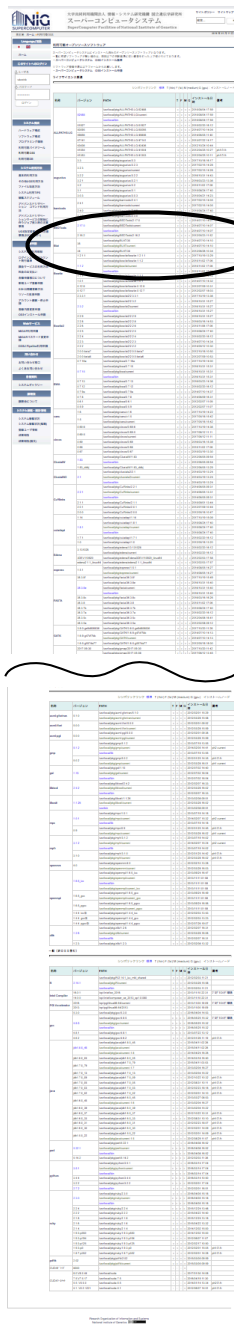
```
echo $PATH
```

/usr/binというディレクトリは確かに環境変数PATHの中に記載されている

```
[dstep@nt098 ~]$ echo $PATH  
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/ibutils/bin:/opt/intel/  
itac/8.1.3.037/bin:/home/dstep/local/bin
```

実際の表示とは変えてあります

4. 1-5. 利用可能OSS (7/8) のおさらい (2/6)



canu	0.5.9	/usr/local/pkg/bwa/0.5.9	○ ○ ○ ○	2012/02/07 11:07
	1.6	/usr/local/pkg/canu/v1.6	○ ○ ○ ○	2017/10/19 14:22
	1.5	/usr/local/pkg/canu/v1.5	○ ○ ○ ○	2017/05/18 15:42
		/usr/local/pkg/canu/current	○ ○ ○ ○	2017/05/18 15:42
	0.69.6	/usr/local/pkg/circos/0.69.6	○ ○ ○ ○	2017/10/19 14:36

/usr/local/pkg/canu/current

```
[dstep@nt094 ~]$ /usr/local/pkg/canu/current/bin/canu --version
```

```
Canu snapshot v1.5 +43 changes (r8243 f05c81531c19887459ca53ac3702509ee70eac22)
```

full pathを打たないで使えるようにしたい

```
[dstep@nt094 ~]$ canu
```

```
-bash: canu: コマンドが見つかりません
```

環境変数PATHにcanuのfull pathを追加する

下記は、打ち間違えると嫌な目にあいます。

```
$ export PATH=$PATH: "/usr/local/pkg/canu/current/bin"
```

確かにcanuと打つだけで、canuが使えるようになった。

```
$ which canu
```

```
/usr/local/pkg/canu/current/bin/canu
```

今後もずっと環境変数PATHにcanuのfull pathを追加されるようにする

下記は、打ち間違えるとひどい目にあいます。慎重に！！（もしくは試さない）

```
$ echo 'export PATH=$PATH: "/usr/local/pkg/canu/current/bin"' >> ~/.bashrc  
$ source ~/.bashrc
```

4. 自分の環境を使いやすくする (3/6)

すでにインストールしてあるプログラムについては、PATHを通すだけで利用できる。

```
[dstep@nt098 ~]$ cat ~/.bashrc
```

```
# < 略 >
```

```
# BOWTIE2
```

```
export PATH=$PATH: "/usr/local/pkg/bowtie2/current"
```

```
# CANU
```

```
export PATH=$PATH: "/usr/local/pkg/canu/current/bin"
```

```
# < 略 >
```

→ # .bashrcファイルの中身の例

自分で好みのプログラムをインストールする方法は！？

Gblocksというプログラムのインストールを体験してみましょう。

Gblocksは、SpainのCastresana Labが開発したアライメントの保存領域選択ツールです。

<http://molevol.cmima.csic.es/castresana/Gblocks.html>

<http://genome.gsc.riken.jp/osc/english/dataresource/>

http://molevol.cmima.csic.es/castresana/Gblocks/Gblocks_Linux64_0.91b.tar.Z

4. 自分の環境を使いやすくする (4/6) Gblocksの install

```
# http://molevol.cmima.csic.es/castresana/Gblocks/Gblocks_Linux64_0.91b.tar.Z
# Gblocksのインストール
```

ホームディレクトリにソースファイルを格納するディレクトリを作ります

```
[dstep@nt098 src]$ mkdir ~/src
[dstep@nt098 src]$ mkdir ~/src/gblocks
[dstep@nt098 src]$ cd ~/src/gblocks
[dstep@nt098 gblocks]$ pwd
/home/dstep/src/gblocks
```

wgetでソースファイルを取得取得

```
[dstep@nt098 gblocks]$ wget http://molevol.cmima.csic.es/castresana/Gblocks/Gblocks_Linux64_0.91b.tar.Z
```

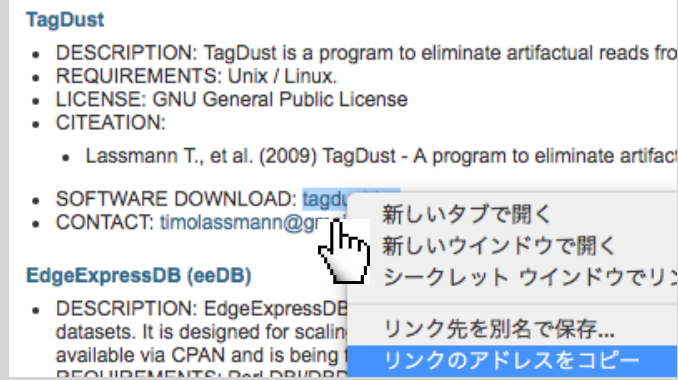
gunzipと tarでソースファイルを解凍  ダウンロード先のURLはコピペで良い

```
[dstep@nt098 gblocks]$ Gblocks_Linux64_0.91b.tar.Z
[dstep@nt098 gblocks]$ ls
Gblocks_Linux64_0.91b.tar
[dstep@nt098 tmp]$ tar xvf Gblocks_Linux64_0.91b.tar
```

```
[dstep@nt098 tmp]$ ls
Gblocks_0.91b  Gblocks_Linux64_0.91b.tar
```

インストールといってもGblocksは解凍するだけですぐ使える

```
[dstep@nt098 tmp]$ cd Gblocks_0.91b
[dstep@nt098 Gblocks_0.91b]$ ls [dstep@nt098 Gblocks_0.91b]$ cat nad3.pir
Documentation  Gblocks  more_alignments  nad3.pir  paths
[dstep@nt098 Gblocks_0.91b]$ cat nad3.pir
[dstep@nt098 Gblocks_0.91b]$ ./Gblocks nad3.pir -t=p -e=-gb1 -b4=5 -d=y
[dstep@nt098 Gblocks_0.91b]$ ls
Documentation  more_alignments  nad3.pir-gb1      nad3.pir-gb1PS
Gblocks        nad3.pir          nad3.pir-gb1.htm  paths
[dstep@nt098 Gblocks_0.91b]$ cat nad3.pir-gb1
```



4. 自分の環境を使いやすくする (5/6)

Gblocksを適切なディレクトリ（今回は ~/local/binを設定）に移動させる。

環境変数 PATH に ~/local/binを追加する。

```
[dstep@nt098 Gblocks_0.91b]$ mkdir ~/local  
[dstep@nt098 Gblocks_0.91b]$ mkdir ~/local/bin  
[dstep@nt098 Gblocks_0.91b]$ cp Gblocks ~/local/bin
```

下記は、打ち間違えるとひどい目にあいます。慎重に！！（不安なら試さない）

```
[dstep@nt098 Gblocks_0.91b]$ echo 'export PATH=$PATH: "/usr/local/pkg/canu/current/bin"' >> ~/.bashrc  
[dstep@nt098 Gblocks_0.91b]$ source ~/.bashrc
```

Gblocksは解凍するだけですぐに使えました。

他の多くのプログラムは、

```
./configure --prefix="~/local/bin"  
make  
make install
```

としないと、動くプログラムが作成されません。

./configure の後ろにオプションとして、--prefix="~/local/bin"

としておくのが、ローカル環境にプログラムをインストールする時のTips(コツ)です。

4. 自分の環境を使いやすくする (6/6)

以上で、自分の環境へ好みのプログラムをインストールする基礎を学びました

1. 環境変数について学びました。特にPATH環境変数を学びました。

プログラムがどのディレクトリに存在するかはPATHに教えなければなりません。

もし自分のプログラムを~/local/binに入れることにした場合は、~/.bashrcファイルに

```
export PATH=$PATH:~/local/bin
```

と記載します。

~/local/bin以外のディレクトリにプログラムが存在する場合は、そのディレクトリを環境変数PATHに追加しましょう。

例えば. nigの大型計算機にすでにインストールされているbowtie2を利用したければ

```
export PATH=$PATH:~/usr/local/pkg/bowtie2/current
```

などすると利用できます。

2. nigの大型計算機に自分でプログラムをインストールする例を学びました。

プログラムのインストール方法は、プログラム毎に様々です。

READMEかINSTALLファイルに記載してあることが多いです。

その手順が **./configure ; make ; make install** の3ステップのものが多いですが、

この場合、./configureをする際に、**./configure --prefix=~/local/bin**

などとする必要がある場合があります。

5. short readのマッピング

ここまでで学んだことは、UNIX系OSの基本コマンドいくつか、大型計算機へのログイン方法、大型計算機へのコマンドの投入方法（キューイングシステム）、必要な生物学データの取得と自分の環境へのアップロード、必要なプログラムを自分の環境へインストールすること、などを学びました。

これらの知識のおさらいとして、実際に配列解析の手順の一例を体験してみましょう。参照配列にshort readの配列をマッピングして見ることにします。

5. マッピング作業 (1/13)

すでにダウンロード済みの下記の配列ファイルをリファレンス配列にマッピングしてみましょう

ERR003990_1.fastq ERR003990_2.fastq

手順は

ゼブラフィッシュのゲノム配列をリファレンス配列として取得しておく。(Ensembl)

今回は下記サイトから、染色体9番の配列だけを取得して利用。

ftp://ftp.ensembl.org/pub/release-91/fasta/danio_rerio/dna/Danio_rerio.GRCz10.dna.chromosome.9.fa.gz

それぞれのfastqファイルのアダプターやクオリティーをfastQCでチェックする

Trimmomaticをインストールしておく

適切な値のクオリティーでフィルタリングする。Trimmomatic

得られたHigh Qualityのreadを、リファレンスにbowtie2でマッピングする。

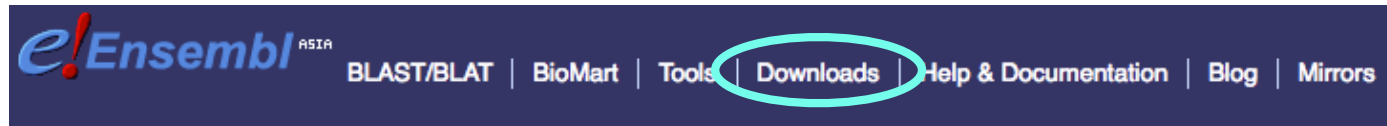
samtools でマッピングの様子を眺める。

samtools でpileupしたファイルを作成する。

5. マッピング作業 (2/13)

ゼブラフィッシュのゲノム配列をリファレンス配列として取得しておく。(Ensembl)

<http://asia.ensembl.org/index.html>



[Download a sequence or region](#)

クリック



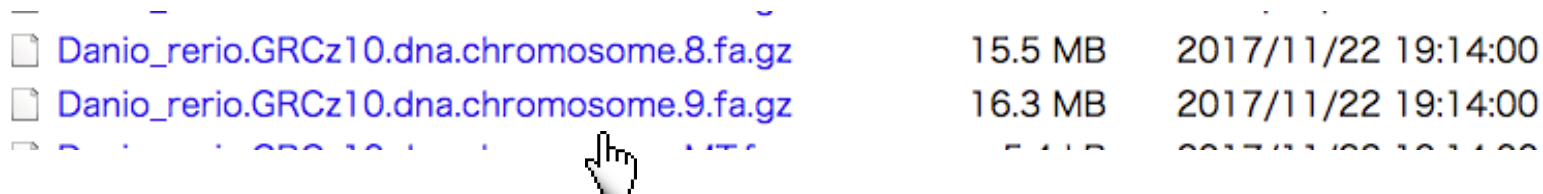
FTP Download クリック

<http://asia.ensembl.org/info/data/ftp/index.html>

Show	10	entries		
★	Species	DNA (FASTA)	cDNA (FASTA)	CD (FASTA)
Y	Human <i>Homo sapiens</i>	FASTA	FASTA	FASTA
Y	Mouse <i>Mus musculus</i>	FASTA	FASTA	FASTA
Y	Zebrafish <i>Danio rerio</i>	FASTA	FASTA	FASTA
	Algerian	FASTA	FASTA	FASTA

Zebrafishの DNA (FASTA)を クリック

ftp://ftp.ensembl.org/pub/release-91/fasta/danio_rerio/dna/



	Danio_rerio.GRCz10.dna.chromosome.8.fa.gz	15.5 MB	2017/11/22 19:14:00
	Danio_rerio.GRCz10.dna.chromosome.9.fa.gz	16.3 MB	2017/11/22 19:14:00
	Danio_rerio.GRCz10.dna.chromosome.10.fa.gz	15.5 MB	2017/11/22 19:14:00

ftp://ftp.ensembl.org/pub/release-91/fasta/danio_rerio/dna/Danio_rerio.GRCz10.dna.chromosome.9.fa.gz

5. マッピング作業 (3/13)

`wget ftp://ftp.ensembl.org/pub/release-91/fasta/danio_rerio/dna/Danio_rerio.GRCz10.dna.chromosome.9.fa.gz`

↑
ダウンロード先のURLはコピペで良い

```
[dtep@nt098 dstep20180126]$ ls
DRR016430.sra                               dat
Danio_rerio.GRCz10.dna.chromosome.9.fa.gz  hemoglobin_b.fa
```

```
[dtep@nt098 dstep20180126]$ gunzip Danio_rerio.GRCz10.dna.chromosome.9.fa.gz
```

```
[dtep@nt098 dstep20180126]$ ls
DRR016430.sra                               dat
Danio_rerio.GRCz10.dna.chromosome.9.fa     hemoglobin_b.fa
```

これでゼブラフィッシュのリファレンスゲノム配列を取得できた

5. マッピング作業 (4/13)

本来はそれぞれのfastqファイルのアダプターやクオリティをfastQCでチェックする

fastQCはご自身のパソコンで行うほうがだいたいうまくいきます。今回はスルー

trimmomaticをインストールする

<http://www.usadellab.org/cms/?page=trimmomatic>

Trimmomatic: A flexible read trimming tool for Illumina NGS data

Citations

Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.

Downloading Trimmomatic

Version 0.36: [binary](#), [source](#) and [manual](#)



Downloading Trimmomatic

Version 0.36: [binary](#), [source](#) and [manual](#)

Quick start

Paired End:

- リンクを新規タブで開く
- リンクを新規ウィンドウで開く
- リンク先のファイルをダウンロード
- リンク先のファイルを別名でダウンロード...
- リンクをブックマークに追加...
- リンクをリーディングリストに追加
- リンクをコピー



——— 続く ———

5. マッピング作業 (5/13)

——trimmomaticのインストール、続き——

```
[dstep@nt098 ~]$ cd ~  
[dstep@nt098 ~]$ mkdir src/  
[dstep@nt098 ~]$ mkdir src/trimmomatic  
[dstep@nt098 ~]$ cd src/trimmomatic
```

```
[dstep@nt098 trimmomatic]$ wget http://www.usadellab.org/cms/uploads/  
supplementary/Trimmomatic/Trimmomatic-0.36.zip
```

↑
ダウンロード先のURLはコピペで良い

```
[dstep@nt098 trimmomatic]$ ls  
Trimmomatic-0.36.zip  
[dstep@nt098 trimmomatic]$ unzip Trimmomatic-0.36.zip  
Archive: Trimmomatic-0.36.zip  
  creating: Trimmomatic-0.36/  
    inflating: Trimmomatic-0.36/LICENSE  
...  
[dstep@nt098 trimmomatic]$ ls  
Trimmomatic-0.36 Trimmomatic-0.36.zip  
[dstep@nt098 trimmomatic]$ cd Trimmomatic-0.36  
[dstep@nt098 Trimmomatic-0.36]$ ls  
LICENSE adapters trimmomatic-0.36.jar  
[dstep@nt098 Trimmomatic-0.36]$ pwd  
/home/dstep/src/trimmomatic/Trimmomatic-0.36
```

<- # このpathを覚えておく

trimmomaticもダウンロードして解凍すればそれで使える

5. マッピング作業 (6/13)

ここまでで、リファレンス配列とクエリーになるRNAseq配列の取得がすみ、
RNAseqのクオリティーフィルターをかけるためのTrimmomaticのインストール
が済んだ。次はTrimmomaticでRNAseqのクオリティーフィルターをかける。

データのあるディレクトリに戻る

```
[dstep@nt098 Trimmomatic-0.36]$ cd ~/dstep20180126
```

小量データのサンプルファイルを作る

```
[dstep@nt098 dstep20180126] head -4000 ERR003990_1.fastq >s1.fastq
```

```
[dstep@nt098 dstep20180126] head -4000 ERR003990_2.fastq >s2.fastq
```

```
[dstep@nt098 dstep20180126] java -jar ~/src/trimmomatic/Trimmomatic-0.36/  
trimmomatic-0.36.jar PE s1.fastq s2.fastq s1.tmf.fastq s1.tmf.unp.fastq  
s2.tmf.fastq s2.tmp.unp.fastq TRAILING:20 MINLEN 36
```

5. マッピング作業 (7/13)

——trimmomaticのインストール、続き——

```
[dstep@nt098 ~]$ cd ~  
[dstep@nt098 ~]$ mkdir src/  
[dstep@nt098 ~]$ mkdir src/trimmomatic  
[dstep@nt098 ~]$ cd src/trimmomatic
```

```
[dstep@nt098 trimmomatic]$ wget http://www.usadellab.org/cms/uploads/  
supplementary/Trimmomatic/Trimmomatic-0.36.zip
```

↑
ダウンロード先のURLはコピペで良い

```
[dstep@nt098 trimmomatic]$ ls  
Trimmomatic-0.36.zip  
[dstep@nt098 trimmomatic]$ unzip Trimmomatic-0.36.zip  
Archive:  Trimmomatic-0.36.zip  
  creating: Trimmomatic-0.36/  
    inflating: Trimmomatic-0.36/LICENSE  
...  
[dstep@nt098 trimmomatic]$ ls  
Trimmomatic-0.36  Trimmomatic-0.36.zip  
[dstep@nt098 trimmomatic]$ cd Trimmomatic-0.36  
[dstep@nt098 Trimmomatic-0.36]$ ls  
LICENSE  adapters  trimmomatic-0.36.jar  
[dstep@nt098 Trimmomatic-0.36]$ pwd  
/home/dstep/src/trimmomatic/Trimmomatic-0.36
```

<- # このpathを覚えておく

trimmomaticもダウンロードして解凍すればそれで使える

5. マッピング作業 (8/13)

trimmomaticを使ってクオリティフィルターをします

下記のコマンドで結果が得られるはずですが、

java -jar trimmomatic.jar PE s_1.fq s_2.fq s_1.filt s_1.unpair.fq s_2.filt.fq s_2.unpair.fq TRAILING:5 MINLEN:35

```
java -jar ~/src/trimmomatic/Trimmomatic-0.36/trimmomatic-0.36.jar \
PE ERR003990_1.fastq ERR003990_2.fastq \
ERR003990_1.trimmomatic.fq ERR003990_1.trimmomatic.unp.fq \
ERR003990_2.trimmomatic.fq ERR003990_2.trimmomatic.unp.fq \
TRAILING:8 MINLEN:36
```

すでに学んだ通り、qsub様のスクリプトを作成してqsubに投げる

```
[dstep@nt098 dstep20180126]$ cat trimmomatic.sh
```

```
#!/bin/sh
#$ -S /bin/sh
#$ -cwd
#$ -l s_vmem=10G,mem_req=10G
#$ -l short
#$ -pe def_slot 1
#$ -o ./log
#$ -e ./log
```

```
source ~/.bashrc
cd /home/dstep/tanomare/dstep20180126
```

```
java -jar ~/src/trimmomatic/Trimmomatic-0.36/trimmomatic-0.36.jar PE ERR003990_1.fastq
ERR003990_2.fastq ERR003990_1.trimmomatic.fq ERR003990_1.trimmomatic.unp.fq
ERR003990_2.trimmomatic.fq ERR003990_2.trimmomatic.unp.fq TRAILING:8 MINLEN:36
```


5. マッピング作業 (9/13)

trimmomaticを使ってクオリティーフィルターをします
すでに学んだ通り、qsub様のスクリプトを作成してqsubに投げる

```
[dstep@nt098 dstep20180126]$ mkdir log
[dstep@nt098 dstep20180126]$ qsub trimmomatic.sh
# 少し時間がかかります
[dstep@nt098 dstep20180126]$ qstat
# qstat を何度か使って、コマンドの進み具合を見ます
[dstep@nt098 dstep20180126]$ ls log
[dstep@nt098 dstep20180126]$ cat log/trimmomatic.sh.e*
```

ERR003990_1.fastqとERR003990_2.fastq の二つの元ファイルから、
クオリティーフィルターをかけた ERR003990_1.trimmomatic.fqとERR003990_2.trimmomatic.fq
#が 得られた。

```
[dstep@nt098 dstep20180126]$ ls ERR*
ERR003990.sra                ERR003990_2.fastq
ERR003990_1.fastq           ERR003990_2.trimmomatic.fq
ERR003990_1.trimmomatic.fq  ERR003990_2.trimmomatic.unp.fq
ERR003990_1.trimmomatic.unp.fq
[dstep@nt098 dstep20180126]$ ls ERR*
```

```
[dstep@nt098 dstep20180126]$ wc ERR003990_*.fq
 21290748   42581496   950083932 ERR003990_1.trimmomatic.fq
   377024    754048    16799544 ERR003990_1.trimmomatic.unp.fq
 21290748   42581496   950198864 ERR003990_2.trimmomatic.fq
   427972    855944    19079832 ERR003990_2.trimmomatic.unp.fq
```

ペアは同数
になっている

5. マッピング作業 (10/13)

必要なファイルが揃いました

```
[dstep@nt098 dstep20180126]$ ls
Danio_rerio.GRCz10.dna.chromosome.9.fa
ERR003990_1.trimmomatic.fq
ERR003990_2.trimmomatic.fq
```

-h オプションをつけると使用方法を表示する

```
[dstep@nt098 dstep20180126]$ /usr/local/bin/bowtie2-build -h
```

```
[dstep@nt098 dstep20180126]$ bowtie2-build -f
Danio_rerio.GRCz10.dna.chromosome.9.fa Danio_rerio.GRCz10.dna.chromosome.9.fa
```

#それほど大きくない配列なので、今回はbowtie2-buildをqsubに投げませんが
こういう場合も本来はqsubコマンドを作成してください。

```
[dstep@nt098 dstep20180126]$ ls
Danio_rerio.GRCz10.dna.chromosome.9.fa
Danio_rerio.GRCz10.dna.chromosome.9.fa.1.bt2
Danio_rerio.GRCz10.dna.chromosome.9.fa.2.bt2
Danio_rerio.GRCz10.dna.chromosome.9.fa.3.bt2
Danio_rerio.GRCz10.dna.chromosome.9.fa.4.bt2
Danio_rerio.GRCz10.dna.chromosome.9.fa.rev.1.bt2
Danio_rerio.GRCz10.dna.chromosome.9.fa.rev.2.bt2
```

5. マッピング作業 (11/13)

必要なファイルが揃いました

```
[dstep@nt098 dstep20180126]$ cat bowtie2.qsub
```

```
#!/bin/sh
```

```
#$ -S /bin/sh
```

```
#$ -cwd
```

```
#$ -l s_vmem=10G,mem_req=10G
```

```
#$ -l short
```

```
#$ -pe def_slot 1
```

```
#$ -o ./log
```

```
#$ -e ./log
```

```
source ~/.bashrc
```

```
bowtie2 -q -x Danio_rerio.GRCz10.dna.chromosome.9.fa -1  
ERR003990_1.trimmomatic.fq -2 ERR003990_2.trimmomatic.fq  
-S danio_ERR003990.sam
```

→ # bowtie2.qsub

-Sオプションに与えた名前が出力

```
[dstep@nt098 dstep20180126]$ qsub bowtie2.qsub
```

```
[dstep@nt098 dstep20180126]$ wc danio_ERR003990.sam
```

5. マッピング (12/13)

samtoolsでの一連の作業

```
[dstep@nt098 dstep20180126]$ samtools view -bt Danio_rerio.GRCz10.dna.chromosome.9.fa -o
danio_ERR003990.bam danio_ERR003990.sam
[dstep@nt098 dstep20180126]$ samtools sort -o danio_ERR003990.sorted.bam danio_ERR003990.bam
[dstep@nt098 dstep20180126]$ samtools index danio_ERR003990.sorted.bam
[dstep@nt098 dstep20180126]$ samtools faidx Danio_rerio.GRCz10.dna.chromosome.9.fa
[dstep@nt098 dstep20180126]$ samtools tview danio_ERR003990.sorted.bam
Danio_rerio.GRCz10.dna.chromosome.9.fa
```

[illegible]

キーボードのLとHで配列上を見ることができる。

```
# キーボードのqで終了することができる
```

5. マッピング (13/13)

以上で、

リファレンスとなるゲノム配列にshort readのクエリーをマッピングする手順を学びました。

実際には、この後さらに、マッピングの深さの分布やエラーやSNPの分布をpileupして

データを統計的に解析する (集団遺伝学的解析など)、マッピングによって得られたピークを検出する

(ChIPSeq、DNase-seq、ATAC-seqなど)、マッピング領域を取り出す (RNAseqによる遺伝子予測など)、様々な解析が必要になります。

これら個別の解析方法についてはお伝えした参考図書が参考になるでしょう。

6. キューイングシステム利用のための簡単なBASHプログラミング

ここまでで学んだqsubコマンドの利用方法は非常に初歩的なもののみです。ご自身の解析時間の短縮のためにも、大型計算機のリソースを有効利用するためにも、より効率的なshellプログラムの書き方を学びましょう。

6. キューイングシステム利用のための簡単なBASHプログラミング (1/2)

```
[dstep@nt098 dstep20180126]$ cat bowtie2_multi.qsub
```

```
source ~/.bashrc
cd /home/dstep/tanomare/dstep20180126

arr=( ERR003990_1.trimmomatic.fq ERR003990_2.trimmomatic.fq ERR003991_1.fastq
ERR003991_2.fastq ERR003992_1.fastq ERR003992_2.fastq )

i=$(( $SGE_TASK_ID - 1 ))          # <- $SGE_TASK_ID という変数を利用する
j=$(( $i * 2 ))
k=$(( $j + 1 ))

bowtie2 -q -x Danio_rerio.GRCz10.dna.chromosome.9.fa -1 ${arr[$j]} -2 ${arr[$k]} -S danio.${arr[$j]}.pe.sam
```

↓ # \$SGE_TASK_ID に-tで与えた数値が順に入る

```
[dstep@nt098 dstep20180126]$ qsub -t 1-3 bowtie2_multi.qsub
Your job-array 10398586.1-3:1 ("bowtie2_multi.qsub") has been submitted
```

```
[dstep@nt098 dstep20180126]$ qstat
```

job-ID	prior	name	user	state	submit/start at	queue	jclass	slots	ja-task-ID
10397226	0.25133	QLOGIN	dstep	r	01/19/2018 13:12:54	login.q@nt098i		1	
10398580	0.25011	QLOGIN	dstep	r	01/19/2018 16:39:45	login.q@nt099i		1	
10398586	0.25000	bowtie2_mu	dstep	r	01/19/2018 16:58:54	short.q@nt115i		1	1
10398586	0.25000	bowtie2_mu	dstep	r	01/19/2018 16:58:54	short.q@nt115i		1	2
10398586	0.25000	bowtie2_mu	dstep	r	01/19/2018 16:58:54	short.q@nt115i		1	3

6. キューイングシステム利用のための簡単なBASHプログラミング (2/2)

下記三つのファイルが得られた。

```
danio.ERR003990_1.trimmomatic.fq.pe.sam  
danio.ERR003991_1.fastq.pe.sam  
danio.ERR003992_1.fastq.pe.sam
```


7. (時間に余裕があれば) 描画のための簡単なRプログラミング (1/9)

Rについては当日、講演者が作業をお見せするのみにいたします。