バクテリアゲノムの解析実習

de novo アッセンブリ・ゲノムアノテーション

2018.3.23
「先進ゲノム支援」情報解析講習会

谷澤靖洋 / 国立遺伝学研究所 大量遺伝情報研究室
/ 先進ゲノム支援特任研究員

本日の講習内容

de novo アッセンブリ

乳酸菌 *Lactobacillus oligofermentans* の新規ゲノム解読

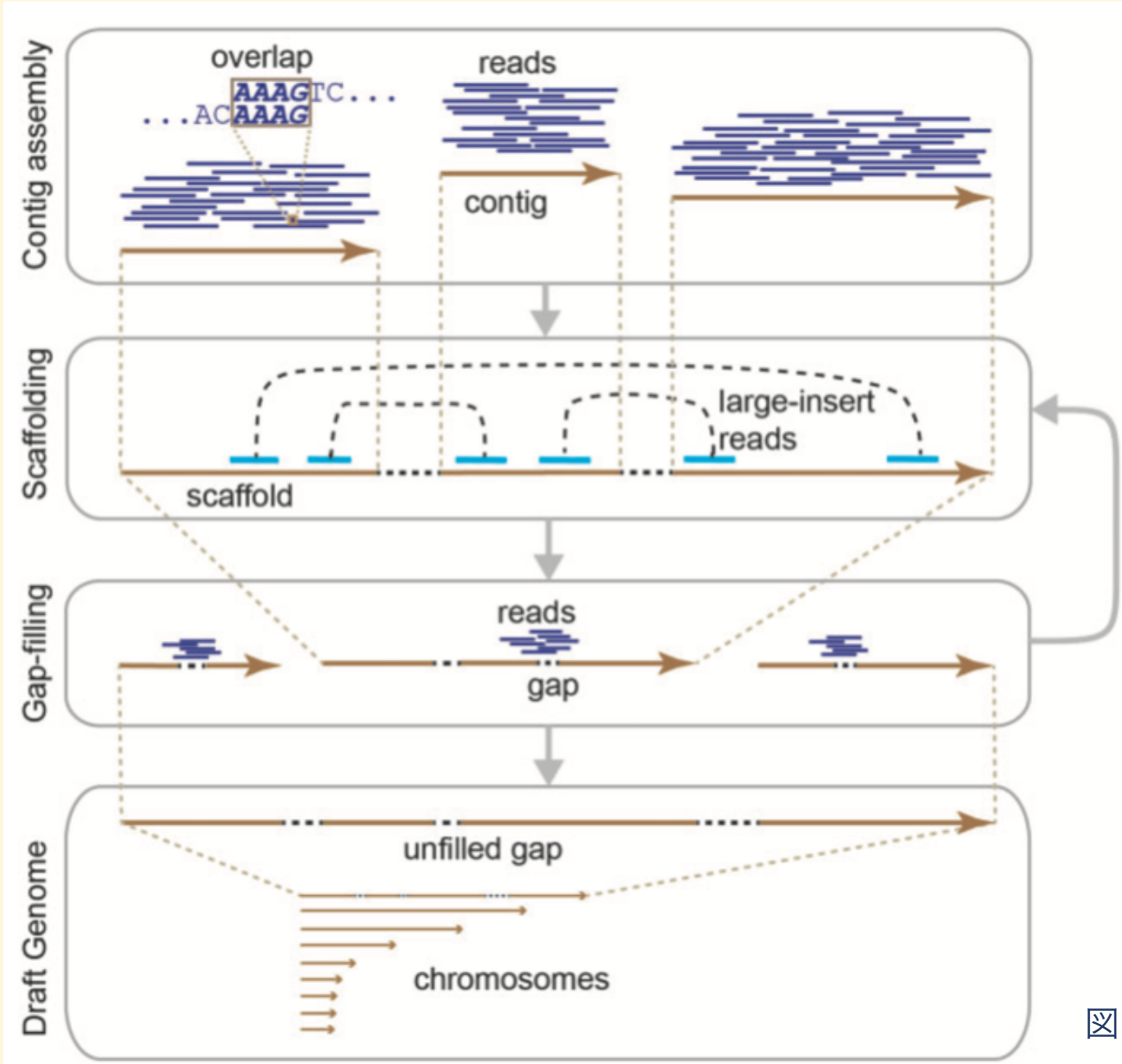
DDBJ Sequence Read Archive (DRA) の公開データを利用
(SRR1151187)

Platanus（Kajitani, et al. 2014) を用いてドラフトゲノムを構築

ゲノムアノテーション

微生物ゲノムアノテーションパイプラインDFASTを用いて
遺伝子領域の予測・遺伝子機能予測 (Tanizawa, et al. 2018)

de novo アッセンブリの一般的な流れ



コンティグ

↓

スキャッフォールド

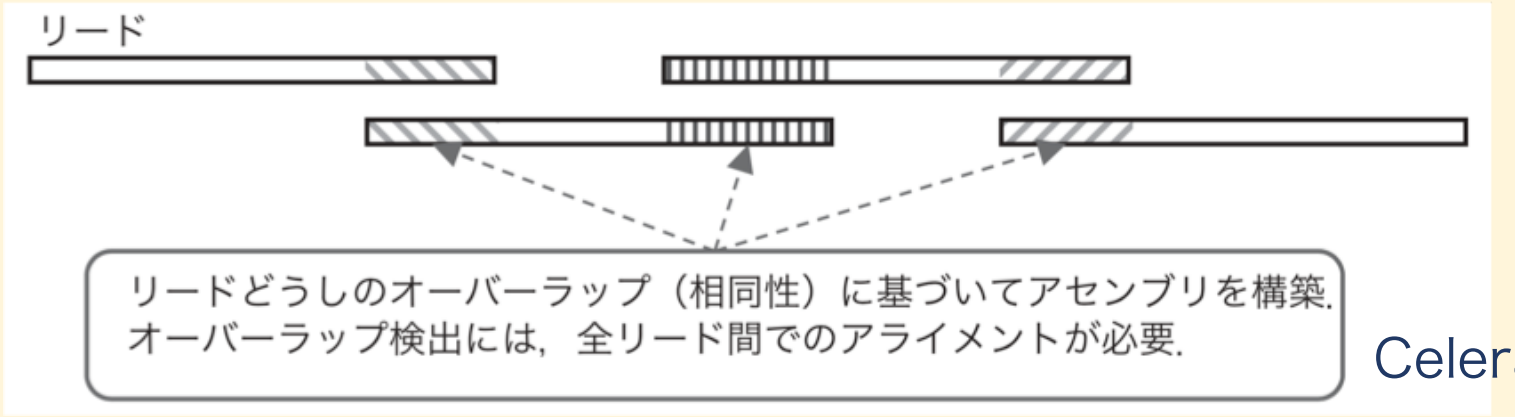
↓

ギャップ埋め

図：Sohn and Nam, 2018

de novo アッセンブリアルゴリズムの二大潮流

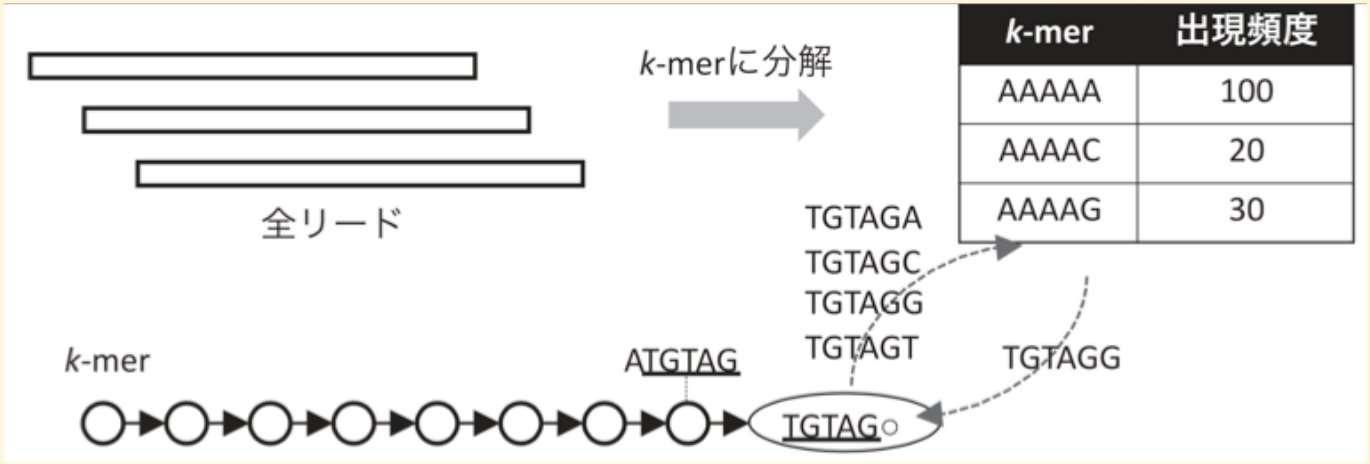
Overlap Layout Concensus (OLC)



（例）
CAP
PHRAP
Celera Assembler

ロングリードの時代になり再評価

de Bruijn graph (DBG)



（例）
EULER
Velvet
Trinity
Platanus

ショートリードのアセンブリに最適

図. 野口, 生物工学 95(9), 2017

de Brujin グラフによるアッセンブルの考え方

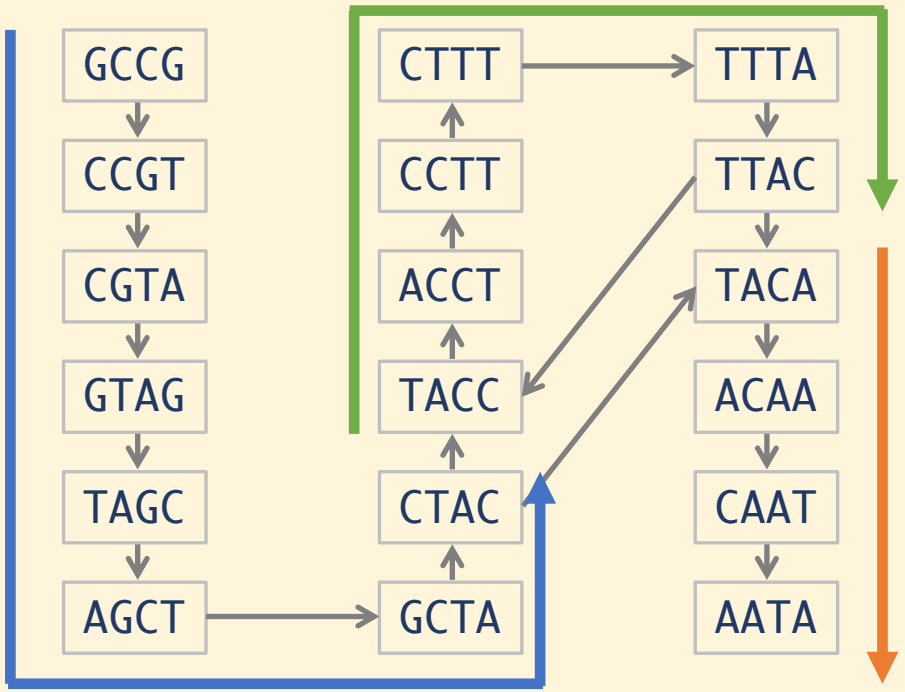
GCCGTAGCTACCTTTACAATA　ゲノム配列

```
GCCGTAGCT    AGCTACC    GCTACCTTT    CCTTTAC    CTTTACAATA
```

リード配列

```
GCCG         AGCT       GCTA         CCTT       CTTT
 CCGT         GCTA       CTAC         CTTT       TTTA
  CGTA         CTAC       TACC         TTTA       TTAC
   GTAG         TACC       ACCT         TTAC       TACA
    TAGC                    CCTT                    ACAA
     AGCT                    CTTT                    CAAT
                                                      AATA
```

K-mer に分解
(*K*=4)



各*K-mer* を頂点

K-1のオーバラップを利用し、
*K-mer*間のつながりを辺で表す

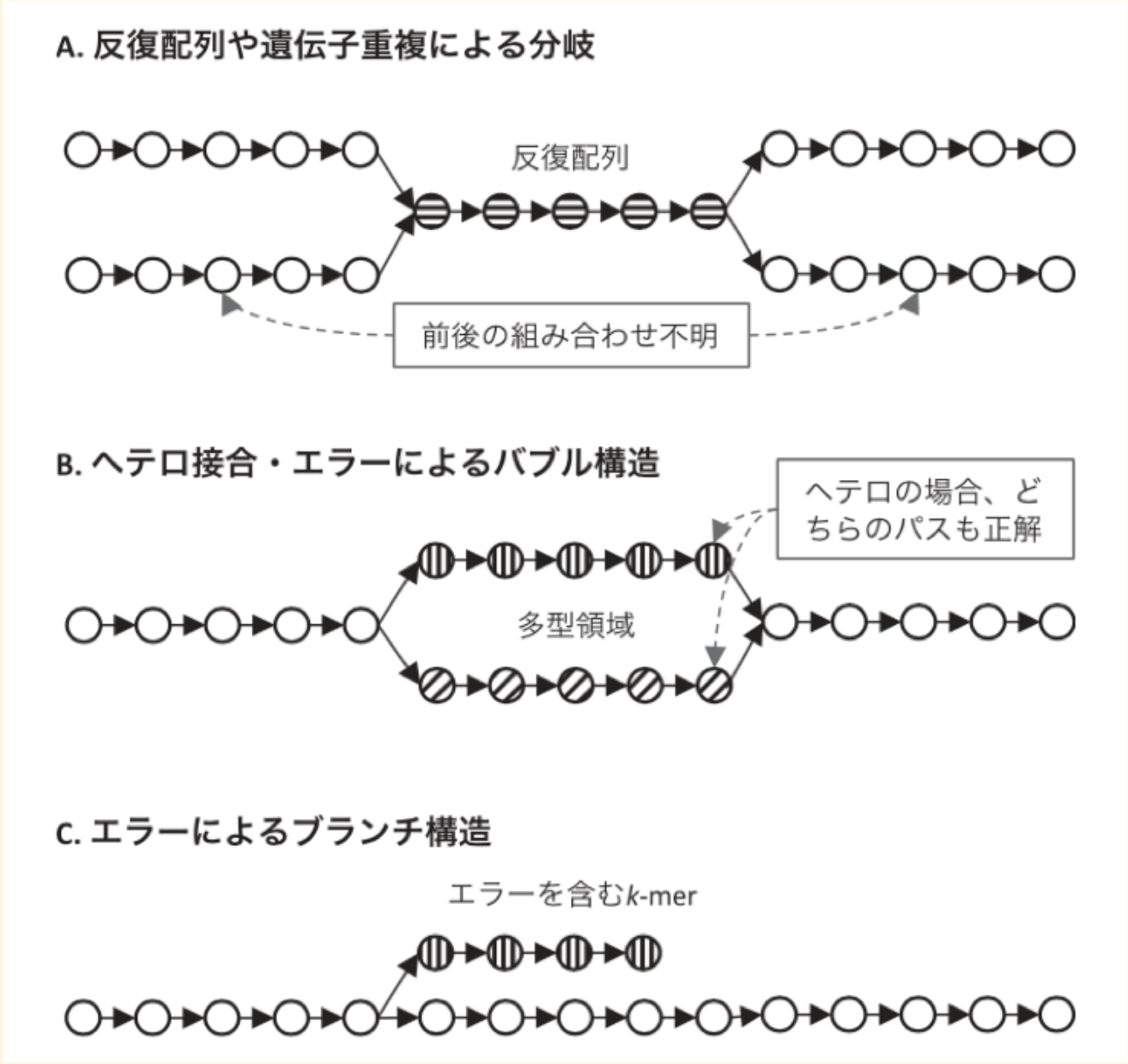
再構築されたコンティグ

GCCGTAGCTAC

TACCTTTAC

TACAATA

グラフを複雑にさせる要因

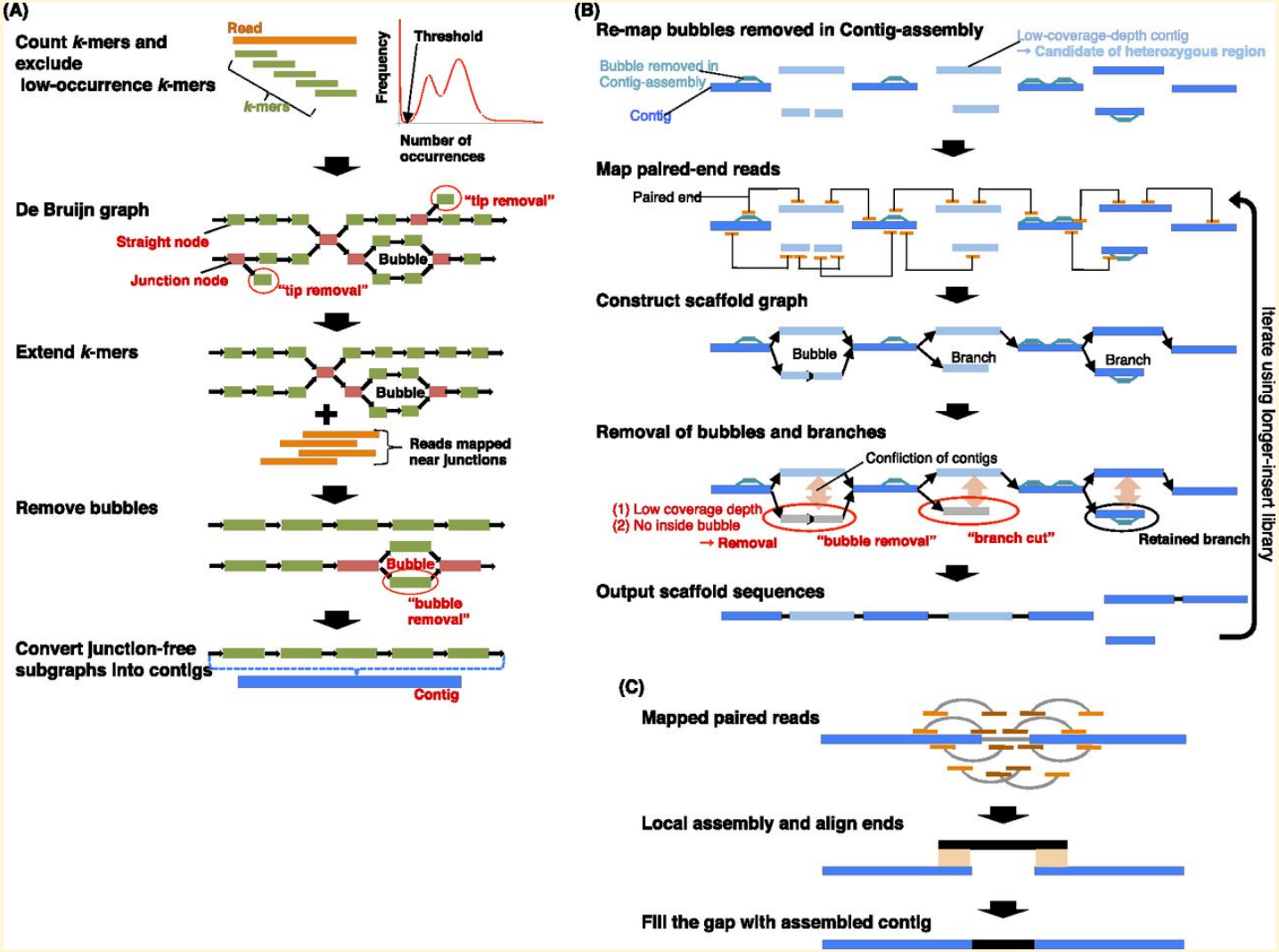


Platanus について

旧「ゲノム支援」において開発（Kajitani et al. 2014）

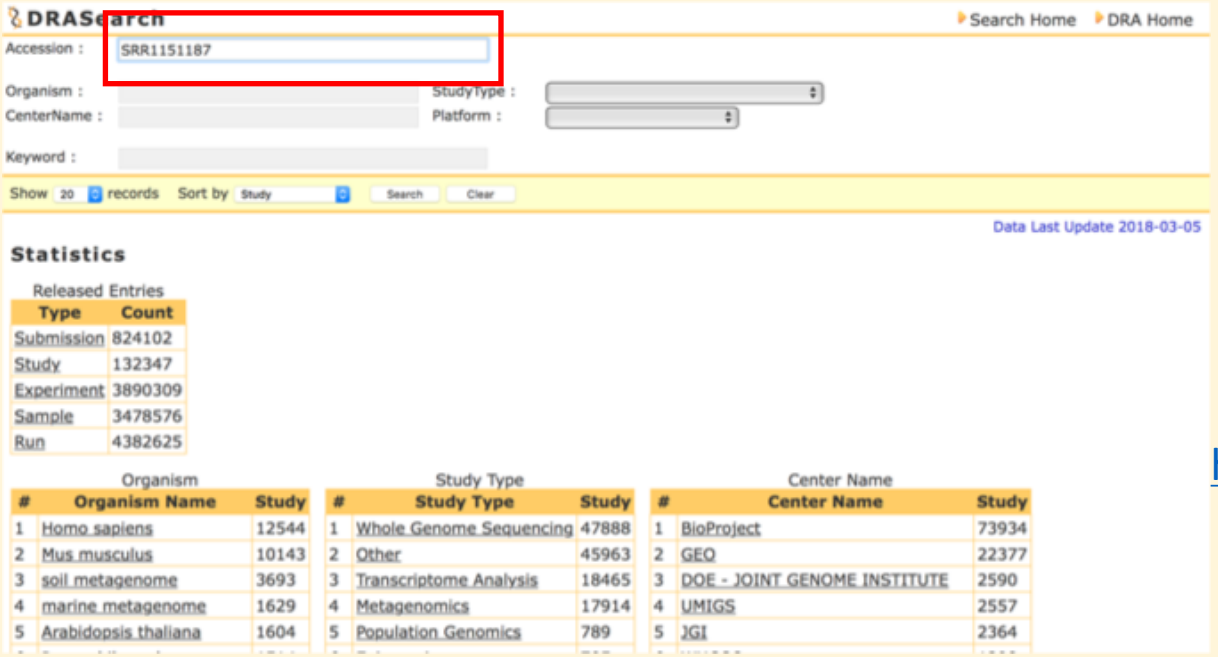
複数の *K-mer* 値を考慮してアッセンブリを最適化

Heterozygosityの高いゲノムのアセンブルに強い



今回使用するデータ

Zheng et al (2015)で使用された*Lactobacillus oligofermentans*のデータ（SRR1151187）



DRA ウェブサイト

↓

http://ddbj.nig.ac.jp/DRASearch/

↓

アクセッション番号で検索

↓

FTPサーバーから
ダウンロード

実習：データの取得

作業ディレクトリの作成（assembly and annotationの略でaaaとします）

```
mkdir aaa
cd aaa
```

データの取得

URLが長いので改行されているように見えますが、実際にはwgetの後に改行はありません。

```
wget
ftp://ftp.ddbj.nig.ac.jp//ddbj_database/dra/fastq/SRA117/SRA117449/SRX45628
7/SRR1151187_1.fastq.bz2
wget
ftp://ftp.ddbj.nig.ac.jp//ddbj_database/dra/fastq/SRA117/SRA117449/SRX45628
7/SRR1151187_2.fastq.bz2
```

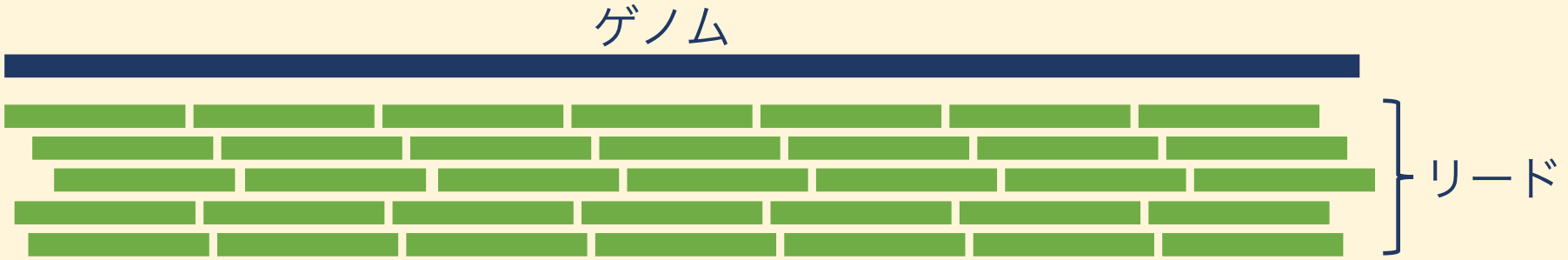
講習用データの準備

それぞれのファイルの先頭から200万行（50万リード）を取り出す

```
bzcat SRR1151187_1.fastq.bz2 | head -2000000 > SRR1151187_1.1M.fastq
bzcat SRR1151187_2.fastq.bz2 | head -2000000 > SRR1151187_2.1M.fastq
```

カバレッジの計算

1塩基あたり何回シークエンスされているか？



この場合はカバレッジは 5x

$$\text{カバレッジ} = \frac{\text{リード数} \times \text{1リードあたり長さ}}{\text{ゲノムサイズ}}$$

$$= \frac{1\text{M} \times 100\text{ bp}}{2\text{Mbp}}$$

$$= 50\text{x}$$

実習：Platanus のダウンロード

Platanusの取得
実際には、Platanus配布元のウェブサイトにいき、URLをコピーして貼り付ける。
-O オプションでダウンロードしたファイルの名称を変更する。

```
wget http://platanus.bio.titech.ac.jp/?ddownload=145 -O platanus
```

実行権限の付与

```
chmod a+x platanus
```

意味：全てのユーザー(a)に対して、実行権限(x)を付加(+)する

コマンドが実行できるか確認

```
./platanus
```

カレントディレクトリにあるコマンドを実行するには、明示的に ./ をつける必要あり。

出力結果

```
Platanus version: 1.2.4
./platanus

Usage: platanus Command [options]

Command: assemble, scaffold, gap_close
```

実習：Platanus の実行

本講習用に用意した実行スクリプトを取得します。

```
wget https://www.evernote.com/shard/s205/sh/4b2497a5-f63a-42d5-afcc-ad07b7376ede/2afddc0e11c6d81e/res/346d91e8-76d9-4f1a-9624-c2b0a93f0ab5/run_platanus.sh
```

スパコンへのジョブ投入

```
qsub run_platanus.sh
```

実行ファイルの中身

```
#$ -S /bin/bash
#$ -pe def_slot 4
#$ -cwd
#$ -l mem_req=4G,s_vmem=4G
#$ -l short
READ1=SRR1151187_1.1M.fastq
READ2=SRR1151187_2.1M.fastq
FILE_PREFIX=SRR1151187

./platanus assemble -t 4 -m 16 -o ${FILE_PREFIX} -f $READ1 $READ2

./platanus scaffold -t 4 -o ${FILE_PREFIX}¥
 -c ${FILE_PREFIX}_contig.fa -b ${FILE_PREFIX}_contigBubble.fa¥
 -IP1 $READ1 $READ2

./platanus gap_close -t 4 -o ${FILE_PREFIX}¥
 -c ${FILE_PREFIX}_scaffold.fa¥
 -IP1 $READ1 $READ2
```

実習：アセンブル後の処理

本講習用に用意した短い配列を除くPythonスクリプト取得します。

```
wget https://www.evernote.com/shard/s205/sh/4b2497a5-f63a-42d5-afcc-ad07b7376ede/2afddc0e11c6d81e/res/f986e7fa-e95f-4204-99c4-7582452cee46/fastaLengthFilter.py
```

200塩基未満の配列を除去し、配列の名称変更およびソートを行う。

```
python fastaLengthFilter.py SRR1151187_gapClosed.fa 200 > SRR1151187_200.fa
```

本講習用に用意したアセンブル結果の統計値を求めるPythonスクリプト取得します。

```
wget https://www.evernote.com/shard/s205/sh/4b2497a5-f63a-42d5-afcc-ad07b7376ede/2afddc0e11c6d81e/res/8f2b14b9-8725-4b9e-8020-100a4d07ed62/fasta_stat.py
```

実行

```
python fasta_stat.py SRR1151187_200.fa
```

結果

```
TOTAL SEQUENCE LENGTH bp: 1788584
TOTAL SEQUENCE NUMBER #: 13
LENGTH of 10 LONGEST SEQUENCES: [574295, 571013, 299830, 168523,
                                 106529, 63258, 1415, 1114, 1019, 630]
N50: 571013
N ratio: 0.011070%
```

N50

ドラフトゲノムの完成度を評価する指標

配列を長いものから順に足していったときに、全長の50%に達した時の配列の長さ

（配列長を考慮した加重平均）

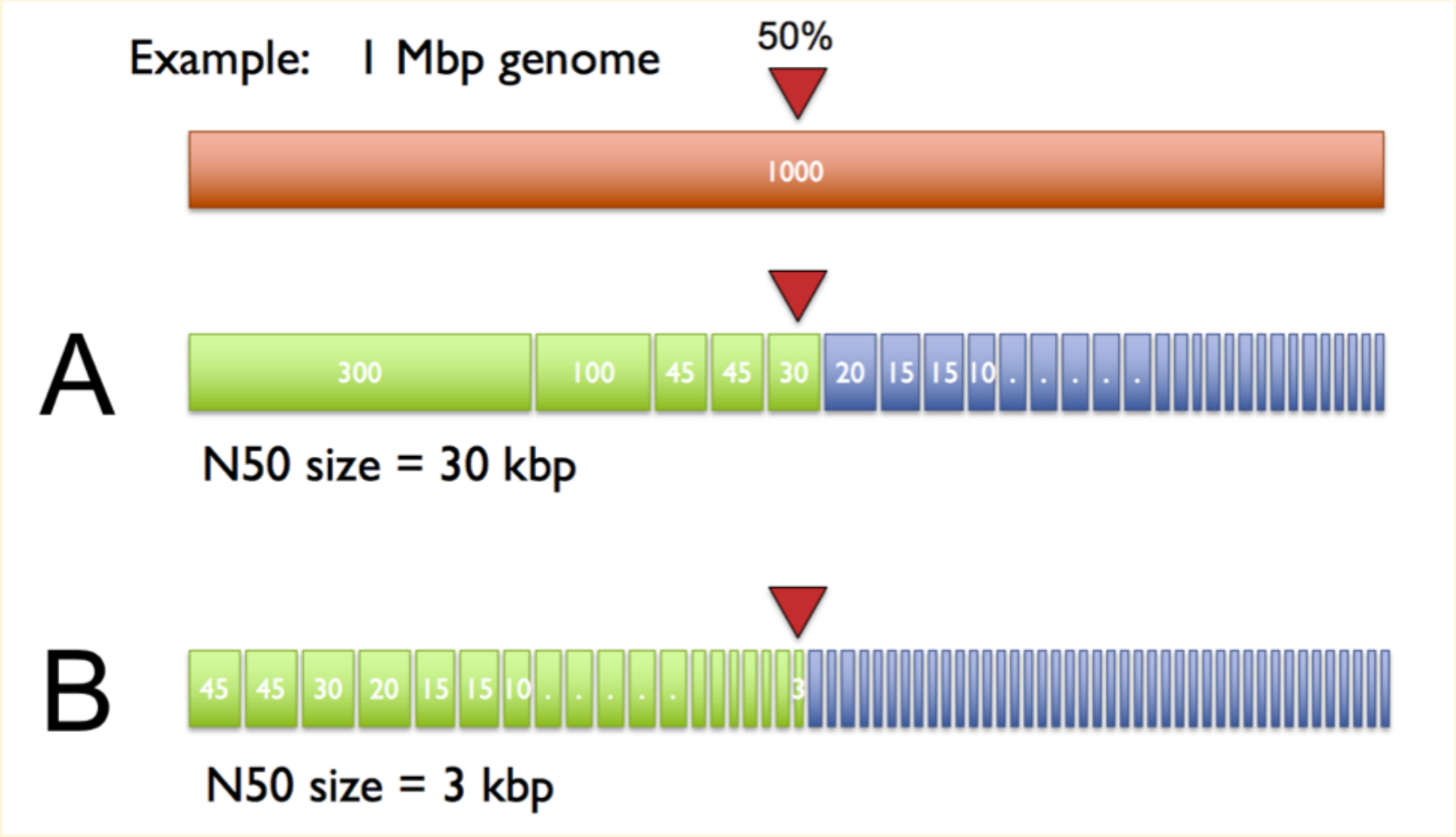
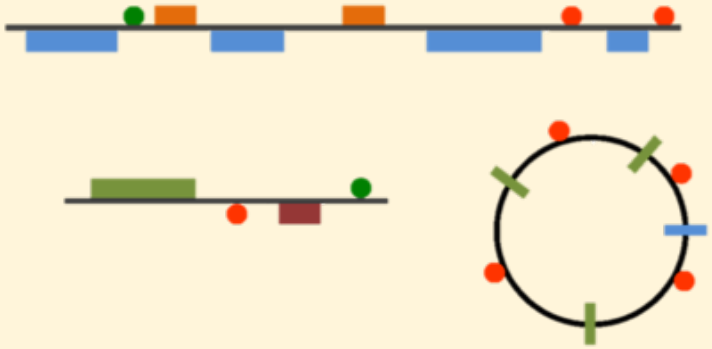


図. M. Schatz http://schatz-lab.org/appliedgenomics2018/lectures/03.GenomeAssembly.pdf

ゲノムアノテーションとは

公共データベースへの塩基配列登録



Sequence data

Data submission

Prerequisite for paper

Public sequence databases

Genome annotation / data submission pipelines

PGAP

1~2 weeks to get results

Limited for GenBank submitters

NCBI

GenBank

Microbial Genome Annotation Pipeline **MiGAP**

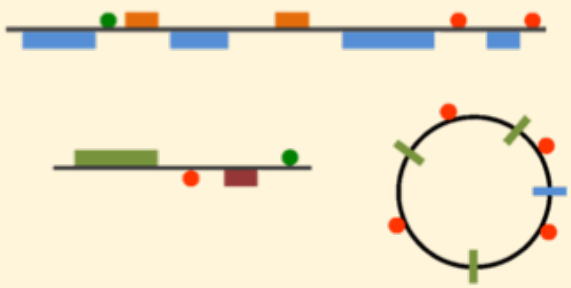
Intensive manual curation

DDBJ

DNA Data Bank of Japan

微生物ゲノムアノテーションパイプライン DFAST

DDBJ Fast Annotation and Submission Tool



DFAST

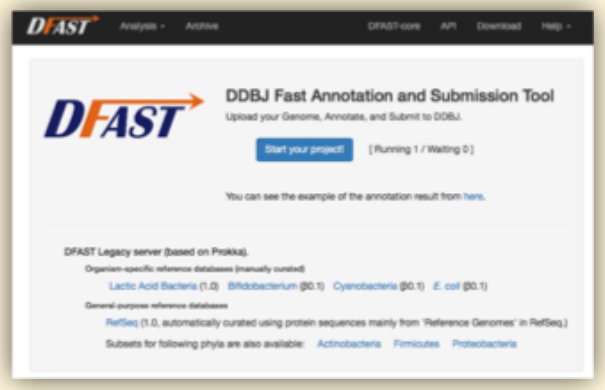
Prokaryotic genome annotation

Data submission to DDBJ

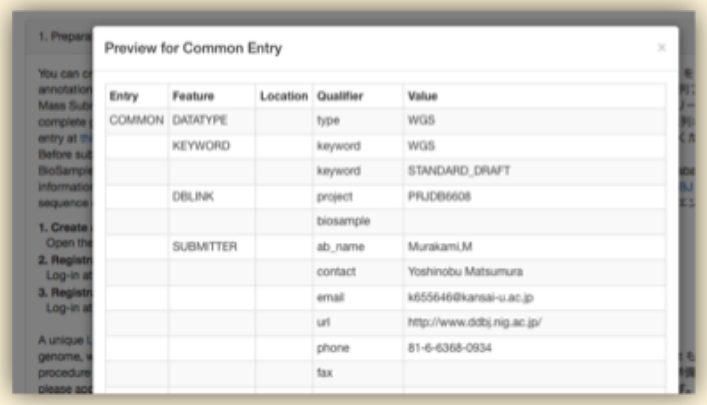
Fast, flexible, and powerful

微生物ゲノムアノテーションパイプライン DFAST

Graphical user interface for beginners

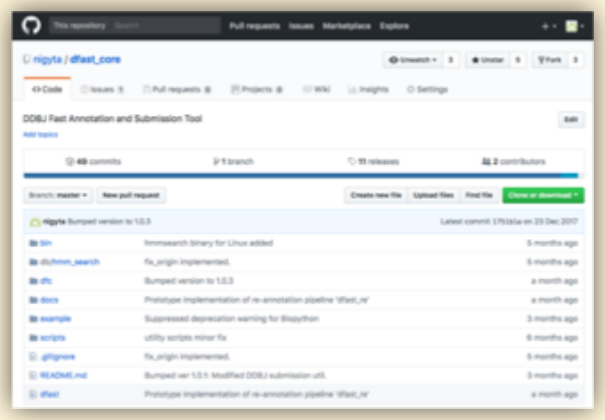


https://dfast.nig.ac.jp



Create DDBJ submission file using online editor

Command operations for experts (DFAST-core)

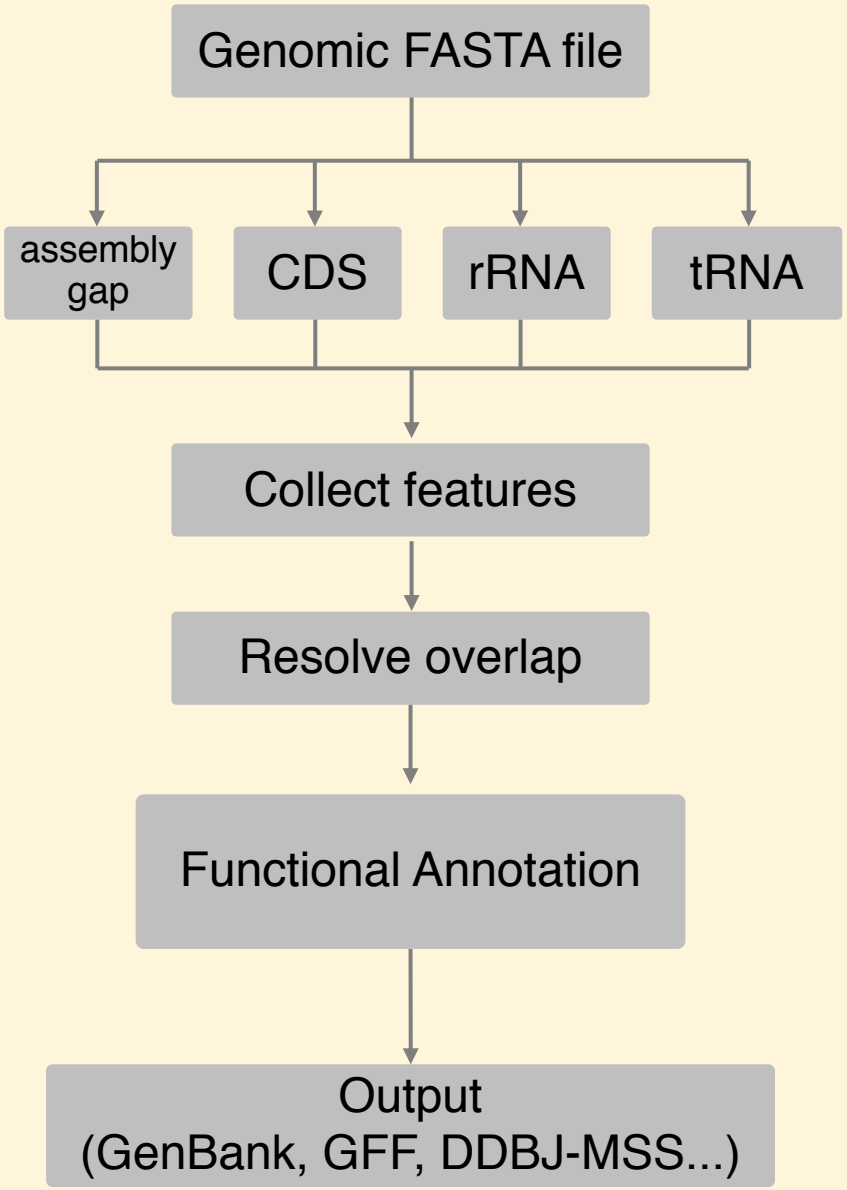


Sample usage

```
dfast --genome your_genome.fna
      --config sample.cfg
```

Stand-alone version available for download (https://github.com/nigyta/dfast_core)

アノテーションの流れ



Structural annotation phase

de facto standard gene prediction tools

parallel processing

Functional annotation phase

Ultrafast homology search using GHOSTX (Suzuki et al. 2014)

- 10 times faster

Small, but well-curated references

- Default database constructed from 120 representative genomes
- Optional organism-specific database

Pseudogene detection

Flexible and customizable

実習：DFAST-coreのインストール準備

インストールの必須要件を確認

https://github.com/nigyta/dfast_core

Prerequisites

- **Python (3.4- or 2.7)**
  DFAST is developed in Python 3.6 and runs both on Python 3.4 or later and Python 2.7.
- **BioPython package**
  You can install this with the Python package management tool `pip`:

  ```
  (sudo) pip install biopython
  ```

  If `pip` is not available, please follow the instruction of BioPython.
- **'futures' and 'six' packages (required only on Python 2.7)**
  DFAST uses the `concurrent.futures` module for multiprocessing and the `six` module for compatibility with Python 2 and 3. To run on Python 2.7, you need to install them:

  ```
  (sudo) pip install futures six
  ```

- **Perl and Java**
  Some of the external programs called from DFAST depend on Perl or Java. Basically, they work with the pre-installed versions on your system.
  For **RedHat/CentOS/Fedora**, the Time::Piece module should be installed:

  ```
  sudo yum install perl-Time-Piece
  ```

実習：DFAST-coreのインストール

ダウンロード

```
wget https://github.com/nigyta/dfast_core/archive/1.0.5.tar.gz
```

解凍しディレクトリに移動

```
tar xvfz 1.0.5.tar.gz
cd dfast_core-1.0.5/
```

tarコマンドのオプションは、x=extract, v=verbose, z=gz圧縮ファイルの場合に指定、
f=ファイルの指定を意味します。事前にアーカイブの中身を確認する “tar tf” も便利です。

ヘルプが表示されるか確認

```
python3 dfast -h
```

スパコンにインストールされているver 2系のpythonが古いので、ver 3系を使います。

参照データベースのダウンロード

```
python3 scripts/file_downloader.py --protein dfast
```

動作確認

```
python3 dfast --config example/test_config.py
```

RESULT_TESTディレクトリに結果が出力される

実習：DFAST-coreの実行

元のディレクトリに戻ります

```
cd ..
```

Javaの環境変数を設定しておきます。

```
export _JAVA_OPTIONS="-Xmx256m -XX:ParallelGCThreads=1"
```

スパコンでJavaで作られたプログラムを動作させようとするとエラーが生じます。
これを回避するためにJava実行時のオプションを環境変数で指定しておきます。
参考）https://sc2.ddbj.nig.ac.jp/index.php/programming#link15

実行します

```
python3 dfast_core-1.0.5/dfast --genome SRR1151187_200.fa --out result --
no_cdd --no_hmm
```

結果は result ディレクトリ内に出力されます。