

RNA-Seqによる 大量発現データ解析の基礎・応用

国立研究開発法人 農業・食品産業技術総合研究機構

次世代作物開発研究センター

高度解析センター（兼任）

川原 善浩

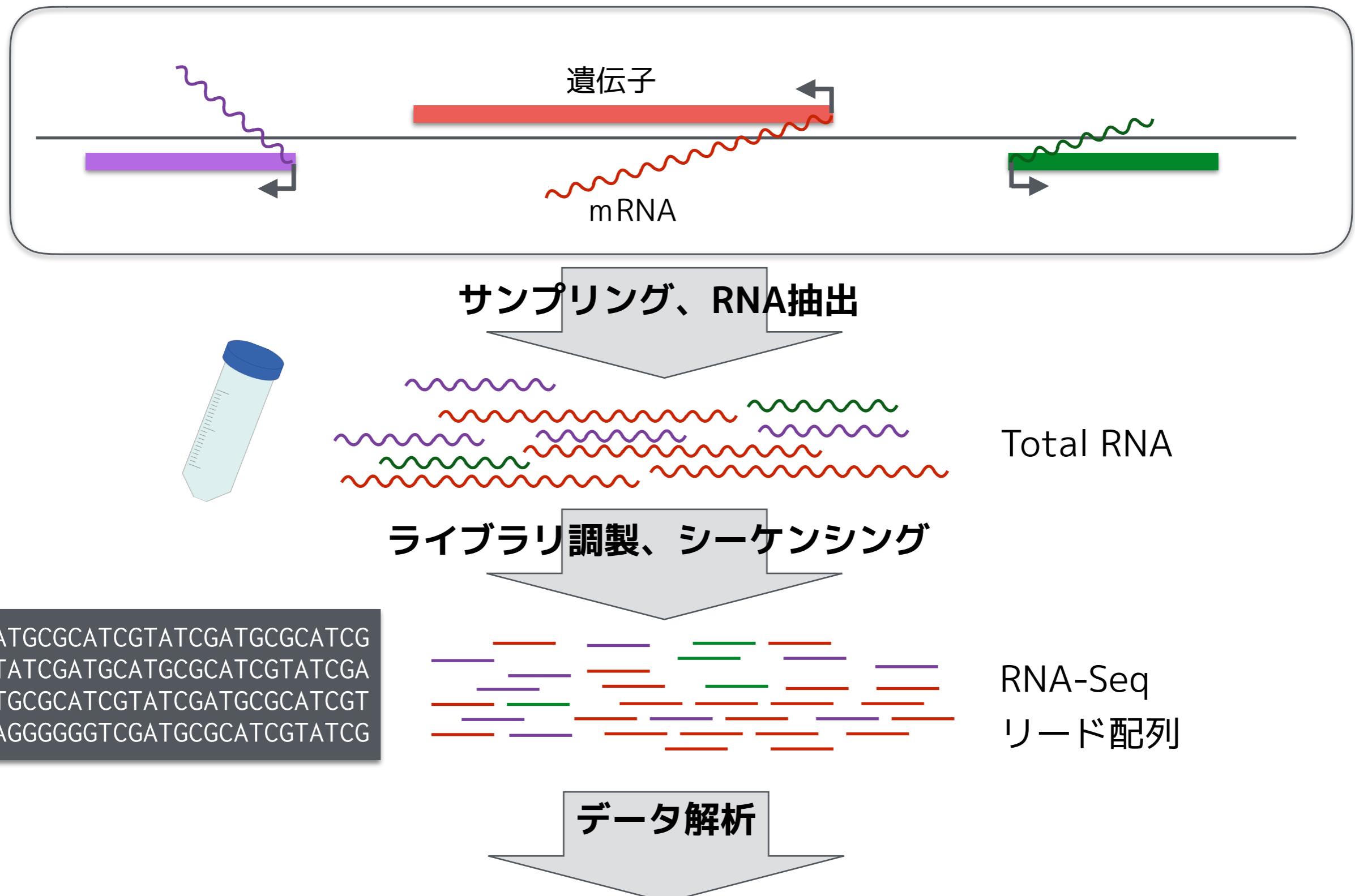


y.kawahara@affrc.go.jp



@YoshiKawahara

RNA-Seqとは

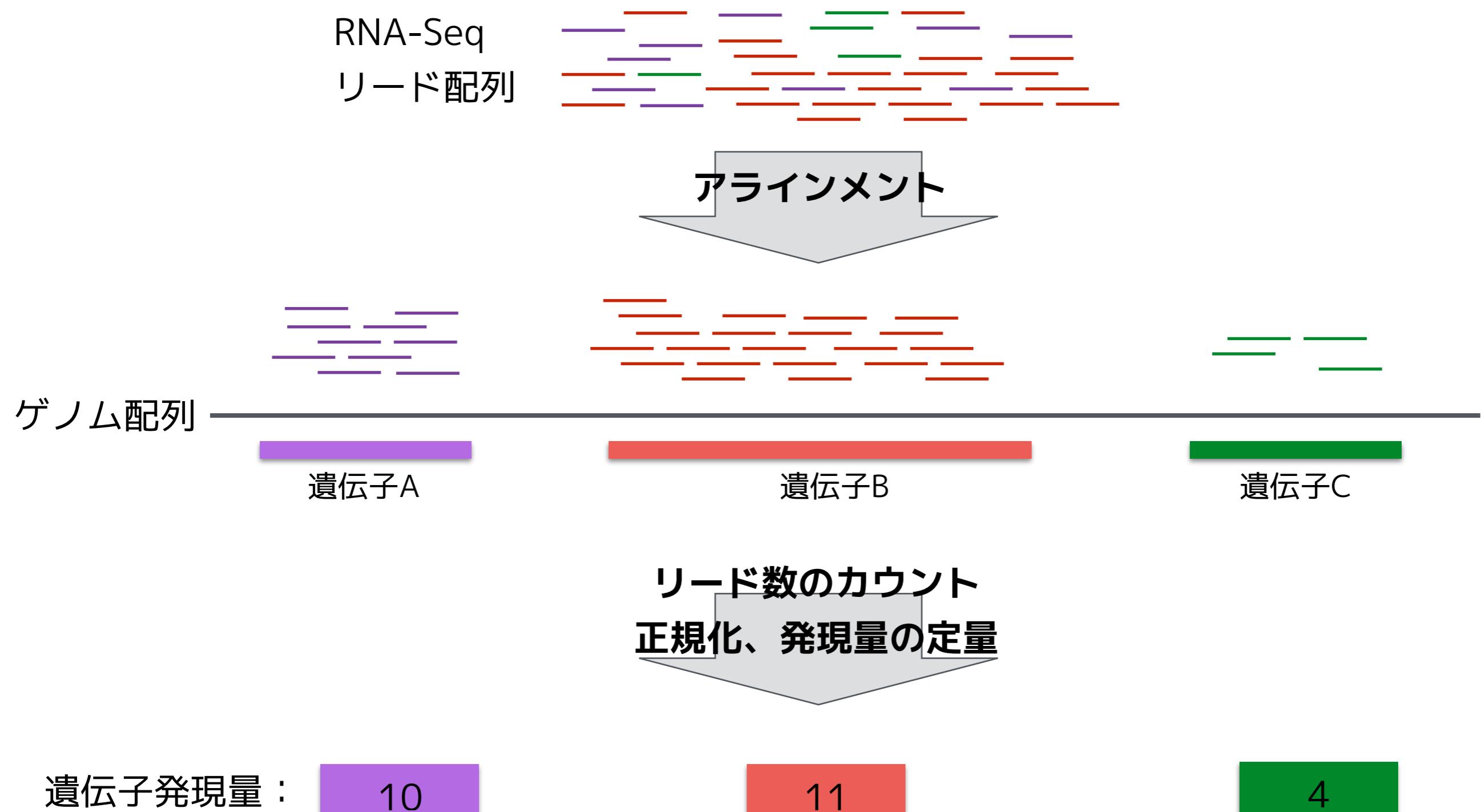


発現している遺伝子カタログの作成、
遺伝子発現解析など、様々な目的に利用可能

手法1：ゲノム配列や遺伝子アノテーションを用いた遺伝子発現解析



RNA-Seqリードをゲノム配列にアラインメント（マッピング）し、
遺伝子ごとにマップされたリードをカウントし、遺伝子発現を定量する。



遺伝子発現量：

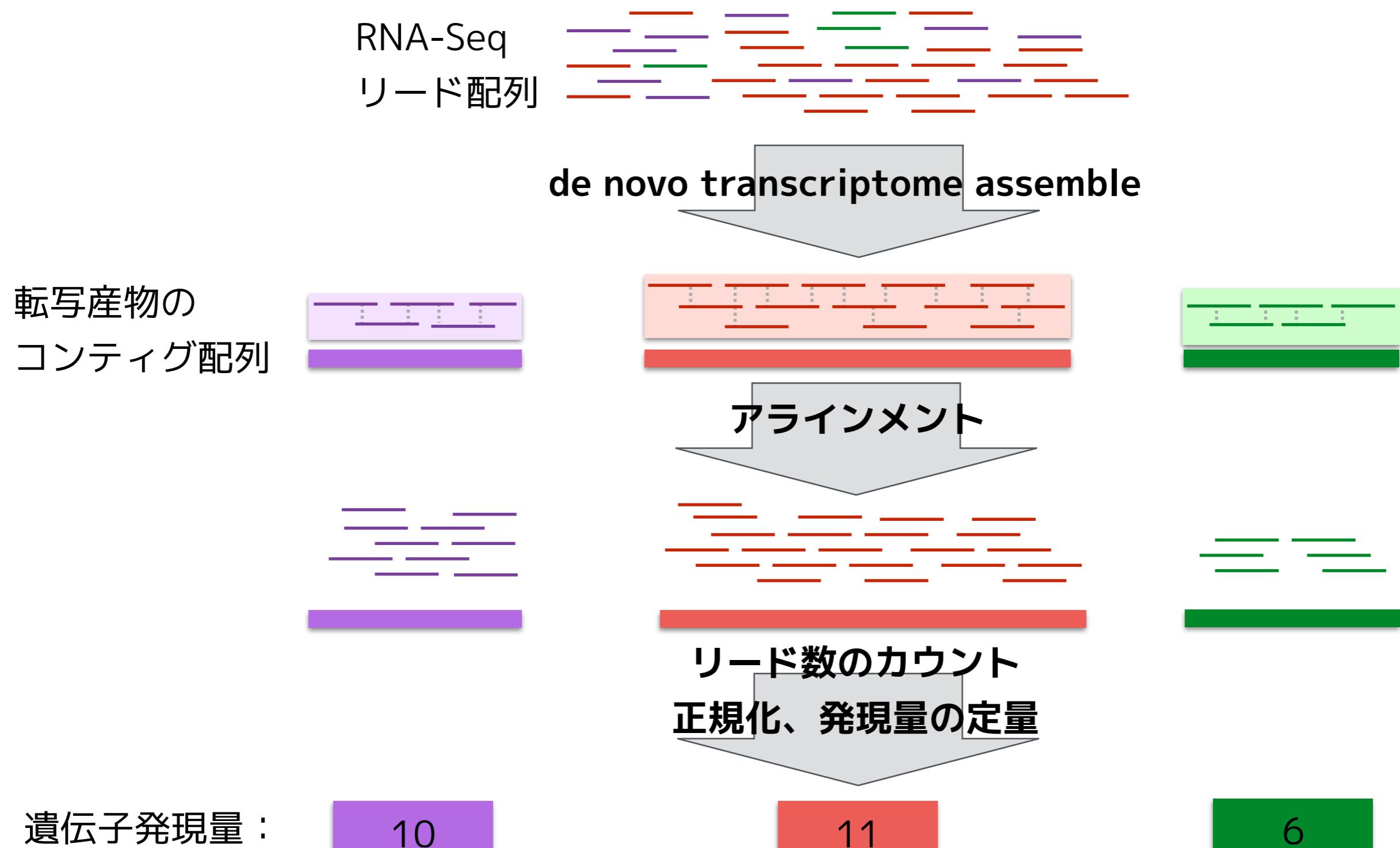
10

11

4

手法2：de novo transcriptome assemble法を用いた遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物の全長配列を再構築する。
アセンブルされた転写産物配列上にRNA-Seqリードをアラインメントし、転写産物ごとにリード数をカウントすることにより遺伝子発現を定量する。



遺伝子発現量：

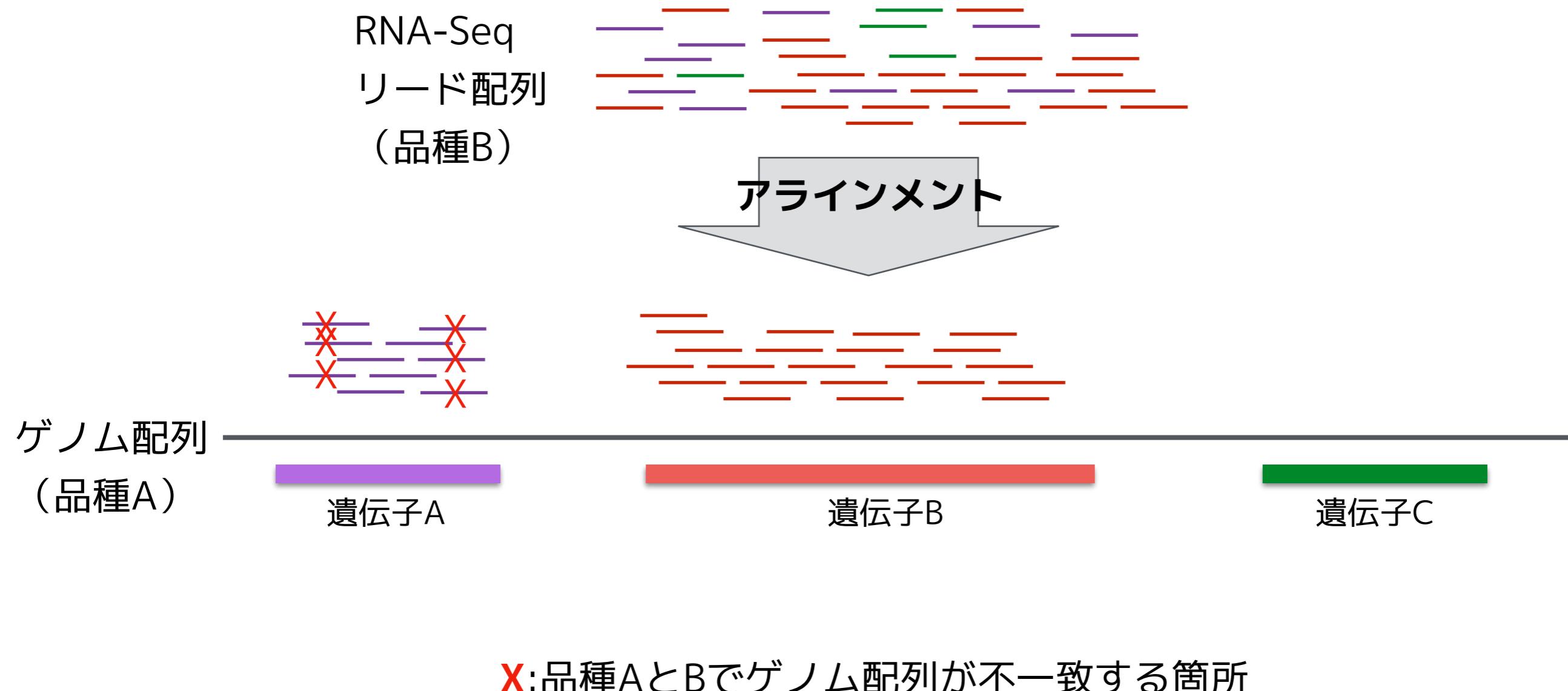
10

11

6

RNA-Seqデータからは発現量以外にも得られる情報はある

リファレンスとは異なる種や品種由来のRNA-Seqリードをゲノム配列にアラインメントし、ゲノムとリード配列の違い（種間や品種間の転写領域上の多型情報）を得る。



9時00分-12時00分

- ▶ リファレンス情報を用いたRNA-Seq解析
- ▶ De novoトランскriプトーム解析

「RNA-Seqによる大量発現データ解析の基礎」



～お昼休み～



13時00分-14時00分

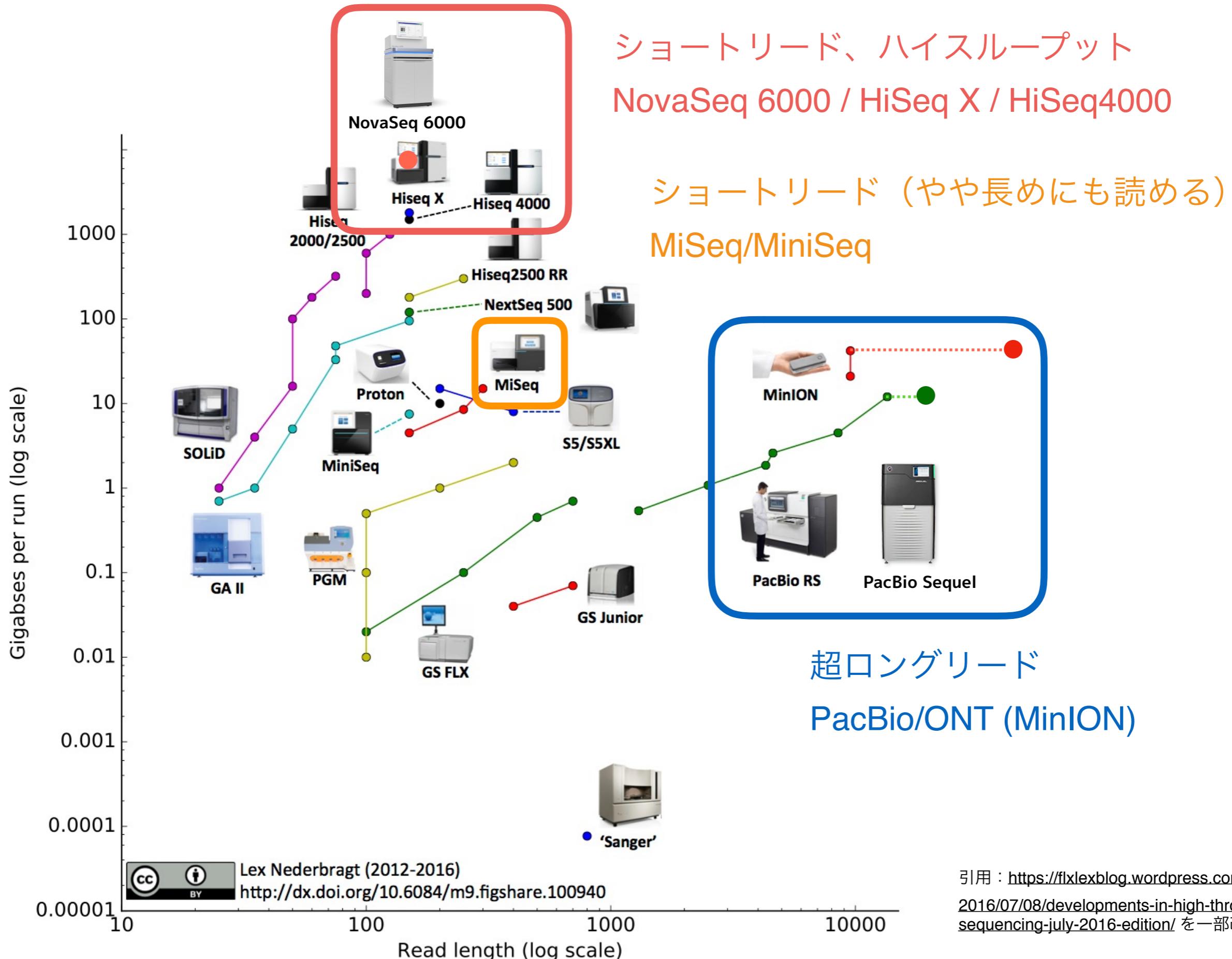
- ▶ RNA-Seqデータを用いた多型検出

「RNA-Seqによる大量発現データ解析の応用」

ライブラリ調製、シーケンシングの前に どのようにRNA-Seq解析をするのか考えよう

「データが出てから考える」
では手遅れ！（ということもある）

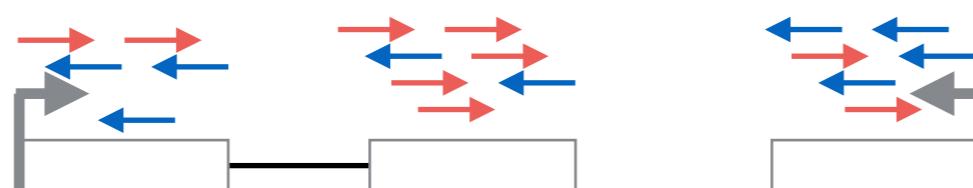
シークンサーの種類はいろいろある



ライブラリ調製やシークエンシング方法にもいろいろある

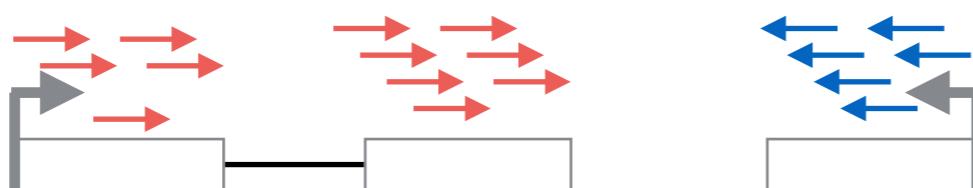


普通の (standard) RNA-Seq



転写の向きとリードの向きは無関係

strand-specific (stranded) RNA-Seq



転写の向きとリードの向きに対応関係があり、アセンブルや発現解析の精度が向上する。

断片化したcDNAの片側だけを読むか両端を読むかの違い。ある一定の距離をもつペアリード情報を用いることで、遺伝子構造予測やアセンブル精度が向上する。

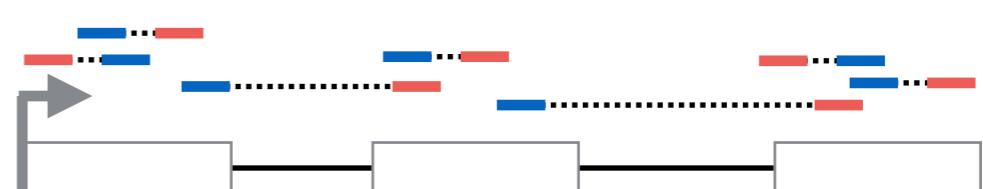
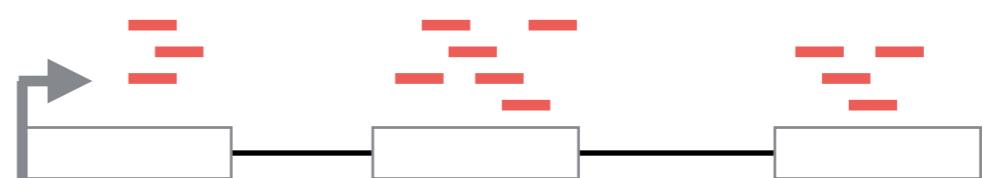
シークエンシング

single read
(SR/SE)



アラインメント

paired-end read
(PE)



目的にあったシーケンシング手法やライブラリ調製法を選びましょう



リファレンス情報を用いた
遺伝子発現解析

- リファレンス情報なし
の遺伝子発現解析
- 多型検出

遺伝子カタログ作成

ゲノムや遺伝子情報は充分。
とにかく発現量が知りたい。

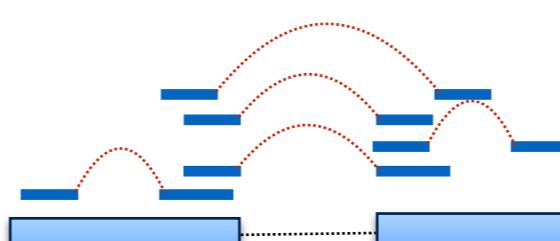
とにかく安価に多くの
リードを読む。

HiSeq, single-read, 50bp



リード数と転写構造の情
報をバランスよく得る。

HiSeq, paired-end, 100/150bp



発現量よりもまずはどん
な遺伝子が発現している
か知りたい。

リード数は少ないが、
とにかく長いリードで
転写産物の構造を得る。

PacBio, ONT



- ▶ リファレンス情報を用いたRNA-Seq解析
- ▶ De novoトランスクリプトーム解析
- ▶ RNA-Seqデータを用いた多型検出

解析ツールの取得、インストール



リファレンス情報を用いたRNA-Seq解析

1. FastQC
2. Trimmomatic
3. HISAT2
4. StringTie
5. Samtools

De novoトランскriプトーム解析

1. FastQC
2. Trimmomatic
3. Trinity
4. Bowtie2
5. Salmon
6. Jellyfish
7. BLAST+
8. edgeR (R/Bioconductor)

RNA-Seqデータを用いた多型検出

1. FastQC
2. Trimmomatic
3. Samtools
4. STAR2
5. Picard
6. GATK

データの可視化

1. IGV (デスクトップアプリ)
2. R/RStudio (デスクトップアプリ)
3. Ballgown (R/Bioconductor)
4. genefilter (R/Bioconductor)
5. tidyverse (R)

解析ツールの取得、インストール



各ツールのウェブサイトから最新版のツールのバイナリ、またはソースファイルをダウンロード、インストールする（資料末尾に各ツールのウェブサイト等の情報あり）。

biocondaやhomebrew（Mac）などのパッケージマネージャ、コンテナ型仮想環境「Docker」を用いても簡単にインストールすることができる。

Samtoolsの場合 (<http://www.htslib.org>)

The screenshot shows the Samtools website's download page. A red arrow points from the text "HTSlib is also distributed as a separate package which can be installed if you are writing your own programs against the HTSlib API." to the "Downloads" menu item in the top navigation bar. Another red arrow points from the "Building and installing" heading to the "Building each desired package from source is very simple:" section.

Current releases

SAMtools and BCFtools are distributed as individual packages. The code uses HTSlib internally, but these source packages contain their own copies of htslib so they can be built independently.

HTSlib is also distributed as a separate package which can be installed if you are writing your own programs against the HTSlib API. HTSlib also provides the `bgzip`, `htsfile`, and `tabix` utilities, so you may also want to build and install HTSlib to get these utilities, or see the additional instructions in [INSTALL](#) to install them from a samtools or bcftools source package.

Download current source releases: [samtools-1.9](#) [bcftools-1.9](#) [htslib-1.9](#)

See also release notes for [samtools](#), [bcftools](#), and [htslib](#).

New releases are announced on the [samtools mailing lists](#) and by [@htslib](#) on Twitter. Previous releases available from the [samtools GitHub organisation](#) (see [samtools](#), [bcftools](#), or [htslib](#) releases) or from the [Sourceforge project](#).

Building and installing

Building each desired package from source is very simple:

```
cd samtools-1.x    # and similarly for bcftools and htslib
./configure --prefix=/where/to/install
make
make install
```

See [INSTALL](#) in each of the source directories for further details.

The executable programs will be installed to a `bin` subdirectory under your specified prefix, so you may wish to add this directory to your `$PATH`:

```
export PATH=/where/to/install/bin:$PATH    # for sh or bash users
```

```
setenv PATH /where/to/install/bin:$PATH    # for csh users
```

演習用データの準備

以下の要領で演習用データ（/work/NGSworkshop/RNA-Seq.tar.gz）を各自のホームディレクトリにコピー、解凍、展開して中身を確認する。

```
$ cd ..... ホームディレクトリへ移動
$ pwd ..... カレントディレクトリを表示
/home/guest01
$ cp /work/NGSworkshop/RNA-Seq.tar.gz ./ ..... 解析用データをカレントディレクトリにコピー
$ ls -lh
合計 169M
-rw-rw-r-- 1 guest01 guest01 169M 9月 17 11:11 RNA-Seq.tar.gz
$ tar xfz RNA-Seq.tar.gz ..... 解析用データを解凍、展開
$ ls -lh
合計 169M
drwxrwxr-x 6 guest01 guest01 109 9月 17 11:02 RNA-Seq
-rw-rw-r-- 1 guest01 guest01 169M 9月 17 11:11 RNA-Seq.tar.gz
$ cd RNA-Seq
$ ls ..... 解析用ディレクトリの中身を確認
data denovo install_tools.sh useref varcall
```

RNA-Seqディレクトリ中の各ファイル/ディレクトリの説明

- data : 解析に用いるデータ（ゲノム配列、遺伝子アノテーション、RNA-Seqリードなど）
- denovo : De novoトランск립トーム解析用のディレクトリ
- useref : リファレンス情報を用いたRNA-Seq解析用のディレクトリ
- varcall : RNA-Seqデータを用いた多型検出解析用のディレクトリ
- install_tools.sh : 解析ツールをインストールするシェルスクリプト

解析ツールのダウンロードとインストール



全解析ツールをまとめてインストールするためのシェルスクリプト

```
$ less install_tools.sh
```

- less、more、catなどのコマンドをつかってスクリプトの中身を見ることができる。
- lessやmoreを終了するためには「q」キーを打つ。

- install_tools.shの先頭部分 -

```
mkdir tool
cd tool
ToolDir=`pwd`  
  
echo "START:" `date`  
### Tools for "useref" analysis  
echo "installing FastQC..."  
wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.7.zip  
unzip fastqc_v0.11.7.zip  
chmod +x FastQC/fastqc  
rm fastqc_v0.11.7.zip  
echo "installing Trimmomatic..."  
wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.38.zip  
unzip Trimmomatic-0.38.zip  
rm Trimmomatic-0.38.zip
```

ツール類を置くためのディレクトリの作成と移動
ツールディレクトリのパスの取得
ツールインストールの開始時間を出力
#で始まるのはコメント行
FastQCのダウンロード
解凍・展開
fastqcプログラムに実行権限を与える
不要なダウンロードファイルは削除
Trimmomaticのダウンロード
解凍・展開
不要なダウンロードファイルは削除

実行ファイル（バイナリファイル）やJavaプログラムのインストールは、
基本的にダウンロード（wget や curl）して、解凍・展開（unzip や tar）するだけ。

解析ツールのダウンロードとインストール

- install_tools.shのつづき1 -

```
echo "installing HISAT2..."                                HISAT2のダウンロード
wget ftp://ftp.ccb.jhu.edu/pub/infphilo/hisat2/downloads/hisat2-2.1.0-Linux_x86_64.zip
unzip hisat2-2.1.0-Linux_x86_64.zip ..... 解凍・展開
rm hisat2-2.1.0-Linux_x86_64.zip ..... 不要なダウンロードファイルは削除
echo "installing StringTie..."                            StringTieのダウンロード
wget http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.4d.Linux_x86_64.tar.gz
tar xfz stringtie-1.3.4d.Linux_x86_64.tar.gz ..... 解凍・展開
rm stringtie-1.3.4d.Linux_x86_64.tar.gz ..... 不要なダウンロードファイルは削除
echo "installing Samtools..."
wget https://github.com/samtools/samtools/releases/download/1.9/samtools-1.9.tar.bz2
tar xfb samtools-1.9.tar.bz2 ..... 解凍・展開                               Samtoolsのダウンロード
rm samtools-1.9.tar.bz2 ..... 不要なダウンロードファイルは削除
cd samtools-1.9 ..... Samtoolsディレクトリに移動
make ..... コンパイル
make prefix=`pwd` install ..... prefix（インストール先）を指定してインストール
cd .. ..... toolディレクトリに戻る
```

- *.tar.gz や *.tar.bz2 の解凍・展開は、tarコマンドにオプションをつけて実行する
(*.tar.gzの場合は「tar xfz」、*.tar.bz2の場合は「tar xfb」)。
- Samtoolsについてはソースファイルをダウンロードし、コンパイル (make) 、インストール (make install) を行なう。

解析ツールのダウンロードとインストール

- install_tools.shのつづき2 -

```
### Tools for "denovo" analysis
echo "installing CMake..."
wget https://cmake.org/files/v3.12/cmake-3.12.1-Linux-x86_64.tar.gz ..... CMakeのダウンロード
tar xfz cmake-3.12.1-Linux-x86_64.tar.gz ..... 解凍・展開
rm cmake-3.12.1-Linux-x86_64.tar.gz ..... 不要なダウンロードファイルは削除
PATH=$ToolDir/cmake-3.12.1-Linux-x86_64/bin:$PATH ..... CMakeの実行ファイルのあるディ
export PATH ..... レクトリをパスに追加、設定
echo "installing Trinity..."
wget https://github.com/trinityrnaseq/trinityrnaseq/archive/Trinity-v2.8.3.tar.gz
tar xfz Trinity-v2.8.3.tar.gz ..... 解凍・展開 ..... Trinityのダウンロード
rm Trinity-v2.8.3.tar.gz ..... 不要なダウンロードファイルは削除
cd trinityrnaseq-Trinity-v2.8.3 ..... Trinityのディレクトリに移動
make ..... コンパイル
cd ..
```

- 最新版のTrinityのコンパイルにはCMakeが必要なので、先にCMakeをインストールし、パスの設定をする。CMakeはソフトウェアのビルドの自動化ツールである。

解析ツールのダウンロードとインストール



- install_tools.shのつづき3 -

```
echo "installing Bowtie2..."  
wget https://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.3.4.2/bowtie2-2.3.4.2-linux-x86_64.zip  
unzip bowtie2-2.3.4.2-linux-x86_64.zip  
rm bowtie2-2.3.4.2-linux-x86_64.zip  
echo "installing Jellyfish..."  
wget https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz  
tar xfz jellyfish-2.2.10.tar.gz  
rm jellyfish-2.2.10.tar.gz  
cd jellyfish-2.2.10  
.configure --prefix=`pwd`  
make  
make install  
cd ..  
echo "installing Salmon..."  
wget https://github.com/COMBINE-lab/salmon/releases/download/v0.11.3/salmon-0.11.3-linux_x86_64.tar.gz  
tar xfz salmon-0.11.3-linux_x86_64.tar.gz  
rm salmon-0.11.3-linux_x86_64.tar.gz  
echo "installing BLAST+..."  
wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST//ncbi-blast-2.7.1+-x64-linux.tar.gz  
tar xfz ncbi-blast-2.7.1+-x64-linux.tar.gz  
rm ncbi-blast-2.7.1+-x64-linux.tar.gz
```

解析ツールのダウンロードとインストール

- install_tools.shのつづき4 -

```
### Tools for "varcall" analysis
echo "installing STAR..."
wget https://github.com/alexdobin/STAR/archive/2.6.1b.tar.gz
tar xfz 2.6.1b.tar.gz
rm 2.6.1b.tar.gz
echo "installing Picard..."
mkdir picard-2.18.12
cd picard-2.18.12
wget https://github.com/broadinstitute/picard/releases/download/2.18.12/picard.jar
cd ..
echo "installing GATK..."
# wget https://github.com/broadinstitute/gatk/releases/download/4.0.8.1/gatk-4.0.8.1.zip
cp /work/NGSworkshop/tool/gatk-4.0.8.1.zip ./
unzip gatk-4.0.8.1.zip
rm gatk-4.0.8.1.zip
echo "END:" `date`
```

GATKのダウンロードは時間がかかるため、あらかじめ
ダウンロードしておいたファイルをコピーする。
他のサーバにインストールする際には、この行をコメ
ントアウトし、1つ上の行のコメントを外して、wget
でダウンロードする。

解析ツールのダウンロードとインストール



ツールのインストールをするためのシェルスクリプトの実行 (実行時間：約5分)

- 普通にbashコマンドで実行すると、スクリプトの標準出力とエラー出力をターミナル画面に表示する。

```
$ bash ./install_tools.sh
```

- リダイレクト機能を使うと、スクリプトの結果はターミナル画面に表示せず、標準出力と標準エラー出力をそれぞれ別のファイルに書き出す。あとでログを見直すことができて便利。

```
$ bash ./install_tools.sh 2> install_tools.stderr > install_tools.stdout
```

- 一連のコマンドが書かれたシェルスクリプトにより、コマンドを連続して実行することができる。スクリプトとして残しておくと、あとで実行コマンドを確認する際にも役立つ。
- ツールによっては、解析結果の統計情報などを標準出力や標準エラー出力として出力するので、それらもファイルに保存しておくとよい。

インストールされた解析ツールを確認



全てのツールはtoolディレクトリにインストールされる。

```
$ ls tool/  
FastQC  
STAR-2.6.1b  
Trimmomatic-0.38  
bowtie2-2.3.4.2-linux-x86_64  
cmake-3.12.1-Linux-x86_64  
gatk-4.0.8.1  
hisat2-2.1.0  
jellyfish-2.2.10  
ncbi-blast-2.7.1+  
picard-2.18.12  
salmon-0.11.3-linux_x86_64  
samtools-1.9  
stringtie-1.3.4d.Linux_x86_64  
trinityrnaseq-Trinity-v2.8.3
```

例えば、HISAT2は「tool/hisat2-2.1.0」以下にインストールされている。

```
$ ls tool/hisat2-2.1.0  
AUTHORS  
hisat2_extract_snps_haplotypes_VCF.py  
LICENSE  
MANUAL  
MANUAL.markdown  
NEWS  
TUTORIAL  
VERSION  
doc  
example  
extract_exons.py  
extract_splice_sites.py  
hisat2  
hisat2-align-l  
hisat2-align-l-debug  
hisat2-align-s  
hisat2-align-s-debug  
hisat2-build  
hisat2-build-l  
hisat2-build-l-debug  
hisat2-build-s  
hisat2-build-s-debug  
hisat2-inspect  
hisat2-inspect-l  
hisat2-inspect-l-debug  
hisat2-inspect-s  
hisat2-inspect-s-debug  
hisat2_extract_exons.py  
hisat2_extract_snps_haplotypes_UCSC.py  
hisat2_extract_splice_sites.py  
hisat2_simulate_reads.py  
hisatgenotype.py  
hisatgenotype_build_genome.py  
hisatgenotype_extract_reads.py  
hisatgenotype_extract_vars.py  
hisatgenotype_hla_cyp.py  
hisatgenotype_locus.py  
hisatgenotype_modules  
hisatgenotype_scripts  
scripts
```

インストールされた解析ツールを確認



多くのツールは「`--help`」や「`-h`」オプションをつけて実行することで簡単な使い方や指定できるパラメータなどを確認できる。

```
$ ./tool/hisat2-2.1.0/hisat2 -h ..... -hオプションをつけてHISAT2を実行
```

HISAT2 version 2.1.0 by Daehwan Kim (infphilo@gmail.com, wwwccb.jhu.edu/people/infphilo)

Usage:

```
hisat2 [options]* -x <ht2-idx> {-1 <m1> -2 <m2> | -U <r> | --sra-acc <SRA accession number>} [-S <sam>]
```

<ht2-idx> Index filename prefix (minus trailing .X.ht2).

<m1> Files with #1 mates, paired with files in <m2>.

Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).

<m2> Files with #2 mates, paired with files in <m1>.

Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).

<r> Files with unpaired reads.

Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).

⋮ (省略)

`--version` print version information and quit

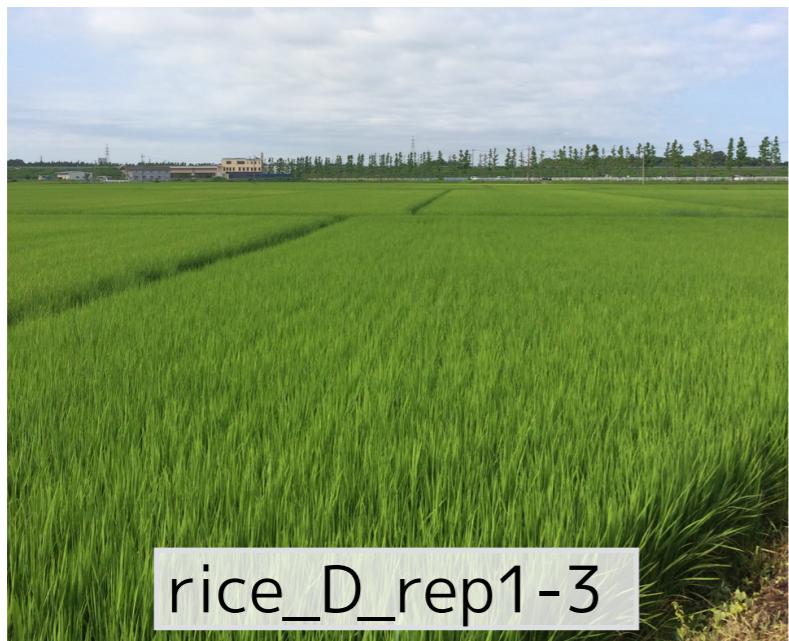
`-h/--help` print this usage message

ここでエラーが出なければ、正しくインストールされている（ことが多い）

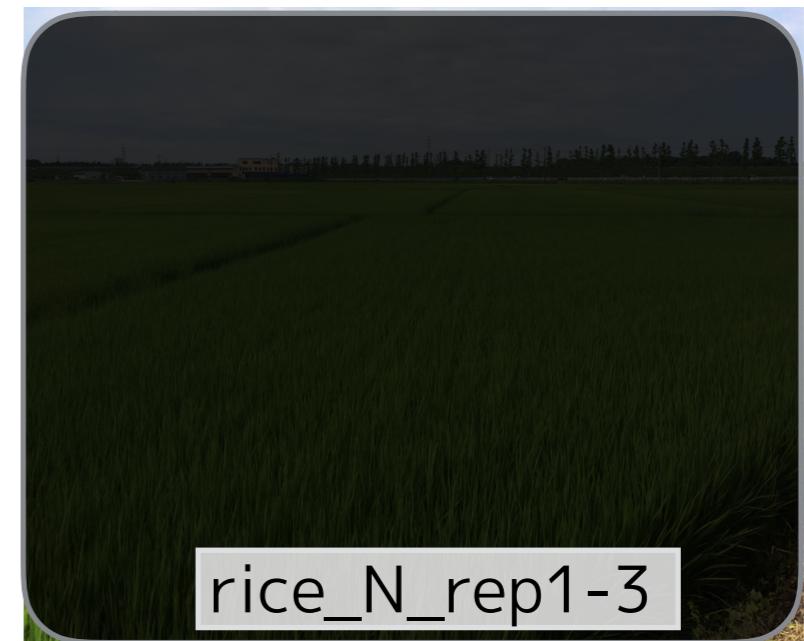
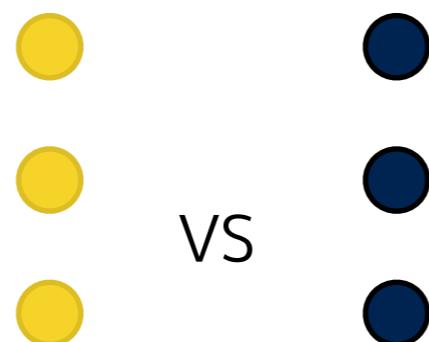
- ▶ リファレンス情報を使ったRNA-Seq解析
- ▶ De novoトランскriプトーム解析
- ▶ RNA-Seqデータを用いた多型検出

目的と使用するデータ

目的：リファレンス情報を用いて、
昼と夜でのイネの葉の遺伝子発現の違いを調べる



12:00 PM



00:00 AM

- 田んぼで栽培する、イネ（コシヒカリ）の葉身のサンプル。
- 3つの異なる生育ステージ（**3反復**）において、昼（12時）と夜（0時）にサンプリング（**2条件**）。
- Illumina社の**Stranded mRNA-Seq法**でライブラリ調製。
- Illumina社のHiSeq2000による、**101bpのPaired-endシーケンシング**。

*未公開データ。全データでは解析に時間がかかるため、2番染色体の特定の領域（2Mbp）にアラインメントされるリードだけを事前に抽出しており、演習ではそれを利用する。

目的と使用するデータ

```
$ cd ..... ホームディレクトリに移動  
$ cd RNA-Seq ..... RNA-Seqディレクトリに移動  
$ ls data ..... dataディレクトリの中身を確認  
annotation.gtf ..... rice_D_rep2_r2.org.fastq.gz rice_N_rep2_r1.org.fastq.gz  
genome.fa ..... rice_D_rep3_r1.org.fastq.gz rice_N_rep2_r2.org.fastq.gz  
rice_D_rep1_r1.org.fastq.gz ..... rice_D_rep3_r2.org.fastq.gz rice_N_rep3_r1.org.fastq.gz  
rice_D_rep1_r2.org.fastq.gz ..... rice_N_rep1_r1.org.fastq.gz rice_N_rep3_r2.org.fastq.gz  
rice_D_rep2_r1.org.fastq.gz ..... rice_N_rep1_r2.org.fastq.gz rice_protein.fa
```

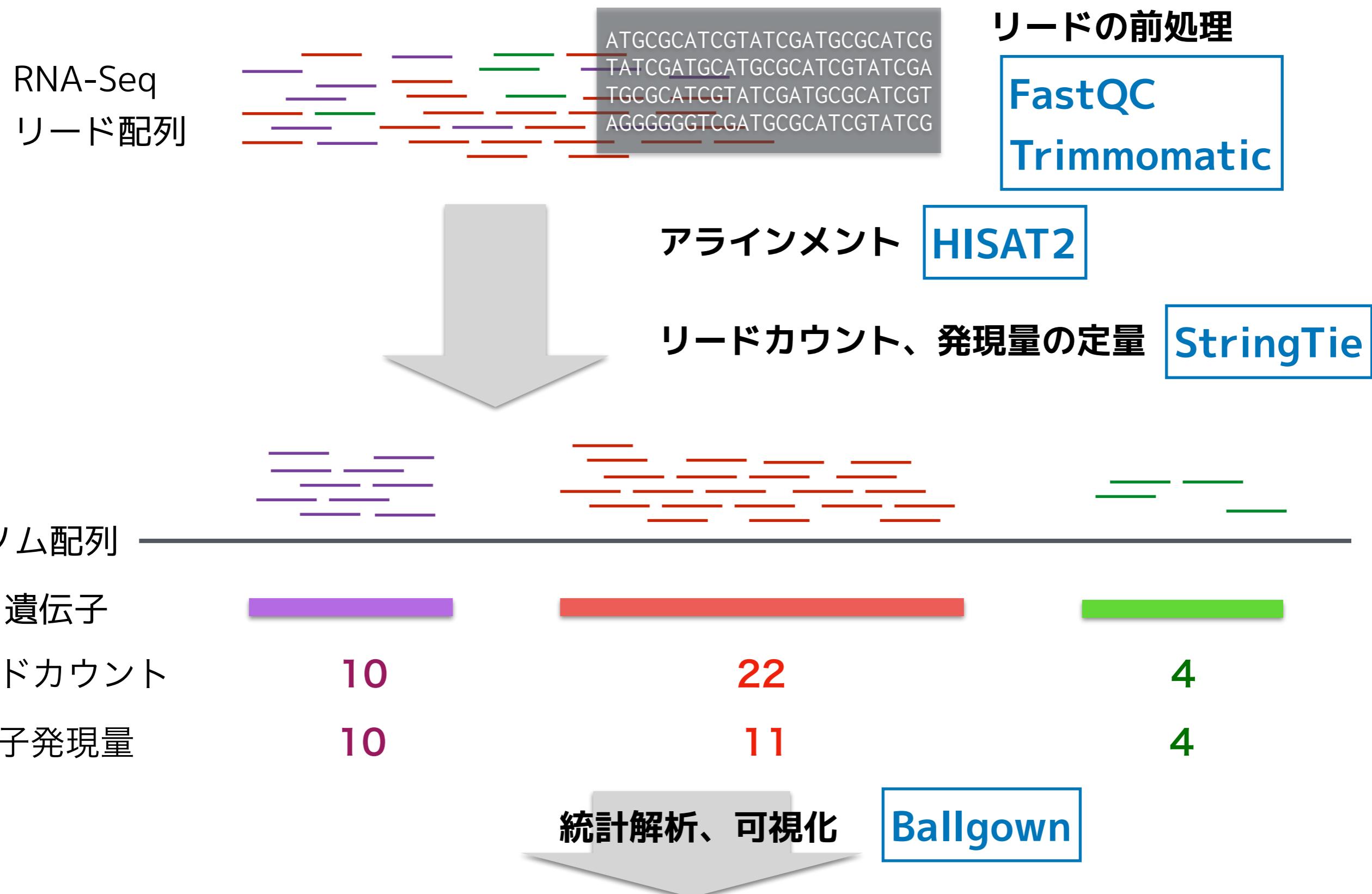
dataディレクトリ中の各ファイルの説明

- ・ リファレンスゲノム配列 (genome.fa)
- ・ 遺伝子アノテーションファイル (annotation.gtf)
- ・ mRNA-Seqリード配列 (*.fastq.gz)
- ・ イネ遺伝子のアミノ酸配列 (rice_protein.fa)



*未公開データ。全データでは解析に時間がかかるため、2番染色体の特定の領域（2Mbp）にアラインメントされるリードだけを事前に抽出しており、演習ではそれを利用する。

リファレンス情報を用いたRNA-Seq解析の流れ



2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

リファレンス情報を用いたRNA-Seq解析のためのツール



HISAT2
graph-based alignment of next generation sequencing reads to a population of genomes

JOHNS HOPKINS UNIVERSITY
CENTER FOR COMPUTATIONAL BIOLOGY
CCB

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes (as well as against a single reference genome). Based on an extension of BWT for graphs [Sirén et al. 2014], we designed and implemented a graph FM index (GFM), an original approach and its first implementation to the best of our knowledge. In addition to using one global GFM index that represents a population of human genomes, HISAT2 also supports multiple local GFM indexes (LGFM) for each transcript.

HISAT2
splice-awareなリードアラインメント

HISAT2 2.0.4 Windows binary available [here](#), thanks to Andre Osorio Falcao
5/24/2016

HISAT2 2.0.4 release 5/18/2016

Version 2.0.4 is a minor release with the following changes.
◦ Improved template length estimation (the 9th column of the SAM format) of RNA-seq

FAQ
News and Updates
New releases and related tools will be announced through the Bowtie mailing list.

<https://ccb.jhu.edu/software/hisat2/index.shtml>

Tuxedo suite tools

(BowtieやTopHat, Cufflinksの後継プログラム)

StringTie
Transcript assembly and quantification for RNA-Seq

JOHNS HOPKINS UNIVERSITY
CENTER FOR COMPUTATIONAL BIOLOGY
CCB

Home Manual Examples CCB » Software » StringTie

▪ Overview
▪ News
▪ Obtaining and installing
▪ Licensing and contact
▪ Publications

StringTie
発現量の定量
遺伝子構造のアセンブル

StringTie is a fast and highly accurate transcript assembler. It uses a novel network flow algorithm as well as an efficient search space reduction strategy to find the best transcript assembly from RNA-seq data. StringTie can align reads to a reference genome, but also alignments longer sequences that have been assembled from those reads. In order to identify differentially expressed genes between experiments, StringTie's output can be processed by specialized software like Ballgown, Cuffdiff or other programs (DESeq2, edgeR, etc.).

a novel network flow representing multiple transcript assemblers, including splice variants for each gene.

<https://ccb.jhu.edu/software/stringtie/>

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

alyssafrazee / ballgown Watch 15 Star 50 Fork 24

Code Issues 25 Pull requests 0 Projects 0 Wiki Pulse Graphs

Bioconductor package "ballgown", developed by alyssafrazee. It is in R.

http://biorxiv.org/content/biorxiv/early/2014/07/10/73333

976 commits

Branch: master New pull request

JMF47 committed on GitHub Merge pull request #85 from alyssafrazee/devel ...

R Fix dimension issue for plotting single-transcript genes. 5 days ago

data Commit made by the Bioconductor Git-SVN bridge. 2 years ago

figure Commit made by the Bioconductor Git-SVN bridge. 2 years ago

Find file Clone or download

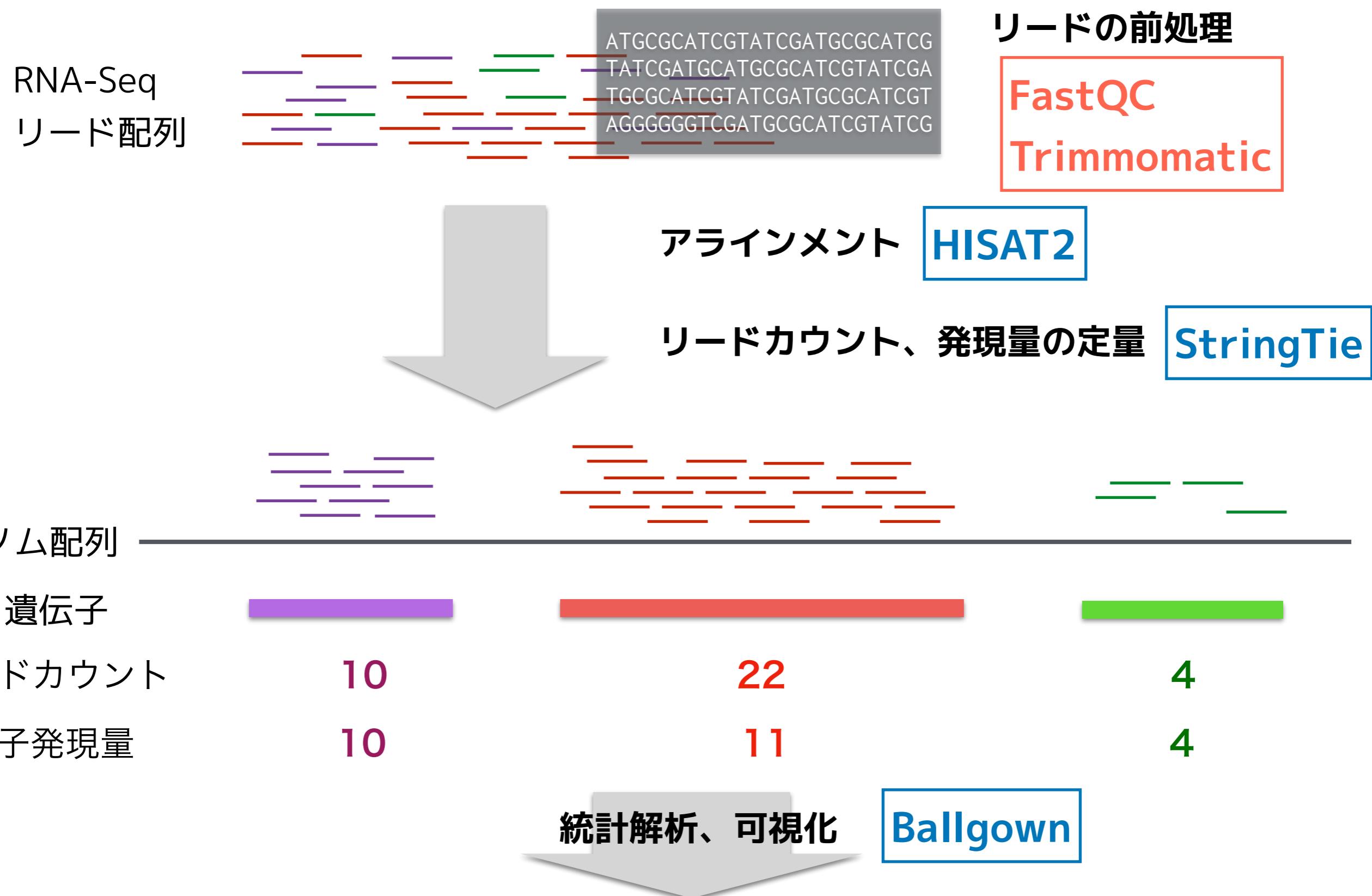
6 contributors

Latest commit a06ad84 5 days ago

Ballgown
発現解析と可視化

<http://bioconductor.org/packages/release/bioc/html/ballgown.html>

リファレンス情報を用いたRNA-Seq解析の流れ



2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

Step 1. 前処理 - 配列データのQC

悪いデータからは良い結果は得られない。

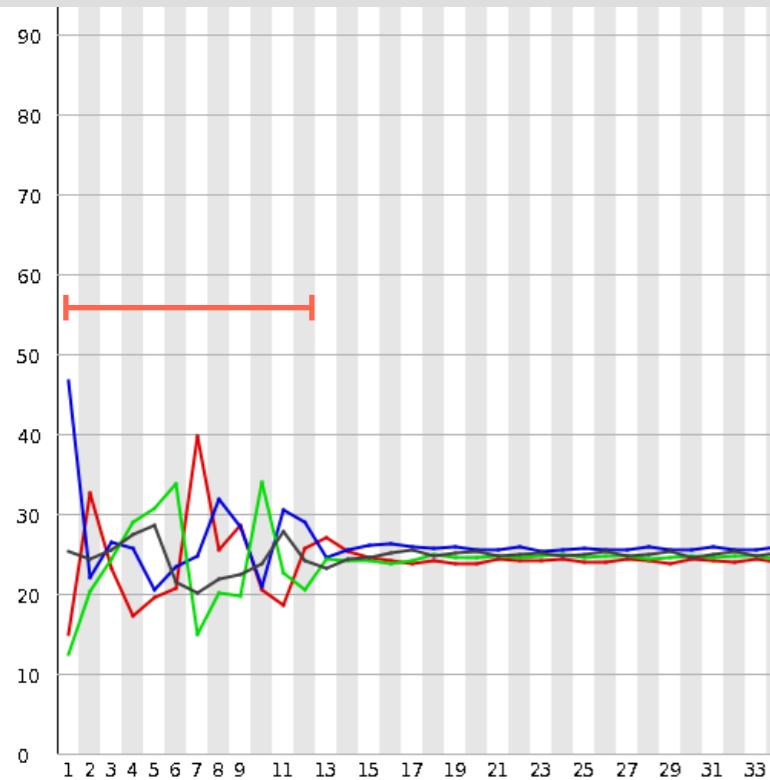
FastQCなどのQCツールを使い、リード数や解読塩基量、クオリティ、塩基出現頻度の偏り、配列長などを事前によくチェックする！

RNA-Seqデータの特徴

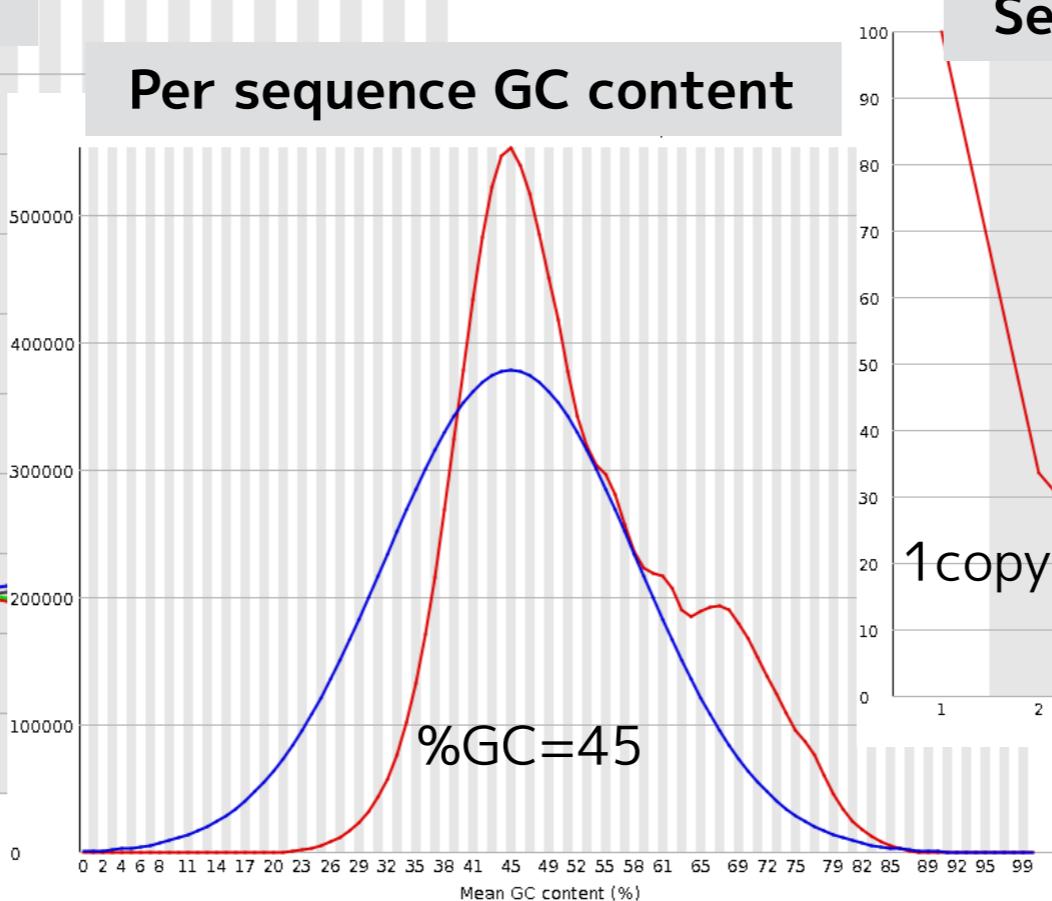
- ・先頭部分の塩基頻度のバイアス
- ・GCバイアス
- ・高発現遺伝子によると思われる
Duplication levelの高さ

これらの特徴はゲノムシーケンスデータとはやや異なるので注意。

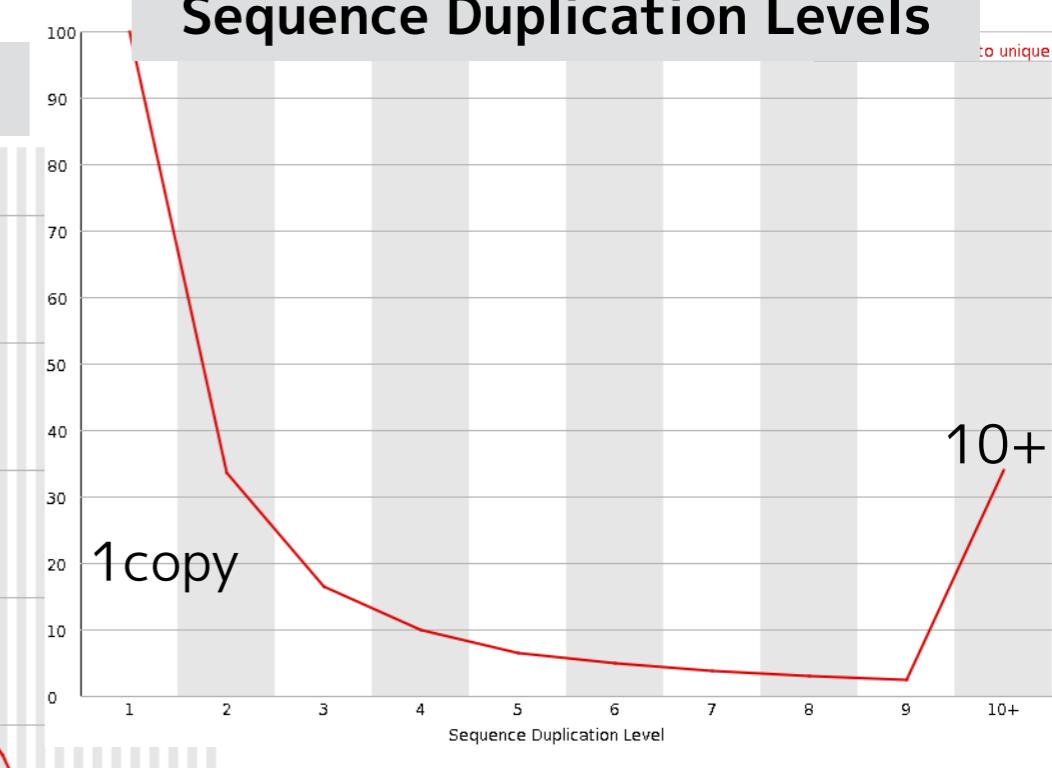
Per base sequence content



Per sequence GC content



Sequence Duplication Levels



Step 1. 前処理 - 配列データのクリーニング



1. 低クオリティ塩基やアダプター配列の除去

TrimmomaticやFASTX-Toolkitなどのツールを用いる。

(例) Trimmomaticで「ILLUMINACLIP:[adapter file]:2:30:10 LEADING:15 TRAILING:15 SLIDINGWINDOW:10:15 MINLEN:50」などと指定して実行する。

2. 実験で除ききれなかったrRNA由来のリードの除去 (本演習ではスキップ)

既知のrRNA配列ライブラリに対してBowtie2等でアラインメントし、そのアンマップリード（つまり、rRNA由来ではないリード）を以降の解析で用いる。

(例) bowtie2 -x [rRNA bowti2 index file] --un-conc-gz [unmap_read.fastq]
-1 [read1.fastq] -2 [read2.fastq] -S [mapped_read.sam]

これらの処理は必須ではないが、塩基やリードの除去率やrRNAへのマップ率は、ライブラリ調製やシーケンシングに何か問題がないかを確認するための指標になる。

Step 1. 前処理 - 配列データのQCとクリーニング

リファレンス情報を用いたRNA-Seq解析用ディレクトリ「useref」に移動

```
$ cd ~/RNA-Seq/useref
```

解析用ディレクトリ内のファイルを確認

```
$ ls
step1_preprocessing.sh  step3_HISAT2.sh
step2_HISAT2-build.sh   step4_StringTie-abundance_estimation.sh
```

シェルスクリプトが4つ用意されており、順番にシェルスクリプトを実行することで解析が進められるようになっている。

```
$ bash ./step1_preprocessing.sh
$ bash ./step2_HISAT2-build.sh
$ bash ./step3_HISAT2.sh
$ bash ./step4_StringTie-abundance_estimation.sh
```

演習では各ステップを説明しながら進めていくのでまだ実行しないでください！

Step 1. 前処理 - 配列データのQCとクリーニング



リードの前処理をするためのシェルスクリプト

```
$ less ./step1_preprocessing.sh
```

- step1_preprocessing.shの前半 -

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
FastQC_bin=$ToolDir/FastQC  
Trimmomatic_bin=$ToolDir/Trimmomatic-0.38  
  
export PATH=$FastQC_bin:$PATH ..... exportコマンドで各ツールのディレクトリをPATHに追加。  
各ツールのコマンドが置かれている  
ディレクトリを変数に格納  
  
# ## Step1. Preprocessing by FastQC and Trimmomatic  
FASTQC_OUTDIR_BEFORE=FastQC_before_preprocess  
FASTQC_OUTDIR_AFTER=FastQC_after_preprocess  
mkdir $FASTQC_OUTDIR_BEFORE $FASTQC_OUTDIR_AFTER ..... FastQCの出力ディレクトリを作成
```

何度も出てくるパス等はスペルミスを避けたり、スクリプトの可読性を上げるために、変数に格納したり、exportコマンドでPATHに設定するとよい。

Step 1. 前処理 - 配列データのQCとクリーニング

- step1_preprocessing.shのつづき -

```

for DATASET in rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3
do ..... 繰り返し開始
    # FastQC before Trimmomatic ..... 前処理前のFastQCの実行 (r*でリード1と2を合せて指定)
    fastqc --threads 1 --nogroup --outdir $FASTQC_OUTDIR_BEFORE --format fastq
    $DataDir/${DATASET}_r*.org.fastq.gz
    # Trimmomatic ..... Trimmomaticの実行
    java -Xmx4G -Xms2G -jar $Trimmomatic_bin/trimmomatic-0.38.jar PE \
        -threads 1 -phred33 -trimlog Trimmomatic_${DATASET}.log \
        $DataDir/${DATASET}_r1.org.fastq.gz $DataDir/${DATASET}_r2.org.fastq.gz \
        ${DATASET}_r1.pe.fastq.gz ${DATASET}_r1.unpe.fastq.gz \
        ${DATASET}_r2.pe.fastq.gz ${DATASET}_r2.unpe.fastq.gz \
        ILLUMINACLIP:$Trimmomatic_bin/adapters/TruSeq3-PE-2.fa:2:30:10 \
        LEADING:15 TRAILING:15 SLIDINGWINDOW:10:15 MINLEN:50
    # FastQC after Trimmomatic ..... 前処理後のFastQCの実行
    fastqc --threads 1 --nogroup --outdir $FASTQC_OUTDIR_AFTER --format fastq ${DATASET}
    _r*.pe.fastq.gz
done ..... 繰り返し終了

```

for関数による繰り返し。変数DATASETにin以降のサンプル名が順番に格納されて実行される。

\（バックスラッシュ）は実行時には無視され、ひと続きのコマンドとして実行される。可読性を上げるために使用。

サンプル数が6つあるため、「前処理前後のFastQCとTrimmomaticによる前処理」を各サンプルごとにfor文を使って繰り返し実行している。

Step 1. 前処理 - 配列データのQCとクリーニング



シェルスクリプトの実行 (実行時間：約3分)

```
$ bash ./step1_preprocessing.sh
```

出力ファイルの確認

```
$ ls
FastQC_after_preprocess          rice_D_rep2_r1.pe.fastq.gz    rice_N_rep2_r1.pe.fastq.gz
FastQC_before_preprocess         rice_D_rep2_r1.unpe.fastq.gz   rice_N_rep2_r1.unpe.fastq.gz
Trimmomatic_rice_D_rep1.log      rice_D_rep2_r2.pe.fastq.gz    rice_N_rep2_r2.pe.fastq.gz
Trimmomatic_rice_D_rep2.log      rice_D_rep2_r2.unpe.fastq.gz   rice_N_rep2_r2.unpe.fastq.gz
Trimmomatic_rice_D_rep3.log      rice_D_rep3_r1.pe.fastq.gz    rice_N_rep3_r1.pe.fastq.gz
Trimmomatic_rice_N_rep1.log      rice_D_rep3_r1.unpe.fastq.gz   rice_N_rep3_r1.unpe.fastq.gz
Trimmomatic_rice_N_rep2.log      rice_D_rep3_r2.pe.fastq.gz    rice_N_rep3_r2.pe.fastq.gz
Trimmomatic_rice_N_rep3.log      rice_D_rep3_r2.unpe.fastq.gz   rice_N_rep3_r2.unpe.fastq.gz
rice_D_rep1_r1.pe.fastq.gz       rice_N_rep1_r1.pe.fastq.gz    step1_preprocessing.sh
rice_D_rep1_r1.unpe.fastq.gz     rice_N_rep1_r1.unpe.fastq.gz   step2_HISAT2-build.sh
rice_D_rep1_r2.pe.fastq.gz       rice_N_rep1_r2.pe.fastq.gz    step3_HISAT2.sh
rice_D_rep1_r2.unpe.fastq.gz     rice_N_rep1_r2.unpe.fastq.gz   step4_StringTie-abundance_estimation.sh
```

- FastQC_after_preprocess: 前処理後のリードのFastQCの結果
- FastQC_before_preprocess: 前処理前のリードのFastQCの結果
- Trimmomatic_rice_*.log: Trimmomaticのログ (リードごとのトリミング情報)
- rice_*.pe.fastq.gz: 前処理後のリード配列ファイル (ペアを維持)
- rice_*.unpe.fastq.gz: 前処理後のリード配列ファイル (ペアの相方が捨てられたもの)

Trimmomaticによる前処理結果の統計情報は標準エラーとして出力される。標準エラー出力をファイルに書き出していくれば、以下のように確認できる。

```
$ less step1_preprocessing.stderr
```

...

```
Input Read Pairs: 18407642 Both Surviving: 16992068 (92.31%) Forward Only  
Surviving: 1173984 (6.38%) Reverse Only Surviving: 159615 (0.87%) Dropped:  
81975 (0.45%)
```

...

- ・ 前処理の結果、16,992,068 ペア (92.31%) が残り、このあとの解析に利用される。
- ・ ペアの片方しか残っていない (* Only Surviving) 、もしくはペアの両方が失われてしまった (Dropped) 割合があまりに多い場合は、トリミングのパラメータを緩めるなど再検討する。あまりに酷い場合は再シーケンシングも検討。

*この例は演習用データではなく、その元となった全ゲノムデータによる結果。
演習用データは前処理後にゲノムにアラインメントできたリードのみ抽出しているため、Both Survivingが100%になっているはず。

前処理で見つかる問題と対応策

1. 低クオリティ塩基が多い

シークエンシング時の問題であることが多いので読み直す。

2. アダプター配列の混入率が高い

RNAの濃度が薄い、解読リード長に対してインサート長が短いなど。ライブラリ調製からやり直す。

3. rRNAの混入率が高い

ポリA精製やrRNAの除去がうまくいっていない。ライブラリ調製からやり直す。

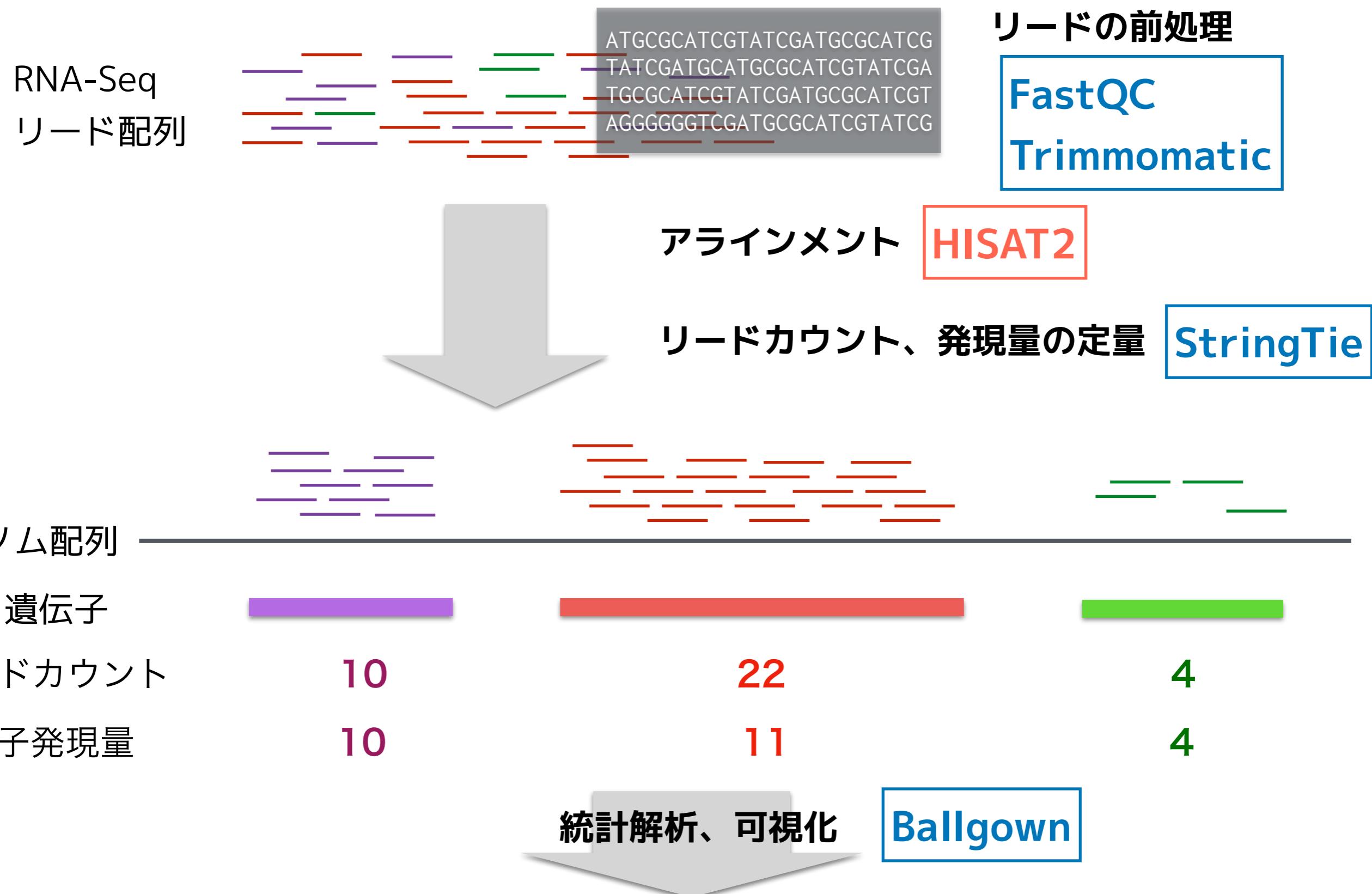
4. ゲノム、転写産物配列へのマップ率が低い

コンタミ等が疑われる。サンプリングからやり直した方が良い場合もある。

5. 有効なリード数が少ない

シークエンシングを追加する。生物種や対象組織、目的等によっても必要なリード数は異なるが、個人的にはイネの葉であれば1反復当たり1-2千万（10-20 million）リードがあれば十分だと思う。

リファレンス情報を用いたRNA-Seq解析の流れ



2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

RNA-Seqリードとゲノム配列のアラインメントの難しさ



ゲノムリシーケンスリード



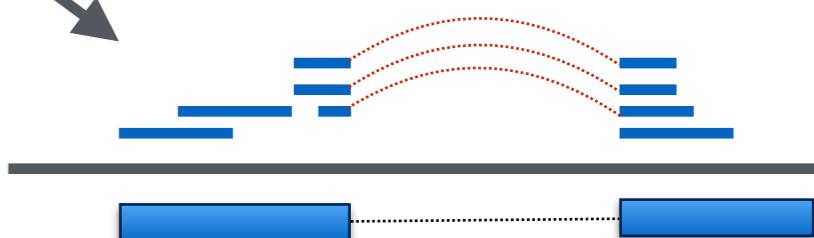
ゲノム配列

RNA-Seqリード



転写産物配列

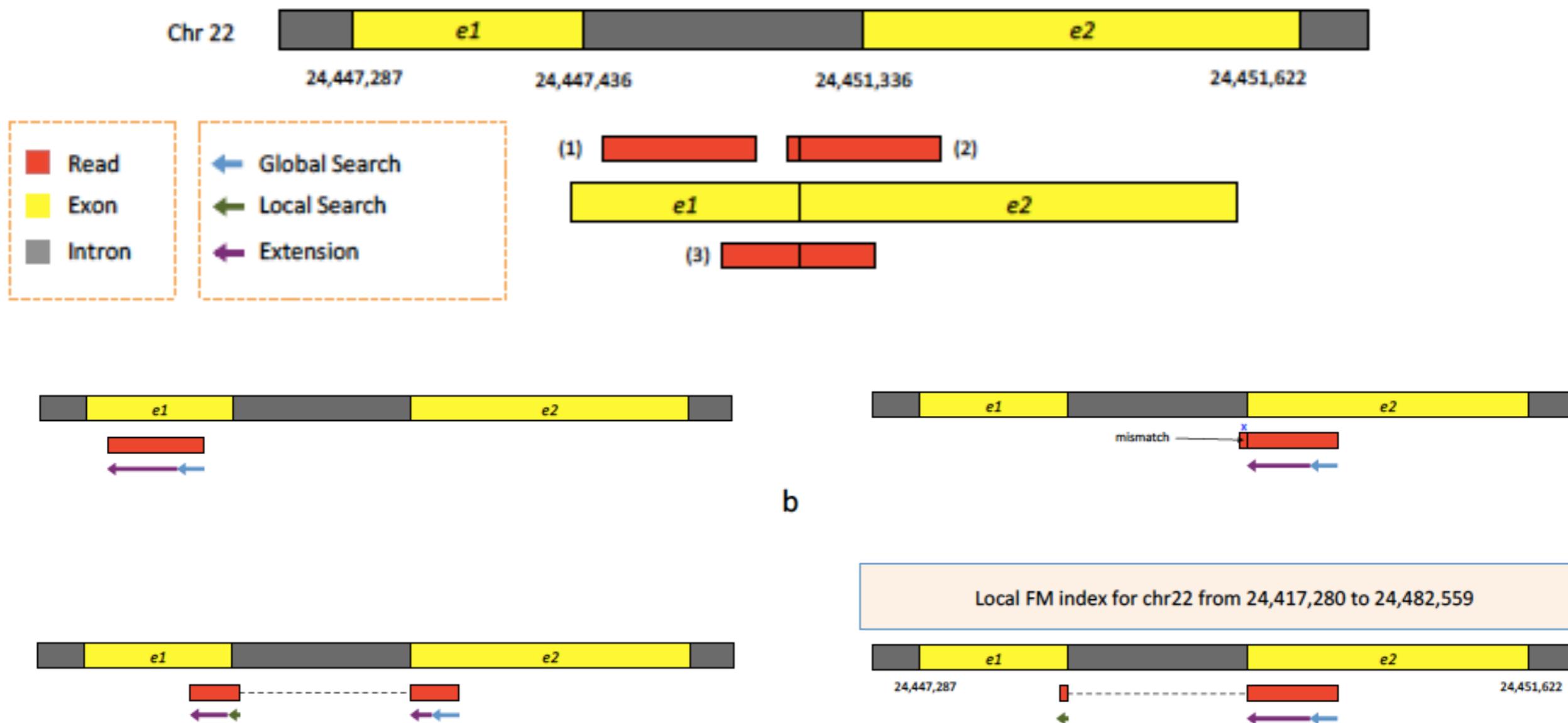
長いギャップ（イントロン）を含むアラインメントを考慮する必要がある



ゲノム配列

- ・ ゲノムリシーケンス解析でよく使われるBWAやBowtie2、NovoalignなどはRNA-Seqデータ解析には向かない。
- ・ RNA-Seqリードアラインメントに特化した様々なツール（HISAT2、TopHat2、STAR2、GSNAPなど）が開発されている。

HISAT2によるsplice-awareアラインメント



modified Supplementary Figure 8 from Kim, D., Langmead, B., & Salzberg, S. L. (2015) Nature Methods, 12(4)

Global SearchとLocal Searchのための2種類のインデックス (hierarchical indexing) を作成しておき、GlobalとLocal Search、Extensionを切り替えながら最適なアラインメントを探索する。

Step 2. HISAT2のためのインデックスの作成

HISAT2のためのインデックスを作成するシェルスクリプト

```
$ less ./step2_HISAT2-build.sh
```

- step2_HISAT2-build.sh -

```
DataDir=$HOME/RNA-Seq/data
ToolDir=$HOME/RNA-Seq/tool
HISAT2_bin=$ToolDir/hisat2-2.1.0
Samtools_bin=$ToolDir/samtools-1.9
export PATH=$HISAT2_bin:$Samtools_bin:$PATH
### Step2. Build index of the reference genome sequence by hisat2-build and samtools
# make splice site and exon position info
python $HISAT2_bin/hisat2_extract_splice_sites.py $DataDir/annotation.gtf > ss.tab
python $HISAT2_bin/hisat2_extract_exons.py $DataDir/annotation.gtf > exon.tab
# build index for HISAT2
hisat2-build --ss ss.tab --exon exon.tab $DataDir/genome.fa genome
# build index for IGV etc.
samtools faidx $DataDir/genome.fa
```

各ツールのコマンドが置かれている
ディレクトリへのパスを変数に格納

exportコマンドで各ツールの
ディレクトリをPATHに追加。

HISAT2のインデックス作成時に指定する
ExonやSplice site位置情報の作成

HISAT2のための
インデックスの作成

IGVによる可視化時に必要なゲノム配列のインデックス作成

HISAT2用のhierarchical index作成時に、既知のスプライスサイト(ss.tab)やExon (exon.tab) の位置情報を指定している。これによりアラインメントの精度と効率が向上する。

Step 3. HISAT2のためのインデックスの作成



シェルスクリプトの実行 (実行時間：約1分)

```
$ bash ./step2_HISAT2-build.sh
```

作成されたインデックスファイルの確認

```
$ ls genome.*  
genome.1.ht2  genome.3.ht2  genome.5.ht2  genome.7.ht2  
genome.2.ht2  genome.4.ht2  genome.6.ht2  genome.8.ht2  
$ ls ../data/genome.fa*  
../data/genome.fa  ../data/genome.fa.fai
```

- genome.*.ht2ファイルはHISAT2のためのインデックス
- genome.fa.faiは、IGVなどでの可視化に必要なインデックス

遺伝子アノテーションファイル (GTF2/GFF3)



ゲノム上のどこにどんな構造の遺伝子が存在するのかを記載した情報。イネであればRAP-DBから代表転写産物のアノテーション情報 (GFF/GTF) がダウンロードできる。

<http://rapdb.dna.affrc.go.jp/download/irgsp1.html>

The screenshot shows the RAP-DB website interface. At the top, there is a navigation bar with icons for Home, News, About, Browser, Tools, Download (which is highlighted in red), Documents, Publications, and Links. Below the navigation bar, there is a breadcrumb trail "Home > Download" and a search bar with "Keywords" and "Search" buttons. The main content area is titled "Annotation data on Os-Nipponbare-Reference-IRGSP-1.0". Under this title, there are two sections: "Genome sequences" and "Gene set (genes supported by FL-cDNAs, ESTs or proteins)". The "Genome sequences" section contains links for genome assemblies (gz file, 116MB, MD5 checksum) and unanchored sequences (gz file, 356KB, MD5 checksum). The "Gene set" section contains links for gene structure and function information in GFF format (gz file, 15MB), gene structure (only exon) information in GTF format (gz file, 2.1MB), gene annotation information in tab-delimited text format (gz file, 2.6MB), and gene sequences (CDS + UTRs + introns) in FASTA format (gz file, 39MB).

Annotation data on Os-Nipponbare-Reference-IRGSP-1.0

Genome sequences

- Genome sequence (Os-Nipponbare-Reference-IRGSP-1.0)
Genome assemblies (12 chromosomes)* [\[DOWNLOAD\]](#) (gz file, 116MB, [MD5 checksum](#))
Unanchored sequences* [\[DOWNLOAD\]](#) (gz file, 356KB, [MD5 checksum](#))
(*) Sequences are masked by [Censor](#) with [MIPS](#) and [MSU](#) repeat data. The masked regions are replaced by lowercase letters.
- Chromosome sequences of the aus rice cultivar 'Kasalath'.
[\[DOWNLOAD\]](#) (gz file, 199MB)

Gene set (genes supported by FL-cDNAs, ESTs or proteins)

- Gene structure and function information in GFF format.
[\[DOWNLOAD\]](#) (gz file, 15MB)
- Gene structure (only exon) information in GTF format.
[\[DOWNLOAD\]](#) (gz file, 2.1MB)
- Gene annotation information in tab-delimited text format.
[\[DOWNLOAD\]](#) (gz file, 2.6MB)
- Gene sequences (CDS + UTRs + introns) in FASTA format.
[\[DOWNLOAD\]](#) (gz file, 39MB)

- 様々なデータベースから遺伝子の位置や機能情報が提供されているが、書式や記載方法は必ずしも統一されていない。
- HISAT2やStringTieで正しく扱えるような形式に整形しておくと解析に便利。RAP-DBのGTFファイルはHISAT2に対応。

色々な生物のアノテーションファイルを提供しているサイト

Illumina社 iGenomes

https://support.illumina.com/sequencing/sequencing_software/igenome.html

EnsemblPlants

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

Phytozome

<http://phytozome.jgi.doe.gov/pz/portal.html>

HISAT2やStringTieで使用可能なGTFファイルの書式



遺伝子の位置やID、アノテーション情報などが記載されたタブ区切りのテキストファイル

```
$ less ../data/annotation.gtf
...
chr02    irgsp1_rep    transcript    30094300    30099072    .    +
gene_id "Os02g0724000"; transcript_id "Os02t0724000-01"; gene_name "DTH2"; note "CONSTANS-like
protein, Heading promotion under long-day condition";
chr02    irgsp1_rep    exon    30094300    30094498    .    +    .
"Os02g0724000"; transcript_id "Os02t0724000-01";
chr02    irgsp1_rep    exon    30096198    30096893    .    +    .
"Os02g0724000"; transcript_id "Os02t0724000-01";
chr02    irgsp1_rep    exon    30097390    30097590    .    +    .
"Os02g0724000"; transcript_id "Os02t0724000-01";
chr02    irgsp1_rep    exon    30097861    30098165    .    +    .
"Os02g0724000"; transcript_id "Os02t0724000-01";
chr02    irgsp1_rep    exon    30098451    30099072    .    +    .
"Os02g0724000"; transcript_id "Os02t0724000-01";
...
...
```

- 上の例では6行で1つの転写産物（5 exons）の構造とアノテーション情報を示す。
- 「gene_id」（遺伝子座ID）や「transcript_id」（転写産物ID）は、遺伝子座や転写産物を対象にした発現解析を行うために必須。
- 「gene_name」（遺伝子名やシンボル）を付けておくと、解析結果に含まれるので便利。

Step 3. HISAT2によるRNA-Seqリードのアラインメント



6サンプル（2条件、3反復）を1つずつ順番にHISAT2でアラインメントするためのシェルスクリプト。

```
$ less step3_HISAT2.sh
```

- step3_HISAT2.sh -

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
HISAT2_bin=$ToolDir/hisat2-2.1.0  
Samtools_bin=$ToolDir/samtools-1.9
```

```
export PATH=$HISAT2_bin:$Samtools_bin:$PATH
```

共通パラメータ

```
### Step3. Align reads to the reference genome by HISAT2
```

```
HISAT2_COMMON_PARAM="--threads 1 --min-intronlen 20 --max-intronlen 10000 --dta --rna-strandness RF -x genome"
```

```
for DATASET in rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3  
do
```

```
    hisat2 $HISAT2_COMMON_PARAM -1 ${DATASET}_r1.pe.fastq.gz -2 ${DATASET}_r2.pe.fastq.gz
```

```
    -S ${DATASET}.sam
```

```
    samtools sort -o ${DATASET}.bam ${DATASET}.sam
```

```
    samtools index ${DATASET}.bam
```

```
    rm ${DATASET}.sam
```

```
done
```

共通パラメータ、リードファイル、
出力ファイルを指定してHISAT2を実行

SAM形式で出力されたアラインメントをソート、BAM形式へ変換したのち、BAMインデックスを作成し、SAMを削除

Step 3. HISAT2によるRNA-Seqリードのアラインメント



シェルスクリプトの実行 (実行時間：約1分)

```
$ bash ./step3_HISAT2.sh
```

作成されたアラインメントファイルとそのインデックスの確認

```
$ ls *.bam*
rice_D_rep1.bam      rice_D_rep3.bam      rice_N_rep2.bam
rice_D_rep1.bam.bai  rice_D_rep3.bam.bai  rice_N_rep2.bam.bai
rice_D_rep2.bam      rice_N_rep1.bam      rice_N_rep3.bam
rice_D_rep2.bam.bai  rice_N_rep1.bam.bai  rice_N_rep3.bam.bai
```

- *.bamファイルはHISAT2によるアラインメントファイル。IGVなどで読み込み可能。
- *.baiファイルはアラインメント情報のインデックス

HISAT2によるRNA-Seqリードのアラインメント結果の確認



HISAT2によるアラインメントの統計情報は標準エラー出力に書き出される。

```
$ less step3_HISAT2.stderr
16992068 reads; of these:
  16992068 (100.00%) were paired; of these:
    491486 (2.89%) aligned concordantly 0 times
    16005810 (94.20%) aligned concordantly exactly 1 time
    494772 (2.91%) aligned concordantly >1 times
    ----
    491486 pairs aligned concordantly 0 times; of these:
      240097 (48.85%) aligned discordantly 1 time
    ----
    251389 pairs aligned 0 times concordantly or discordantly; of these:
      502778 mates make up the pairs; of these:
        344266 (68.47%) aligned 0 times
        140870 (28.02%) aligned exactly 1 time
        17642 (3.51%) aligned >1 times
98.99% overall alignment rate
```

*この例は演習用データではなく、全ゲノムデータによる結果。

- ・ 全16,992,068リードのうち、98.99%がアラインメントされている。
- ・ 温帯ジャポニカのリードを日本晴リファレンスゲノムにアラインメントした場合、特に問題がなければ90%後半のアラインメント率を示すのが一般的。

HISAT2によるRNA-Seqリードのアライメント結果の確認



samtoolsを使い、個々のリードのアライメント情報を確認することができる。

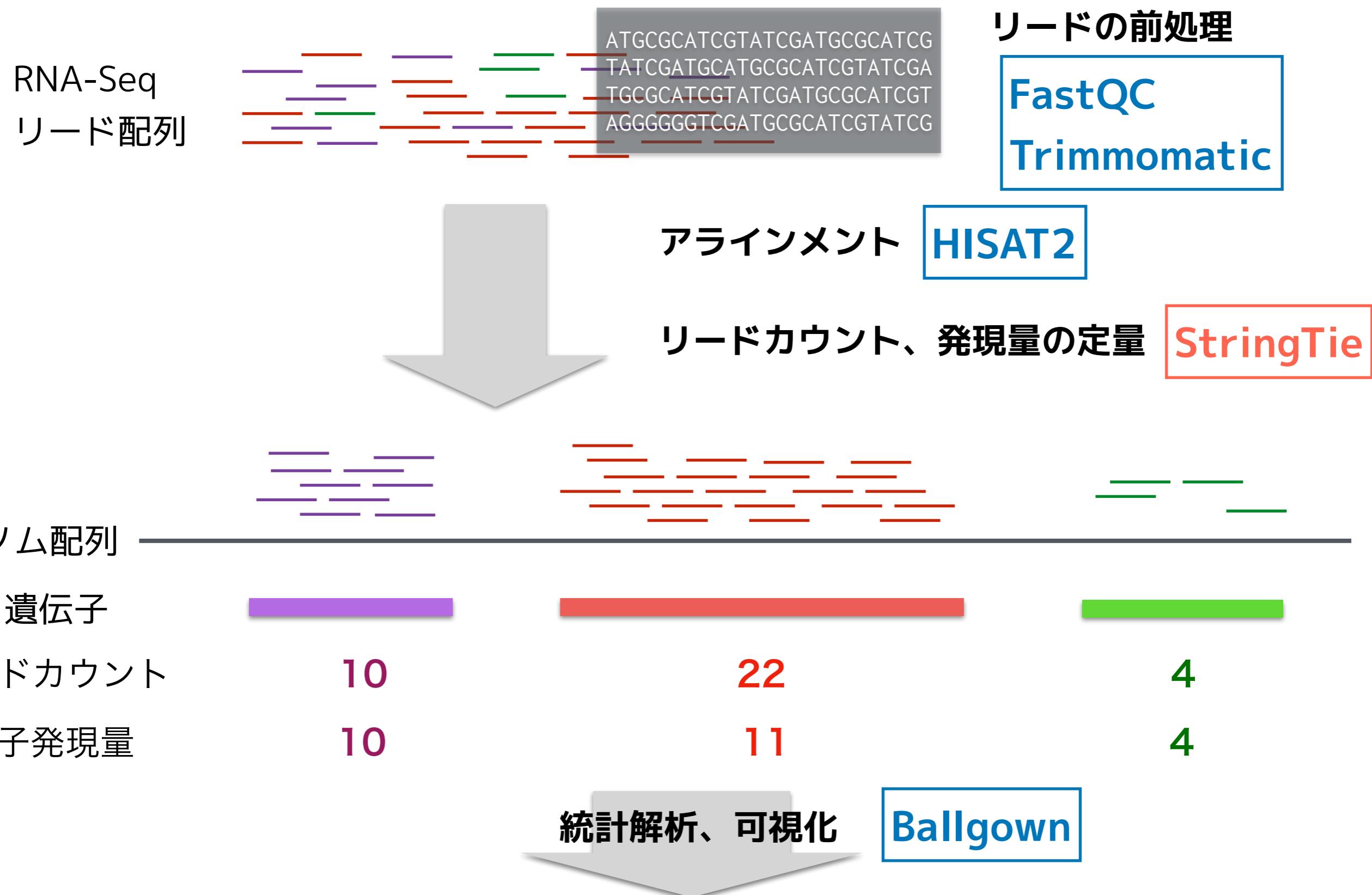
```
$ ./tool/samtools-1.9/bin/samtools view ./rice_D_rep1.bam | less
```

リードID	クエリ位置	ゲノム位置	スコア
MG00HS14:530:C6M7YACXX:8:2208:4438:63887	161	chr02 14245740	60
67M = 31527665 17282026			
TGAACGCTGGCGGCATGCTAACACATGCAAGTCGAACGGGAAGTGGTGTTCAGTGGCGAACGGG			
@@@A7DDDD811@EAFGEB9?BAD38B>FDFDFFEIFA@BEB>E==@@:?=>AADCABABBBB'05B			AS:i:0 XN:i:0
XM:i:0 X0:i:0		DP XS:A:+ NH:i:1	
MG00HS14:530:101M =	75M 112N 21M	chr02 23112840	60
CATCCCTGTCCC	リード	← 1S	
CCACTGGCCGTA	ゲノム配列		ATGTGTTCTCCATCACCGGTCGTGGTACCGTTG
DCBB;@(;@CCG			=CC>@7==@@E=;?=?
XG:i:0 NM:i:0 MD:Z:101	YS:i:0 YT:Z:DP XS:A:+ NH:i:1	AS:i:0 XN:i:0 XM:i:0 X0:i:0	
MG00HS14:530:C6M7YACXX:8:1103:1603:42247	113	chr02 23845634	60
75M112N21M1S = 31055654 7210007			
GTCAGCTTCGCTGCCGCCGTGCTGGGGCTCCTCGCAGCCATCCTCGGGTTCGTCGCGGAAGGCGCCAAGTCCAATTGTTCTCGTGCG			
GTTCGACGGGG C@>CB>><BDB7DDDDBDDC?8(B?BA?B@>;DDDCC@A?<DDDDDDDB@B??			
<FFFHHGEJJIIIIJIIIEFJIHFIIIHGGJIGAFHHHFFFFFCCC AS:i:-14 XN:i:0 XM:i:0			
X0:i:0 XG:i:0 NM:i:0 MD:Z:96 YS:i:-2 YT:Z:DP XS:A:+ NH:i:1			

図解部分：CIGAR文字列「75M 112N 21M」を示す。75Mは正方向のマッチング、112Nはインtron、21Mは逆方向のマッチングである。

- SAMフォーマットについては<http://samtools.github.io/hts-specs/SAMv1.pdf>を参照。
- インtronをまたぐリードアライメントは「75M112N21M1S」のように、インtron部分が“N”で表されている。112bpのインtronを挟むアライメントであるという意味。

リファレンス情報を用いたRNA-Seq解析の流れ



2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

Step 4. StringTieによる遺伝子発現量の定量



StringTieによって、遺伝子アノテーション（GTF）と各サンプルのアラインメント情報（BAM）を元に、各遺伝子ごとにリードカウントや発現量を計算するシェルスクリプト。

```
$ less step4_StringTie-abundance_estimation.sh
```

- step4_StringTie-abundance_estimation.sh -

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
StringTie_bin=$ToolDir/stringtie-1.3.4d.Linux_x86_64  
  
export PATH=$StringTie_bin:$PATH  
  
### Step4. Estimate transcript abundances  
STRINGTIE_COMMON_PARAM="-e -B"  
for DATASET in rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3  
do  
    stringtie $STRINGTIE_COMMON_PARAM -G $DataDir/annotation.gtf -o ballgown/${DATASET}/${DATASET}.gtf ${DATASET}.bam  
done
```

→ 遺伝子アノテーションと出力ファイル、
アラインメントデータを指定して、
StringTieを実行し、発現量を定量。

Step 4. StringTieによる遺伝子発現量の定量

シェルスクリプトの実行 (実行時間：約10秒)

```
$ bash ./step4_StringTie-abundance_estimation.sh
```

サンプルごとの発現量やCoverageデータの確認

```
$ ls ballgown/rice_D_rep1  
e2t.ctab    e_data.ctab   i2t.ctab   i_data.ctab   rice_D_rep1.gtf   t_data.ctab
```

各サンプルごとに指定したディレクトリに結果が出力されている。

- rice_*.gtf : 遺伝子構造と遺伝子発現量
- e2t.ctab : exon id と transcript id の対応表
- e_data.ctab : exonごとの支持するリード数
- i2t.ctab : intron id と transcript id の対応表
- i_data.ctab : intronごとの支持するリード数
- t_data.ctab : transcriptごとの支持するリード数や発現量情報

StringTieの出力結果の確認

transcriptごとの支持するリード数や発現量情報 (t_data.ctab) の確認

```
$ less ballgown/rice_D_rep1/t_data.ctab
```

t_id	chr	strand	start	end	t_name	num_exons	length	gene_id
gene_name		cov	FPKM					
1	chr02	-	29998326		30002783			Os02t0722500-01 9 1436
Os02g0722500		-	5.784122		228.790451			
2	chr02	+	30004088		30004574			Os02t0722600-01 1 487
Os02g0722600		-	0.000000		0.000000			
3	chr02	+	30008280		30008810			Os02t0722650-00 1 531
Os02g0722650		-	2.084746		82.461945			
4	chr02	+	30011066		30015609			Os02t0722700-01 7 1436
Os02g0722700		-	121.268120		4796.750000			
5	chr02	-	30015998		30025935			Os02t0722800-01 13 4485
Os02g0722800	OsWD40-53		4.016716		158.880859			
6	chr02	-	30016149		30018289			Os02t0722800-02 6 1316
Os02g0722800		-	38.850327		1536.721436			
7	chr02	+	30051750		30053275			Os02t0723200-01 1 1526
Os02g0723200	OsGT7		0.394495		15.604217			

- 1転写産物／1行で位置や構造情報、transcript id やgene id、遺伝子名、支持するリード数、発現量 (FPKM) が記載されている。
- 赤点線 ----- は遺伝子座の区切り。

遺伝子発現量の指標（RPKM/FPKM）

シーケンス量や遺伝子の長さで正規化した発現量指標

RPKMやFPKM=Read (Fragment) 数／遺伝子の長さ(kb)／総リード数(M)

RPKM: reads per kilobase of exon model per million mapped reads

FPKM: fragments per kilobase of transcript per million fragments mapped

条件1

リードカウント（≠発現量）

13リード



6リード



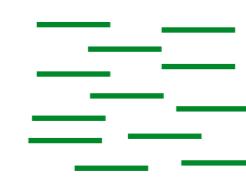
$$\text{RPKM: } 13/2/19 = \mathbf{0.34} \quad 6/1/19 = \mathbf{0.32}$$

条件2

18リード



14リード



$$18/2/32 = \mathbf{0.28} \quad 14/1/32 = \mathbf{0.44}$$

- 全リード中に含まれるある特定の転写産物由来のリードの割合。つまり、サンプル中の全RNAのうちのどれくらいがある特定の転写産物由来かを表す相対的な値。
- 発現する遺伝子の顔ぶれやたくさんの遺伝子の発現量が大きく異なるサンプル間の比較ではサンプル間で発現量を正規化する（発現量の分布を揃える）必要がある。

補足：新規転写産物構造の予測

RNA-Seq

リード配列

ゲノム配列

遺伝子



RNA-Seqのアラインメント
から転写領域の予測が可能

1. サンプルごとにStringtieを実行し、遺伝子構造を予測 (sample*.gtfとして出力)

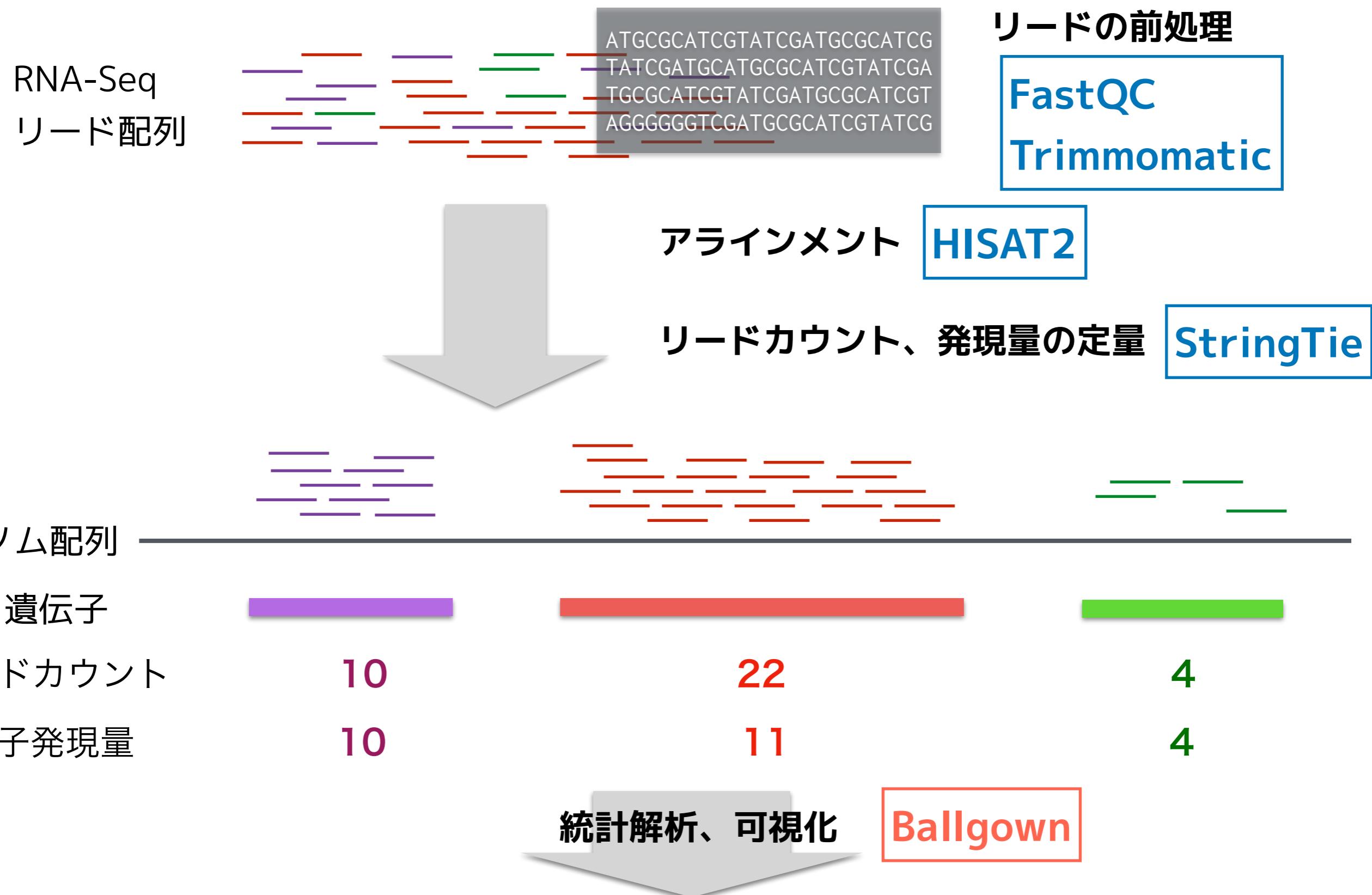
```
$ stringtie -o sample1.gtf -l sample1 sample1.bam  
$ stringtie -o sample2.gtf -l sample2 sample2.bam  
$ ...
```

2. 得られたサンプルごとの遺伝子構造 (*.gtf) をマージする。

```
$ stringtie --merge -G annotation.gtf -o stringtie_merged.gtf assemblies.txt
```

- assemblies.txtは、1.の出力のGTFファイル名が1行ごとに並んだテキストファイル
- -Gオプションで指定することで、既存の遺伝子構造 (annotation.gtf) も含めて、遺伝子構造を統合してくれる。
- step4のStringtie実行時に、この統合された遺伝子構造 (stringtie_merged.gtf) を指定することで予測遺伝子も含めた発現量が得られる。

リファレンス情報を用いたRNA-Seq解析の流れ



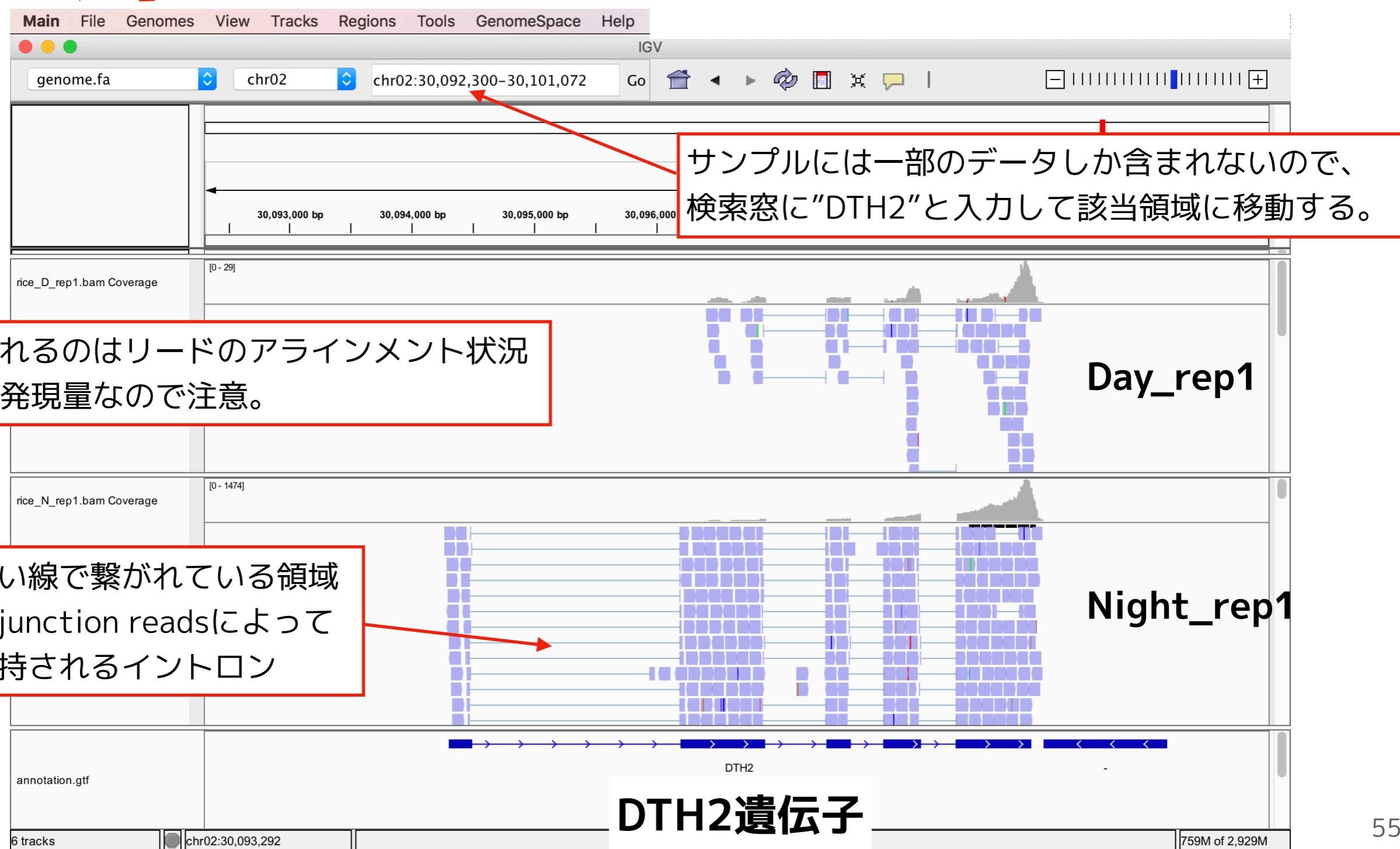
2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

IGVを用いたRNA-Seqデータの可視化



デスクトップ上の「workshop/RNA-Seq/useref/alignment」中のリファレンスゲノムと遺伝子アノテーション、HISAT2が出力するBAMファイルを読み込む。

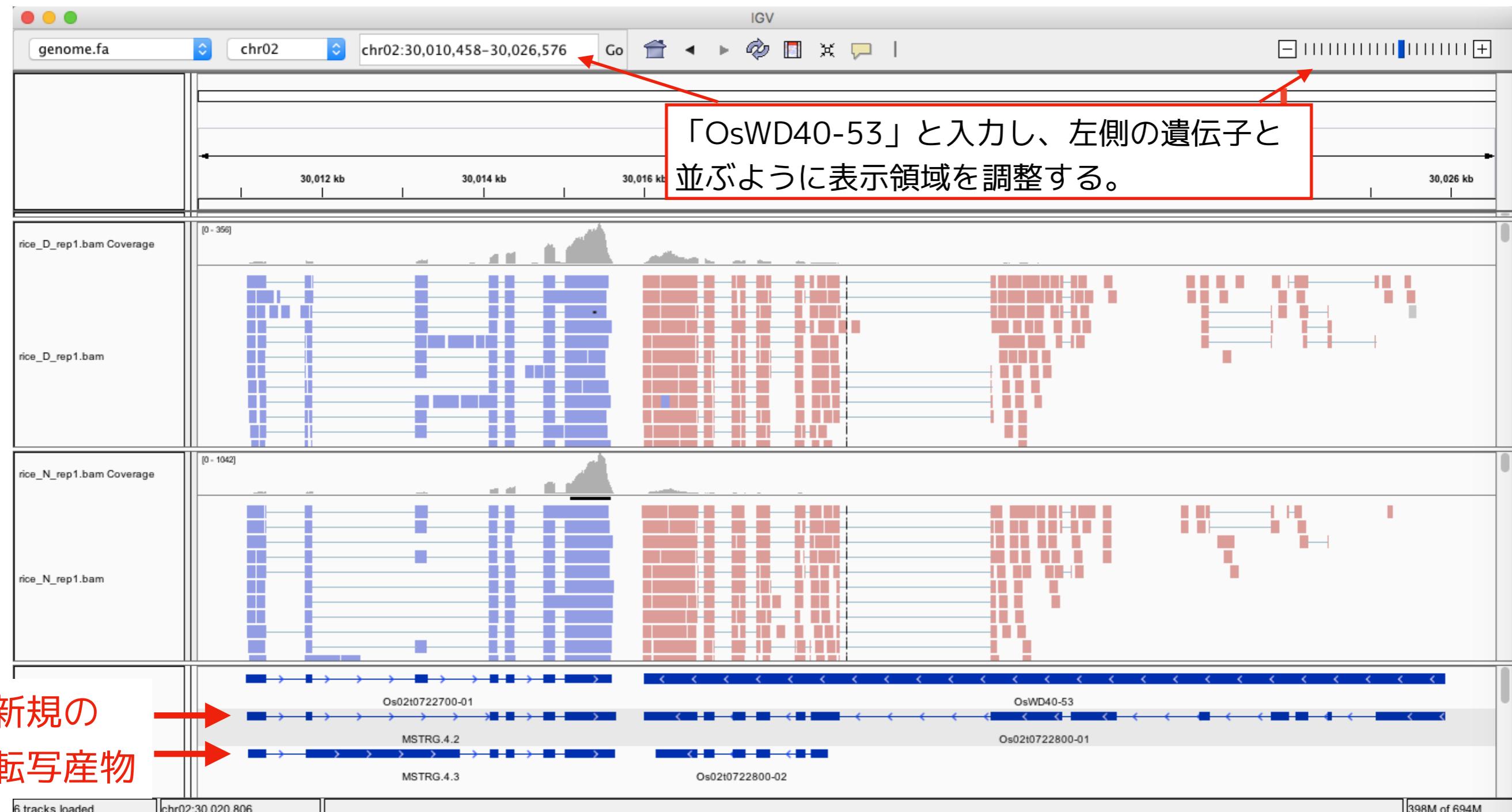
2. 「Load from File」からGTFファイルやBAMファイルを読み込む
1. 「Load Genome from File」からFASTAファイル（ゲノム配列）を読み込む



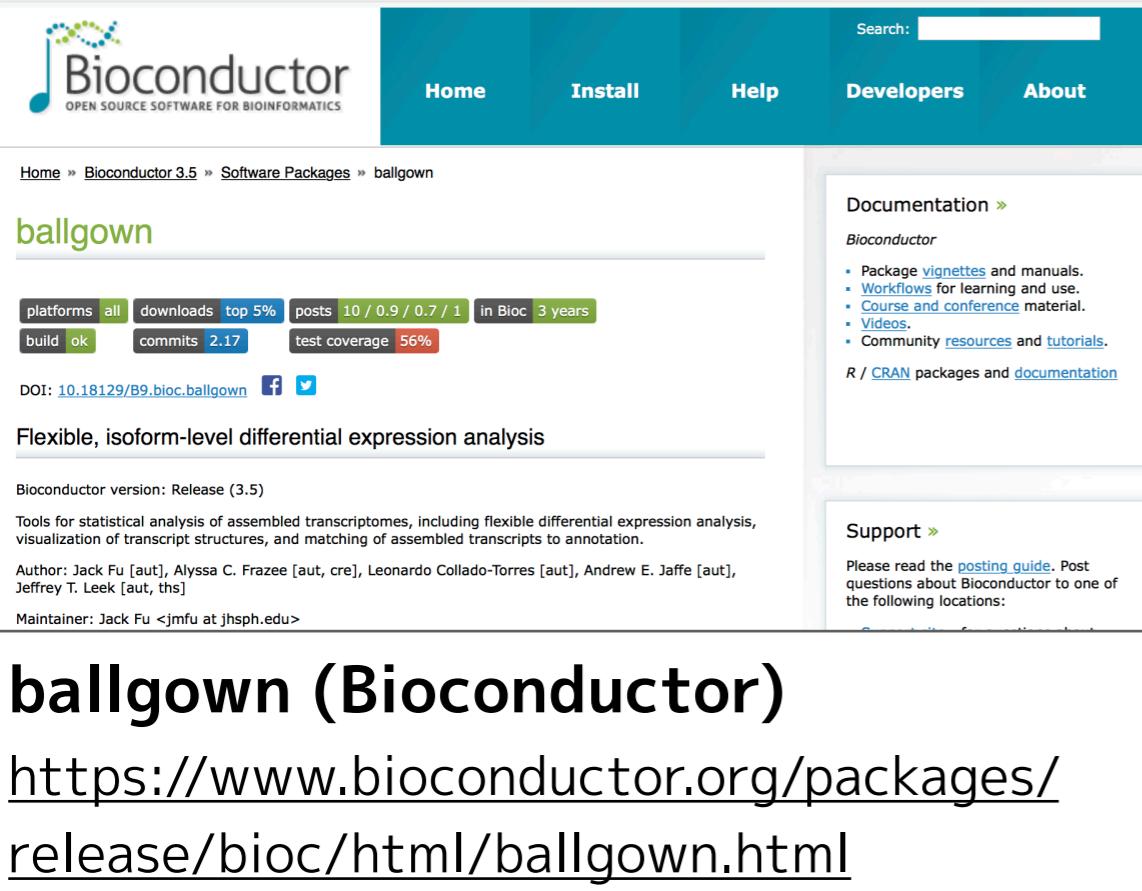
IGVによるRNA-Seqデータのアライメント、遺伝子構造の可視化



- 本演習データはIllumina stranded mRNA-Seq法によるものなので、ペアのリード2の向きと転写方向が一致する（アライメントのトラック上で右クリックし、[Color alignments by] -> [first-of-pair strand] を選択すると色で区別できるようになる）。
- 新規の転写産物構造にはMSTRG.XX.XXといったIDが振られている。



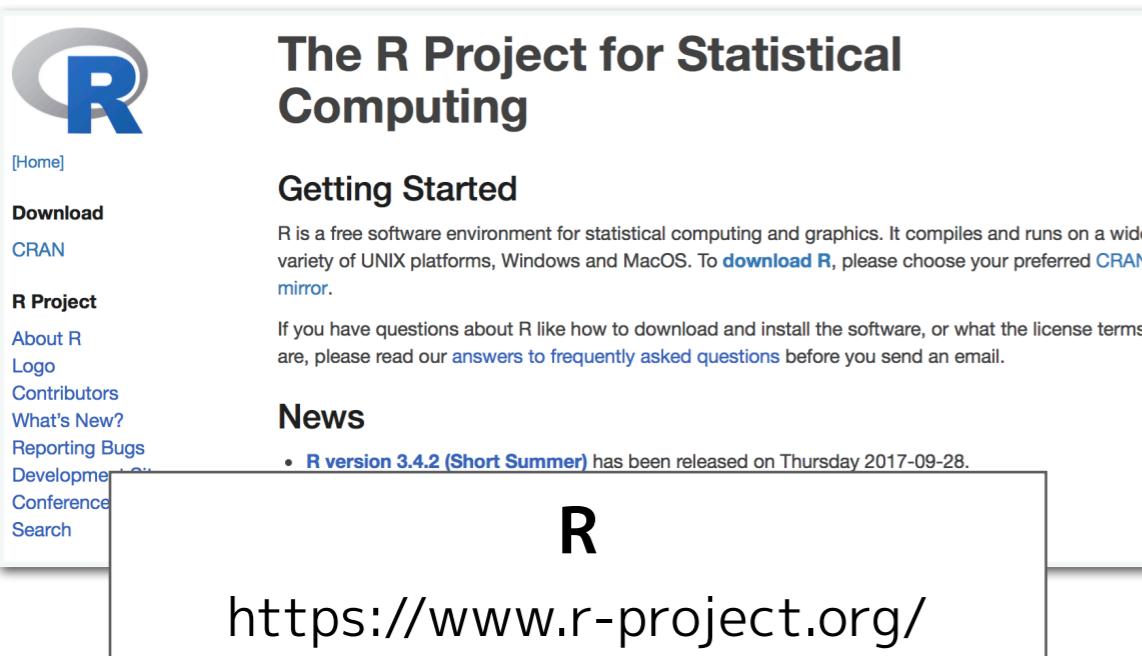
R/Bioconductorによる統計解析と可視化



The screenshot shows the Bioconductor website for the ballgown package. The top navigation bar includes links for Home, Install, Help, Developers, and About. A search bar is also present. Below the navigation, the package name 'ballgown' is displayed, along with its version (3.5) and release date (2017-09-28). Key statistics like platforms (all), downloads (top 5%), posts (10 / 0.9 / 0.7 / 1), build status (ok), commits (2.17), and test coverage (56%) are shown. A DOI link (10.18129/B9.bioc.ballgown) and social media links for Facebook and Twitter are included. The main content area describes 'Flexible, isoform-level differential expression analysis' and provides documentation and support links.

ballgown (Bioconductor)

<https://www.bioconductor.org/packages/release/bioc/html/ballgown.html>



The screenshot shows the R Project for Statistical Computing homepage. It features the R logo and navigation links for Home, Download, CRAN, R Project, About R, Logo, Contributors, What's New?, Reporting Bugs, Development, Conference, and Search. The main content area includes sections for 'Getting Started' (with information about R being a free software environment for statistical computing and graphics), 'News' (mentioning the release of R version 3.4.2), and a large 'R' logo. A URL 'https://www.r-project.org/' is highlighted at the bottom.

ballgownは、R/Bioconductorのパッケージの一つであり、StringTieの結果を読み込んで、「発現比較解析（統計解析）」や「発現情報の可視化」の手法を提供している。

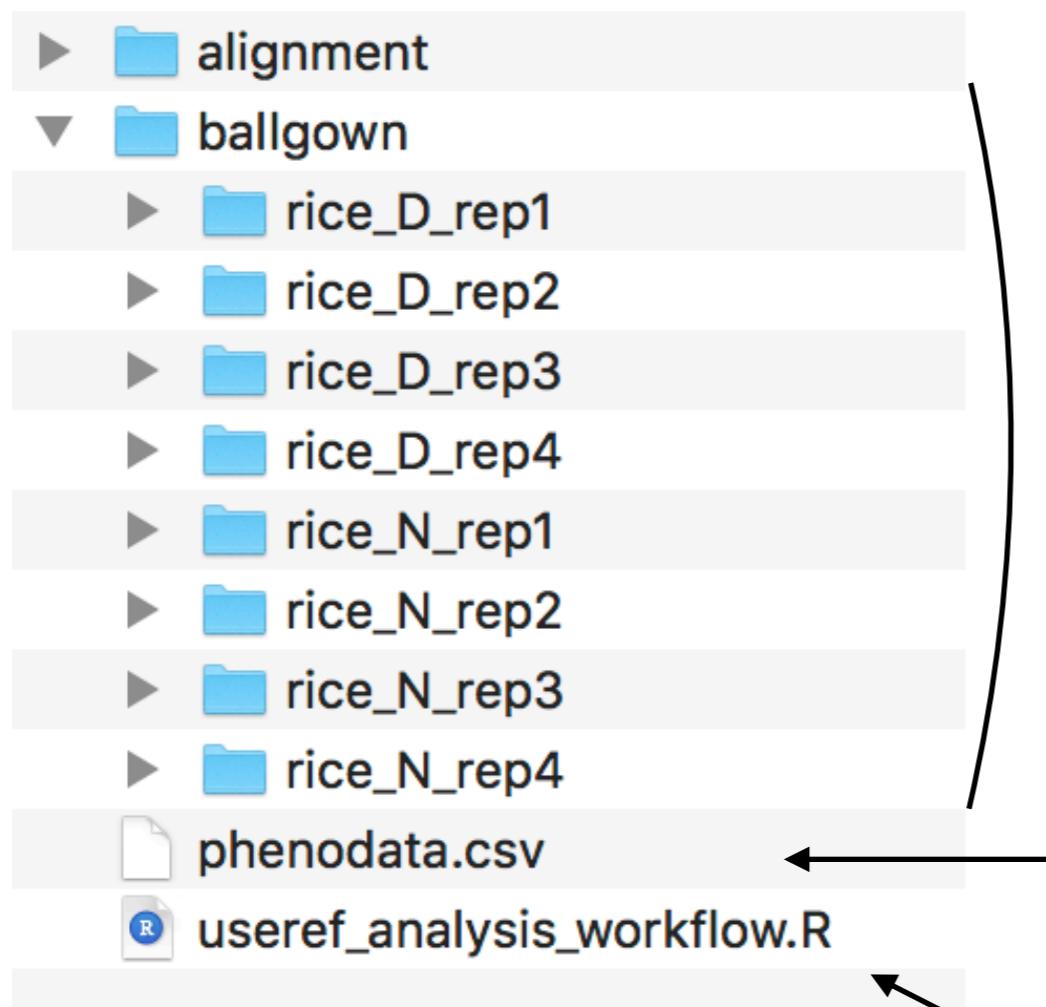
利用するためには統計計算とグラフィックスのための言語・環境である「R」をインストールする必要がある。また、RStudioはRの統合開発環境であり、使いやすいGUIを備えておりとても便利。



The screenshot shows the RStudio homepage. It features the RStudio logo and navigation links for rstudio::conf, Products, Resources, Pricing, About Us, and Blogs. A search icon is also present. The main content area highlights 'RStudio' as 'Open source and enterprise-ready professional software for R'. It includes links for 'Download RStudio', 'Discover Shiny', 'shinyapps.io Login', and 'Discover RStudio Connect'. Below this, there are images of the RStudio interface (Code, Workspace, Plots, Console) and several R packages: rmarkdown, Shiny, tidyverse, ggplot2, and lattice. A large 'RStudio' logo and its URL 'https://www.rstudio.com/' are prominently displayed at the bottom.

データやスクリプトの置かれているディレクトリの確認

デスクトップ上の「NGSworkshop/Data/RNA-Seq/useref」ディレクトリ
中にある、以下の3つのデータを用いて解析を行なう。



1. 「ballgown」ディレクトリにある8サンプル分（2条件、4反復）のStringTieの結果
2. サンプル名や条件を記載した、カンマ区切りのテキストファイル（phenodata.csv）
3. 実行するコマンドを記載したRのスクリプト（useref_analysis_workflow.R）



デフォルトでは4つの区画（ペイン）に分かれている

The screenshot shows the RStudio interface with four main panes:

- Script Editor (Left):** Displays an R script named "useref_analysis_workflow.R". A box highlights the title "Rスクリプトの編集".
- Environment (Top Right):** Shows the Global Environment with objects like "bg", "bg_filtered", and "pheno_data". A box highlights the title "変数一覧" and lists "変数一覧" and "作業履歴".
- Files (Bottom Right):** Displays a file tree and a list of files in the current directory. A box highlights the title "ディレクトリ構造" and lists "ディレクトリ構造", "プロット", and "ヘルプ".
- Console (Bottom Left):** Shows R command-line output. A box highlights the title "Rコンソール" and lists "Rコンソール" and "Wrapping up the results".

作業ディレクトリの指定

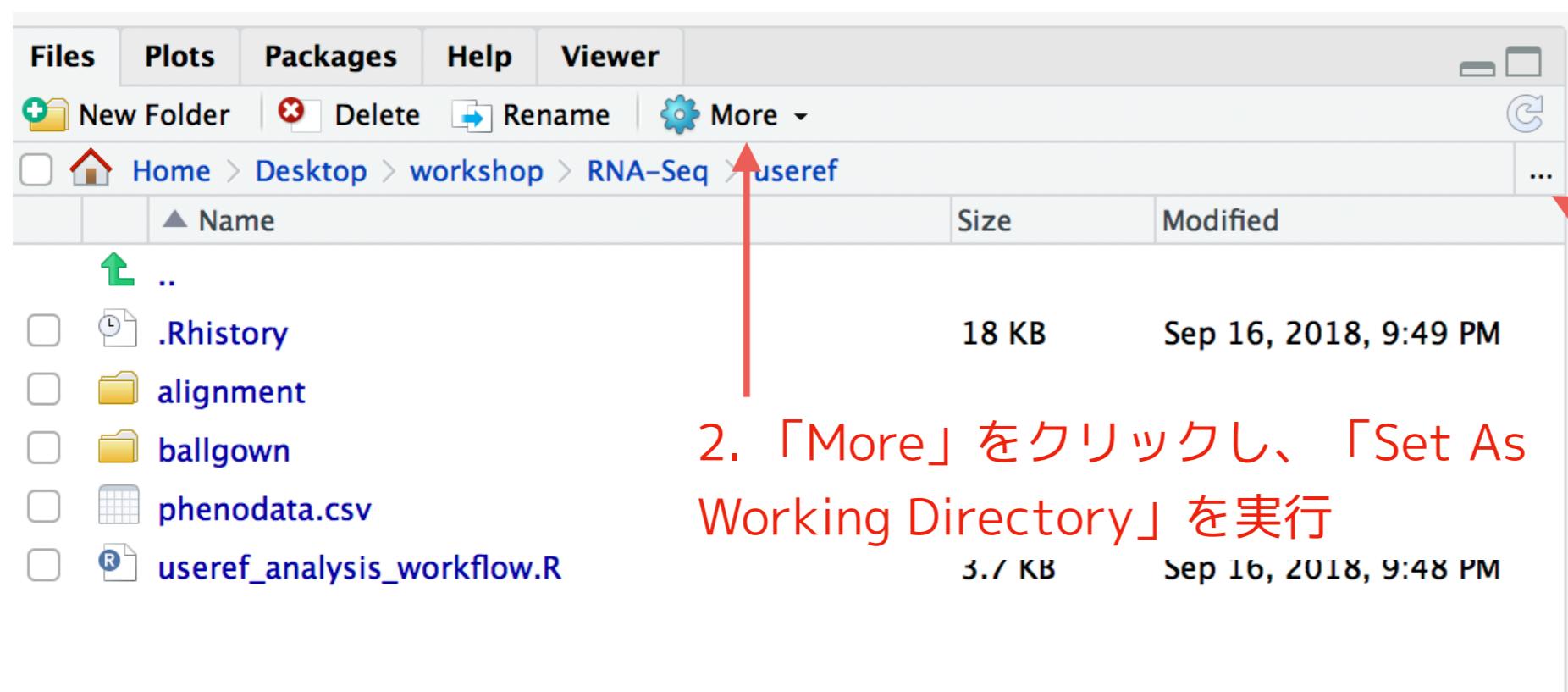
1. Rコンソール（左下のペイン）でコマンドを実行

```
> setwd("c:/Users/user/Desktop/NGSworkshop/Data/RNA-Seq/")
```

ballgownディレクトリがある場所を指定する。

もしくは、

2. ディレクトリ構造が表示されている右下のペインを操作



The screenshot shows the RStudio file browser interface. The top navigation bar includes Files, Plots, Packages, Help, and Viewer. Below the toolbar are buttons for New Folder, Delete, Rename, and More. The main area displays a file tree with the path Home > Desktop > workshop > RNA-Seq > useref. The 'More' button has a red arrow pointing to it. The 'useref' folder is selected. The bottom part of the interface shows a list of files and folders:

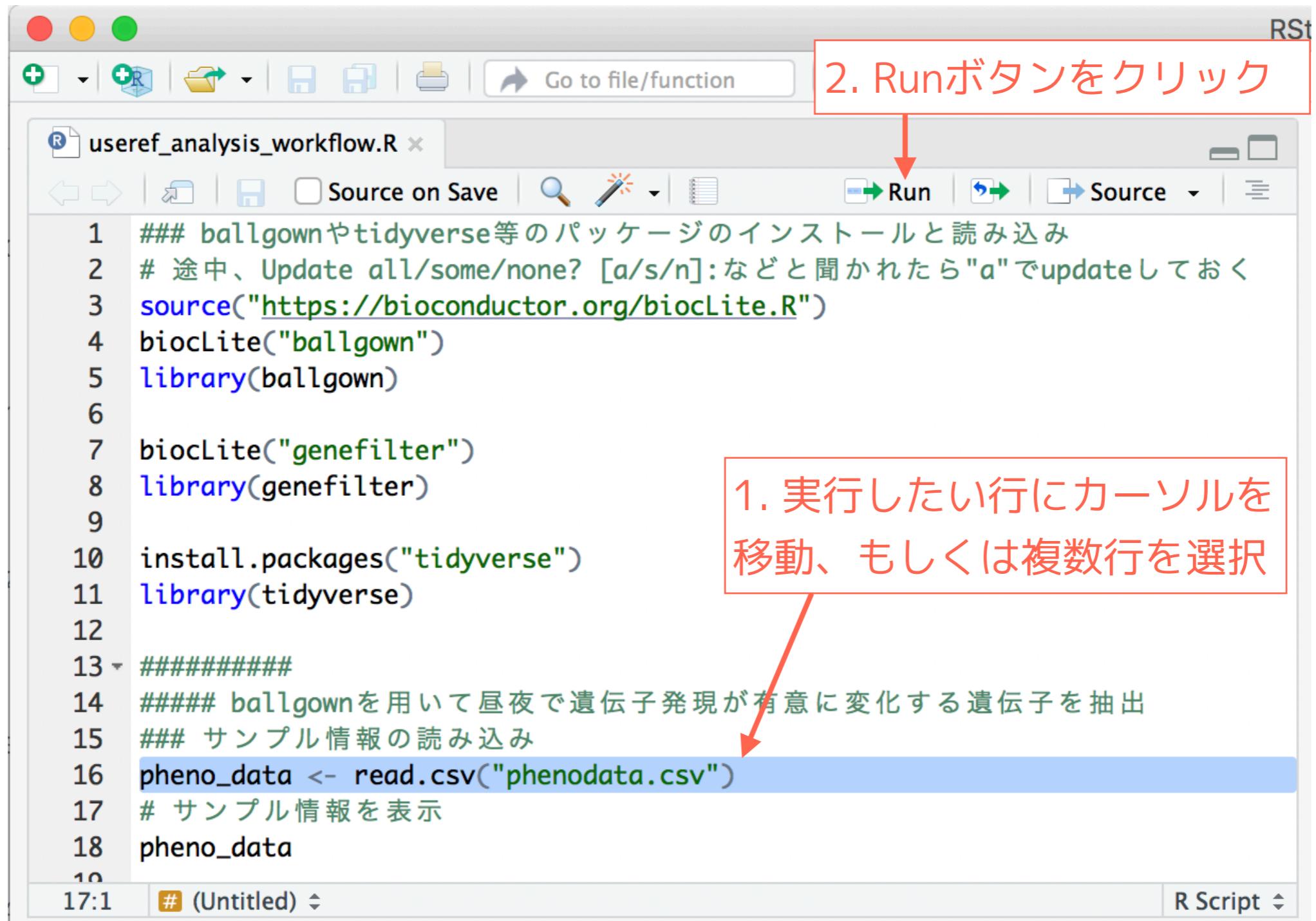
Name	Size	Modified
..	18 KB	Sep 16, 2018, 9:49 PM
.Rhistory	3.7 KB	Sep 16, 2018, 9:48 PM
alignment		
ballgown		
phenodata.csv		
useref_analysis_workflow.R		

Red annotations provide instructions:

2. 「More」をクリックし、「Set As Working Directory」を実行

1. 「…」をクリックし、「RNA-Seq」ディレクトリを選択

Rスクリプト中のコマンドを順次実行していく



The screenshot shows an RStudio interface with the following code in the script:

```
1  ### ballgownやtidyverse等のパッケージのインストールと読み込み
2  # 途中、Update all/some/none? [a/s/n]:などと聞かれたら"a"でupdateしておく
3  source("https://bioconductor.org/biocLite.R")
4  biocLite("ballgown")
5  library(ballgown)
6
7  biocLite("genefilter")
8  library(genefilter)
9
10 install.packages("tidyverse")
11 library(tidyverse)
12
13 #####
14 ##### ballgownを用いて昼夜で遺伝子発現が有意に変化する遺伝子を抽出
15 ### サンプル情報の読み込み
16 pheno_data <- read.csv("phenodata.csv")
17 # サンプル情報を表示
18 pheno_data
19
20
21:1 # (Untitled) R Script
```

Two red annotations are present:

- A box around the "Run" button in the toolbar with the text "2. Runボタンをクリック" (Click the Run button).
- A box around the line "pheno_data <- read.csv("phenodata.csv")" with the text "1. 実行したい行にカーソルを移動、もしくは複数行を選択" (Move the cursor to the line you want to execute, or select multiple lines).

シェルスクリプトと同様、どのような解析をおこなったかをあとで見直す際や、データやパラメータを変えて同じ解析をする際などにRスクリプトとして保存しておくと便利。

パッケージのインストールと読み込み

ballgownパッケージのインストールと読み込み

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("ballgown")
> library(ballgown) ☆
```

- ・ ballgownはR/Bioconductorのパッケージとして提供されており、biocLite関数でパッケージのインストールする。
- ・ 途中、「Update all/some/none? [a/s/n]:」と聞かれたら、「a」と入力しリターンを押し、全てアップデートする。多少エラーが出ても問題ないこともある。

さらに演習で使う2つのパッケージをインストールして読み込む。

```
> biocLite("genefilter") ← ballgownと同様にR/Bioconductorのパッケージ
> library(genefilter) ☆
```

```
> install.packages(tidyverse) ← Rのパッケージのため、install.packages関数を用いて
> library(tidyverse) ☆      インストールする。
```

*演習ではパッケージのインストールは完了しているため、☆印のついたlibrary関数によるパッケージの読み込みだけを行えばOKです。

サンプル情報 (phenodata.csv)

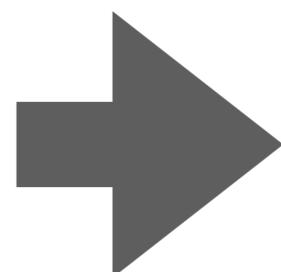


サンプル情報の読み込み

```
> pheno_data <- read.csv("phenodata.csv")
```

カンマ区切りの書式で、サンプル名、
実験条件などを記載したファイル
(phenodata.csv) を準備する。

```
ids,cond,rep ←———— 1行目は項目名  
rice_D_rep1,Day,1  
rice_D_rep2,Day,2  
rice_D_rep3,Day,3  
rice_D_rep4,Day,4  
rice_N_rep1,Night,1  
rice_N_rep2,Night,2  
rice_N_rep3,Night,3  
rice_N_rep4,Night,4
```



上のコマンドを実行すると
データフレームとして読み込まれる。

```
> pheno_data  
      ids cond rep  
1 rice_D_rep1 Day  1  
2 rice_D_rep2 Day  2  
3 rice_D_rep3 Day  3  
4 rice_D_rep4 Day  4  
5 rice_N_rep1 Night 1  
6 rice_N_rep2 Night 2  
7 rice_N_rep3 Night 3  
8 rice_N_rep4 Night 4
```

StringTieデータをballgownオブジェクトに格納する



StringTieデータを読み込み、ballgownオブジェクトに格納することで、様々な解析が可能になる。

```
> bg <- ballgown(dataDir="ballgown", samplePattern="rice", pData=pheno_data)
Mon Sep 10 08:00:18 2018
Mon Sep 10 08:00:18 2018: Reading linking tables
Mon Sep 10 08:00:18 2018: Reading intron data files
Mon Sep 10 08:00:20 2018: Merging intron data
Mon Sep 10 08:00:20 2018: Reading exon data files
Mon Sep 10 08:00:24 2018: Merging exon data
Mon Sep 10 08:00:25 2018: Reading transcript data files
Mon Sep 10 08:00:26 2018: Merging transcript data
Wrapping up the results
Mon Sep 10 08:00:26 2018
```

ballgownオブジェクトを指定してみると

```
> bg
ballgown instance with 44,586 transcripts and 8 samples
```

44,586 transcriptsについて、8サンプル分の遺伝子発現情報が読み込まれていることが分かる。

昼夜で遺伝子発現量が変化した遺伝子（DEG）の検出



ballgownのsubset関数を使い、サンプル間のFPKMの分散が1以上の転写産物に絞り込む、つまり発現変動が小さいものを除く

```
> bg_filtered <- subset(bg, "rowVars(expr(bg))>=1")
> bg_filtered
ballgown instance with 19991 transcripts and 8 samples
```

転写産物数が19,991 transcriptsとなり、絞り込まれている。

転写産物レベルでサンプル（Day/Night）間での遺伝子発現変動を検定し、p-valueやq-valueを計算する

```
> results_transcripts <- stattest(bg_filtered, feature="transcript",
covariate="cond", getFC=TRUE, meas="FPKM")
```

- covariateで比較対象の列名「cond」、measで利用する発現量の指標「FPKM」、getFCで結果にFold-changeを出力するよう指定している。
- 他にもタイムコースサンプルの場合に指定するものなど複数のオプションがある。

昼夜で遺伝子発現量が変化した遺伝子（DEG）の検出



検定の結果の確認。各転写産物ごとにDay/Night間のFold-changeやp-value、q-valueが計算されている。

```
> head(results_transcripts)
  feature id      fc      pval      qval
1 transcript 1 0.9180662 0.5169128022 0.68235630
2 transcript 6 0.8906488 0.2519004300 0.44433240
3 transcript 7 1.0106792 0.9216822885 0.95930393
4 transcript 9 0.4228804 0.1215415562 0.29074712
5 transcript 10 0.9335729 0.4569417800 0.63598991
6 transcript 11 1.4025491 0.0003284612 0.02168817
```

- ・このままでは遺伝子IDや遺伝子名などもなく分かりにくい。
- ・IDや遺伝子名の追加、q-valueによるソートなどを行うとよい。

昼夜で遺伝子発現量が変化した遺伝子（DEG）の検出



遺伝子名や遺伝子ID、転写産物IDの列を追加する。

```
> results_transcripts = data.frame(geneNames=ballgown::geneNames(bg_filtered),  
geneIDs=ballgown::geneIDs(bg_filtered),  
transcriptIDs=ballgown::transcriptNames(bg_filtered), results_transcripts)
```

検定結果をp-valueの小さい順にソートする

```
> results_transcripts = arrange(results_transcripts,pval)
```

適当なq-valueの閾値(qval < 0.01)で切った遺伝子を抽出する（75転写産物がヒット）。

```
> subset(results_transcripts, results_transcripts$qval<0.01)
```

	geneNames	geneIDs	transcriptIDs	feature	id	fc	pval	qval
1	- Os02g0623932	Os02t0623932-00	transcript	9347	1.334693e-01	3.016259e-07	0.003330334	
2	LHY	Os04g0583900	Os04t0583900-01	transcript	19641	2.584726e+02	3.331834e-07	0.003330334
3	- Os06g0660800	Os06t0660800-01	transcript	27355	6.321374e+00	6.792785e-07	0.003964490	
4	OsBBX19, OsM, DTH2, qDTH-2	Os06g0298200	Os06t0298200-01	transcript	25939	6.919100e+01	9.295324e-07	0.003964490
5	PsbP	Os03g0279950	Os03t0279950-01	transcript	13109	2.380650e+01	9.915688e-07	0.003964490
6	- Os08g0360100	Os08t0360100-01	transcript	32883	1.538616e-01	1.254840e-06	0.004073630	
7	- Os06g0281400	Os06t0281400-01	transcript	25839	3.943439e-01	1.426413e-06	0.004073630	
...								
38	SIGA	Os08g0163400	Os08t0163400-01	transcript	31942	1.358213e-01	1.574039e-05	0.008169480
...								

遺伝子発現量の分布の可視化

全サンプルの平均発現量が1以上の転写産物を選び、遺伝子発現量（FPKM値に0.01を足してlog2をとったもの）を取得する（22,292個の転写産物）。

```
> bg.expressed <- subset(bg, "rowMeans(expr(bg))>=1")
> bg.expressed
ballgown instance with 22292 transcripts and 8 samples
> log2fpkm <- log2(expr(bg.expressed, meas="FPKM") + 0.01)
```

変数log2fpkmには、転写産物・サンプルごとに全FPKM値が格納されている。

```
> head(log2fpkm)
   FPKM.rice_D_rep1 FPKM.rice_D_rep2 FPKM.rice_D_rep3 FPKM.rice_D_rep4 FPKM.rice_N_rep1
1      3.539679      3.831836      4.0835609      4.1796822      3.897224
4      1.147461      1.055207      0.5679202     -0.4774179      1.388669
6      3.677559      3.757414      4.0814173      3.9224231      3.593344
7      3.333910      4.173667      4.0908662      4.1635102      3.890806
9      5.925956      6.953405      6.8972279      4.9053619      5.080402
10     3.458069      4.015651      4.1832976      4.0796425      3.893308
   FPKM.rice_N_rep2 FPKM.rice_N_rep3 FPKM.rice_N_rep4
1      4.2026522     4.325626      3.7204283
4      0.1995184     1.501587      0.6323222
6      3.8188390     4.363668      4.1252938
7      4.4518494     4.566686      4.6076217
9      5.5129445     5.091533      4.7157670
10     4.1539759     4.123816      4.2123396
```

遺伝子発現量の分布の可視化



データを整形する。

```
> log2fpkm.long <- as.data.frame(log2fpkm) %>%  
  tidyr::gather(Dataset, FPKM)
```

ワイドからロングフォーマットのデータフレームに変換されている。

```
> head(log2fpkm.long)  
  Dataset      FPKM  
1 FPKM.rice_D_rep1 3.539679  
2 FPKM.rice_D_rep1 1.147461  
3 FPKM.rice_D_rep1 3.677559  
4 FPKM.rice_D_rep1 3.333910  
5 FPKM.rice_D_rep1 5.925956  
6 FPKM.rice_D_rep1 3.458069
```

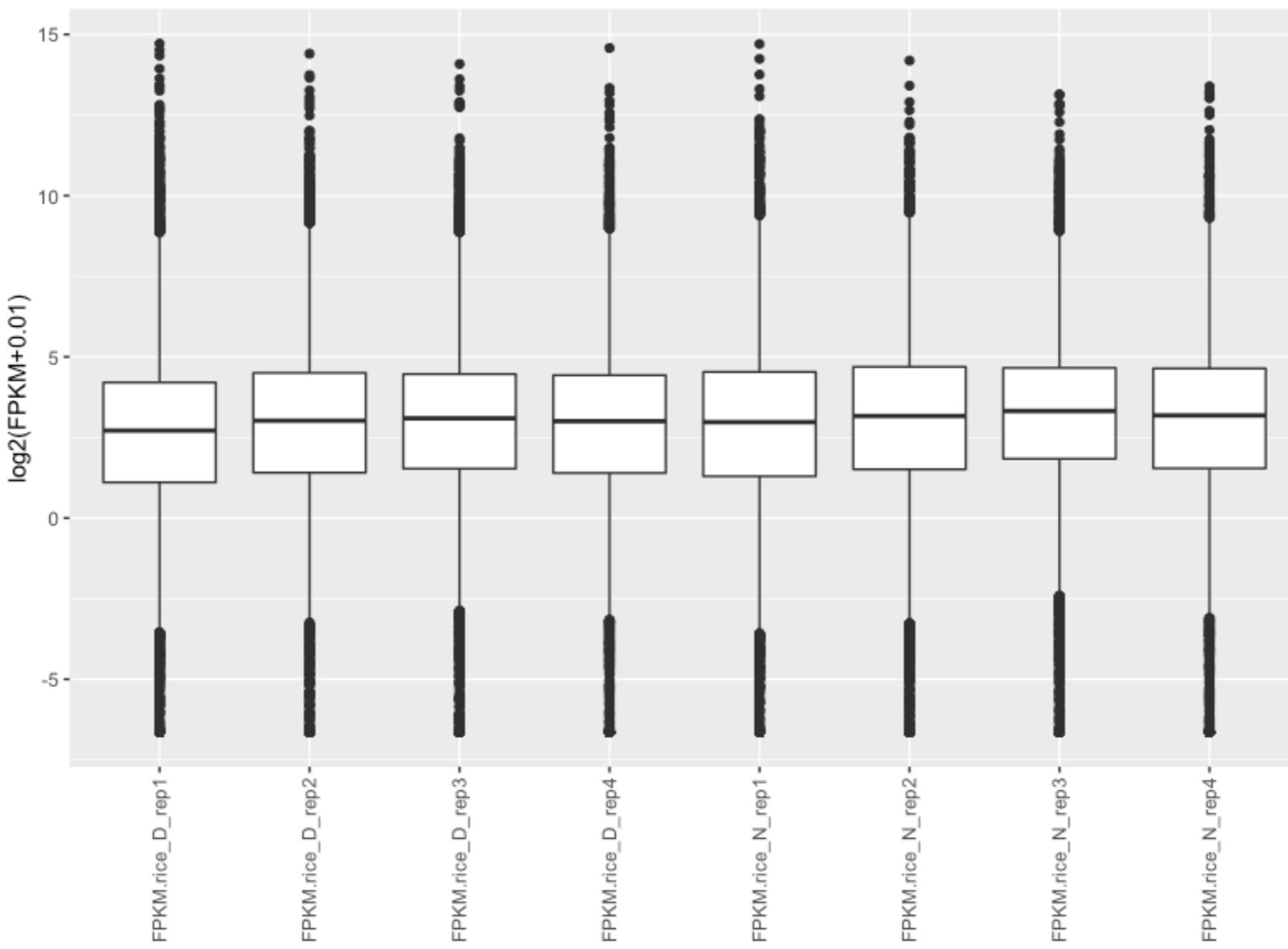
このような書式にしておくと、グラフ描画関数 ggplot でのグラフ化が容易になる。

データ解析の大部分はこのようなデータのフィルタリングや整形などの地味な仕事。

遺伝子発現量の分布の可視化

箱ひげ図を描く。

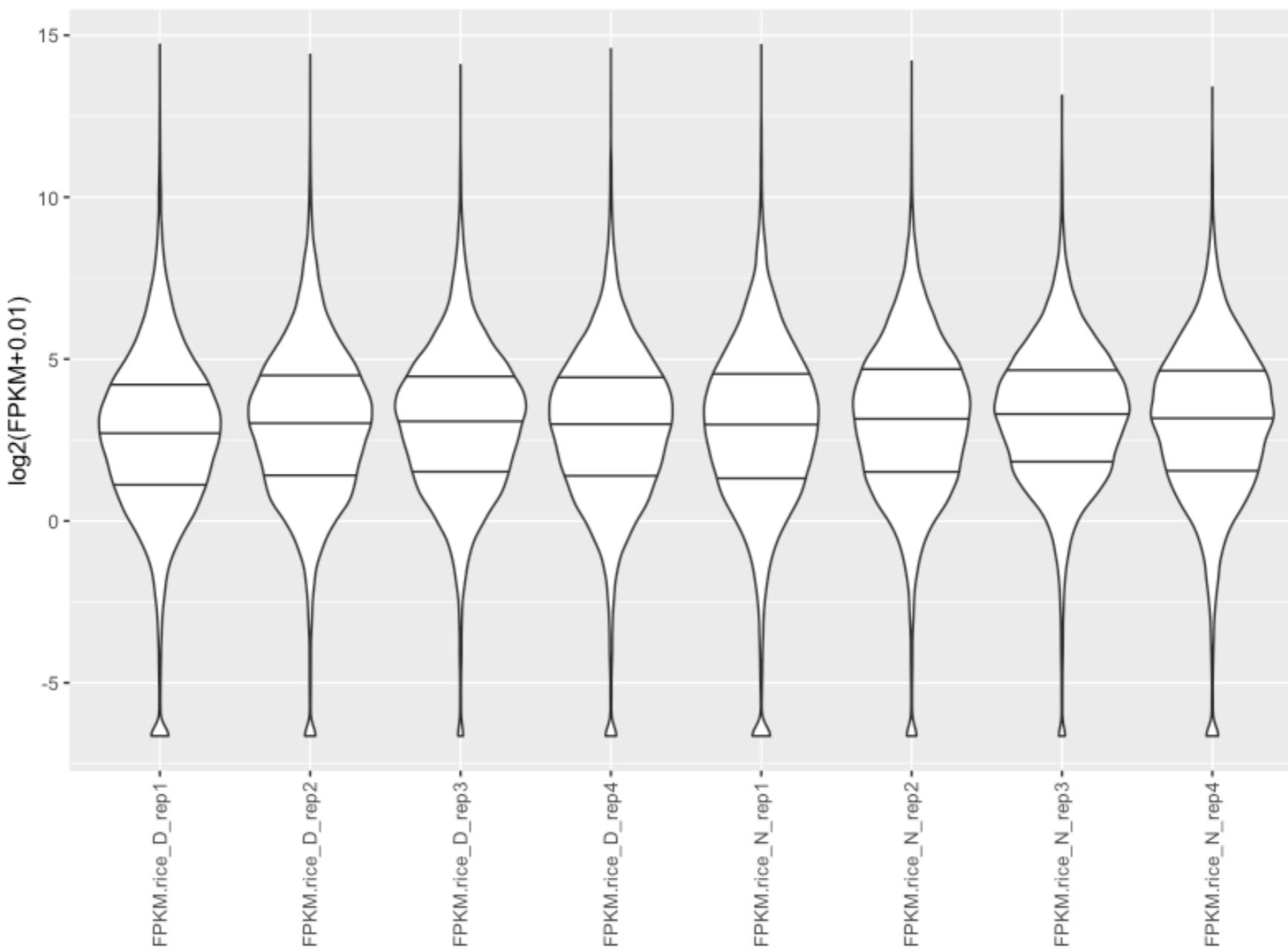
```
> ggplot(log2fpkm.long, aes(y=FPKM, x=Dataset)) + geom_boxplot() +  
  theme(axis.text.x = element_text(angle=90, hjust=0, vjust=.5)) +  
  xlab("") + ylab("log2(FPKM+0.01)")
```



遺伝子発現量の分布の可視化

同じデータを使って、ヴァイオリンプロットを描いてみる。

```
> ggplot(log2fpkm.long, aes(y=FPKM, x=Dataset)) +  
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75)) +  
  theme(axis.text.x = element_text(angle=90, hjust=0, vjust=.5)) +  
  xlab("") + ylab("log2(FPKM+0.01)")
```



特定の遺伝子の構造や発現プロファイルを可視化する



ballgownオブジェクトに入れる際に振られた"id"で遺伝子を指定する必要があるため、遺伝子のIDや遺伝子名と「ballgownが割り振るID」との対応を調べる。

```
> subset(results_transcripts, results_transcripts$qval<0.01)
```

	geneNames	geneIDs	transcriptIDs	feature	id	fc	pval	qval
1	-	Os02g0623932	Os02t0623932-00	transcript	9347	1.334693e-01	3.016259e-07	0.003330334
2	LHY	Os04g0583900	Os04t0583900-01	transcript	19641	2.584726e+02	3.331834e-07	0.003330334
3	-	Os06g0660800	Os06t0660800-01	transcript	27355	6.321374e+00	6.792785e-07	0.003964490
4	OsBBX19, OsM, DTH2, qDTH-2	Os06g0298200	Os06t0298200-01	transcript	25939	6.919100e+01	9.295324e-07	0.003964490
5	PsbP	Os03g0279950	Os03t0279950-01	transcript	13109	2.380650e+01	9.915688e-07	0.003964490
6	-	Os08g0360100	Os08t0360100-01	transcript	32883	1.538616e-01	1.254840e-06	0.004073630
7	-	Os06g0281400	Os06t0281400-01	transcript	25839	3.943439e-01	1.426413e-06	0.004073630
...								
38	SIGA	Os08g0163400	Os08t0163400-01	transcript	31942	1.358213e-01	1.574039e-05	0.008169480
...								

- ballgown id: 19641 = LHY = Os04t0583900-01
 - Ballgown id: 31942 = SIGA = Os08t0163400-01
- であることが分かる。

特定の遺伝子の構造や発現プロファイルを可視化する



LHY遺伝子（ballgown id: 19641）の発現量の箱ひげ図を作成し、個々のサンプルの発現量を個別に重ねてプロットする。

まずはballgown idを指定し、特定の転写産物の発現量データを取り出す。

```
> log2fpkm["19641", ]  
FPKM.rice_D_rep1 FPKM.rice_D_rep2 FPKM.rice_D_rep3 FPKM.rice_D_rep4  
-2.0672131      -0.7016209      -3.1568201      -4.5881052  
FPKM.rice_N_rep1 FPKM.rice_N_rep2 FPKM.rice_N_rep3 FPKM.rice_N_rep4  
8.2926856       8.4396024       7.9646139       8.1115949
```

特定の遺伝子の構造や発現プロファイルを可視化する

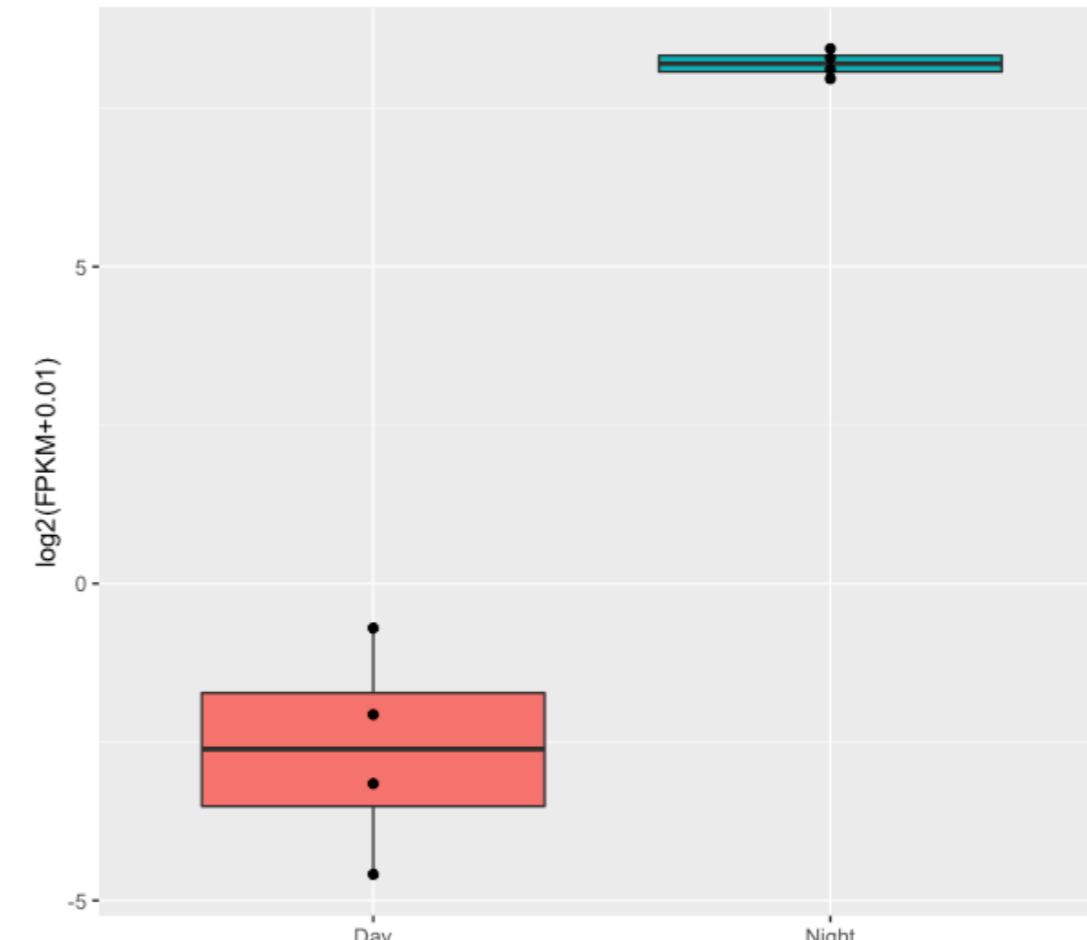
取り出した発現量データを整形し、ggplotで箱ひげ図と散布図を描画。

```
> log2fpkm.LHY.long <- as.data.frame(log2fpkm["19641", ]) %>%  
  tidyr::gather(Dataset, FPKM)  
> log2fpkm.LHY.long <- data.frame(log2fpkm.LHY.long, Condition=c(rep("Day", 4),  
rep("Night", 4)))  
> ggplot(log2fpkm.LHY.long, aes(x=Condition, y=FPKM, fill=Condition)) +  
  geom_boxplot() + geom_point() +  
  xlab("") + ylab("log2(FPKM+0.01)") + theme(legend.position = "bottom")
```

描画用のデータの確認

```
> log2fpkm.LHY.long
```

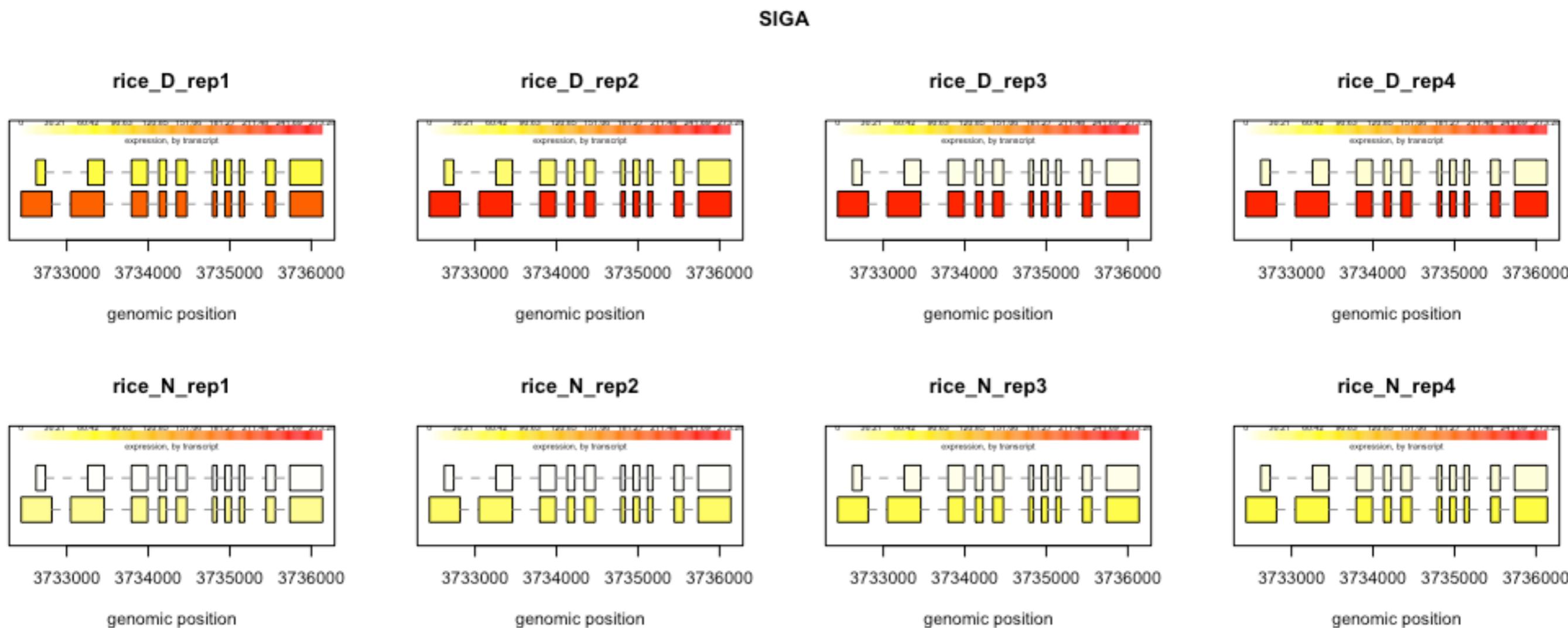
	Dataset	FPKM	Condition
1	log2fpkm["19641",]	-2.0672131	Day
2	log2fpkm["19641",]	-0.7016209	Day
3	log2fpkm["19641",]	-3.1568201	Day
4	log2fpkm["19641",]	-4.5881052	Day
5	log2fpkm["19641",]	8.2926856	Night
6	log2fpkm["19641",]	8.4396024	Night
7	log2fpkm["19641",]	7.9646139	Night
8	log2fpkm["19641",]	8.1115949	Night



特定の遺伝子の構造や発現プロファイルを可視化する

sigA遺伝子 (ballgown id: 31942) の転写産物構造と発現量を合わせて可視化する。

```
> plotTranscripts(ballgown::geneIDs(bg)[31942], bg, main=c('SIGA'),  
sample=c('rice_D_rep1','rice_D_rep2','rice_D_rep3','rice_D_rep4','rice_N_rep1',  
'rice_N_rep2','rice_N_rep3','rice_N_rep4'))
```



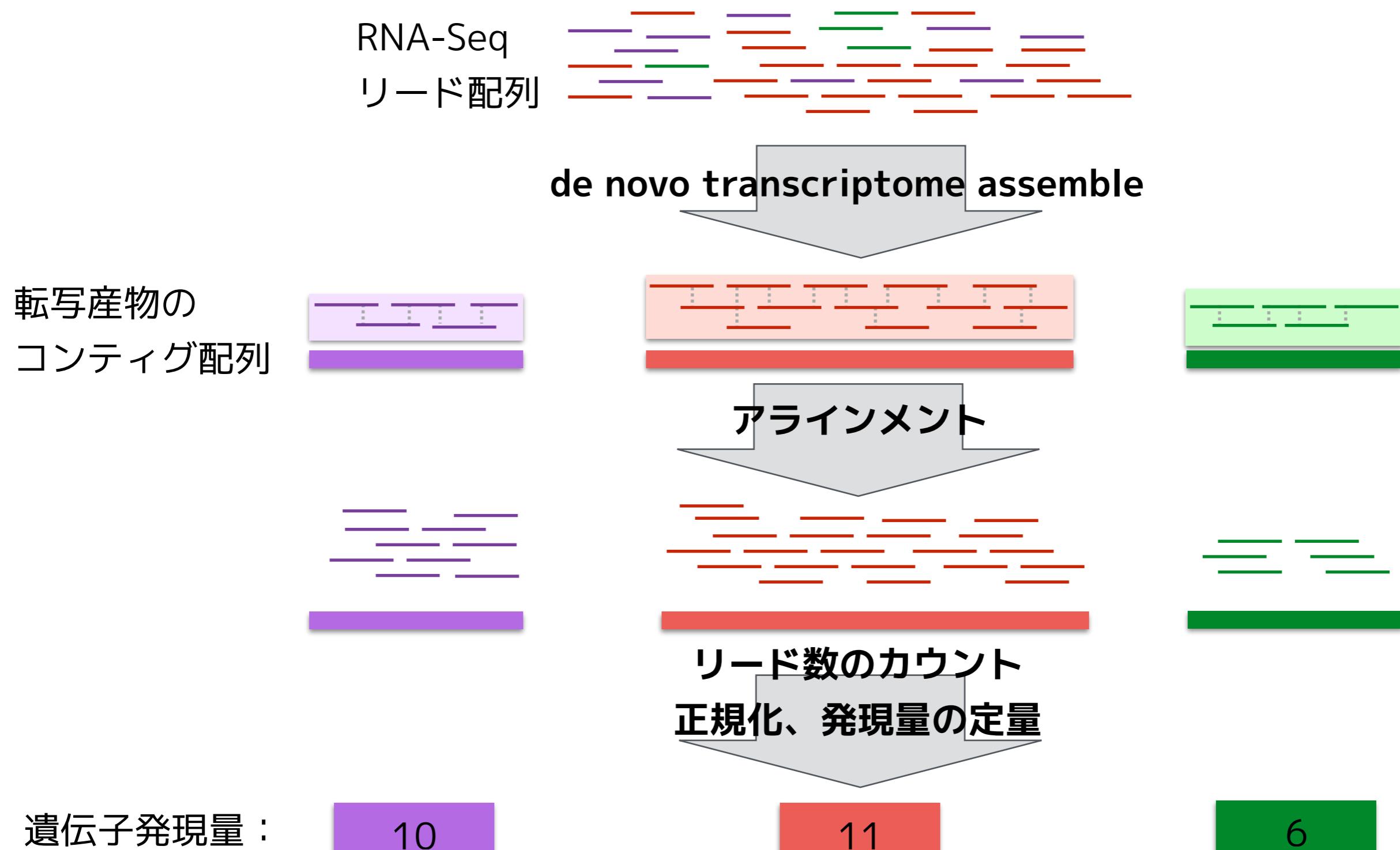
転写調節に関するイネのsigA遺伝子の発現量は昼に高く、夜に低いパターンを示し、2つあるsplicing isoformうちの一方が主に発現していることが分かる。

- ▶ リファレンス情報を用いたRNA-Seq解析
- ▶ **De novo**トランスクリプトーム解析
- ▶ RNA-Seqデータを用いた多型検出

de novo transcriptome assemble法を用いた遺伝子発現解析の流れ



まず始めに、RNA-Seqリード配列から転写産物の全長配列を再構築する。
アセンブルされた転写産物配列上にRNA-Seqリードをアラインメントし、転写産物ごとにリード数をカウントすることにより遺伝子発現を定量する。



遺伝子発現量：

10

11

6

De novo transcriptome assemble

良い点

- ・ゲノム配列を解読することなく、効率よく遺伝子配列を得ることができる上に発現情報も一緒に得られる。

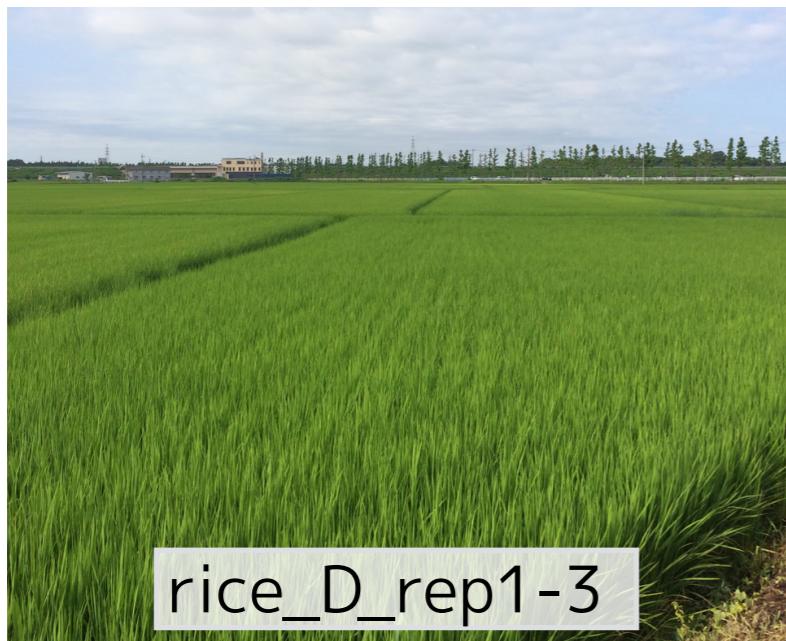
悪い点

- ・リファレンスゲノムベースでの発現解析に比べ、信頼度が劣る（キメラや断片化した転写産物が多い）。
- ・重複遺伝子だけではなく、選択的スプライシングによるアイソフォームも区別する必要があるうえに、発現量によって遺伝子間のリードの厚みにバラつきがあり、アセンブルが困難。

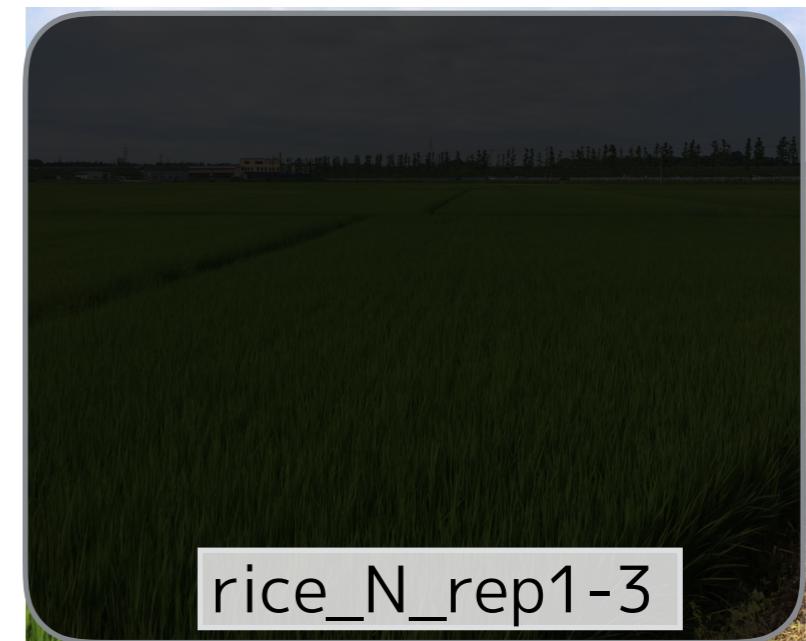
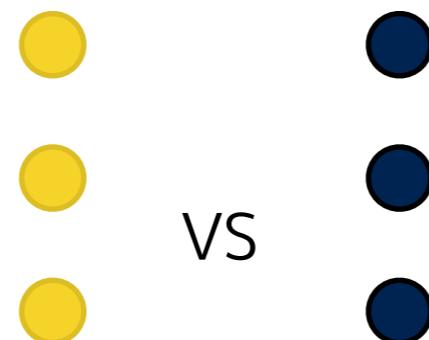
それでもゲノム配列などのない非モデル生物においては、トランскルiptオーム解析の主流になっている。

目的と使用するデータ

目的：de novo transcriptome assemble法を用いて、
昼と夜でのイネの葉の遺伝子発現の違いを調べる



12:00 PM



00:00 AM

- 田んぼで栽培する、イネ（コシヒカリ）の葉身のサンプル。
- 3つの異なる生育ステージ（**3反復**）において、昼（12時）と夜（0時）にサンプリング（**2条件**）。
- Illumina社の**Stranded mRNA-Seq法**でライブラリ調製。
- Illumina社のHiSeq2000による、**101bpのPaired-endシーケンシング**。

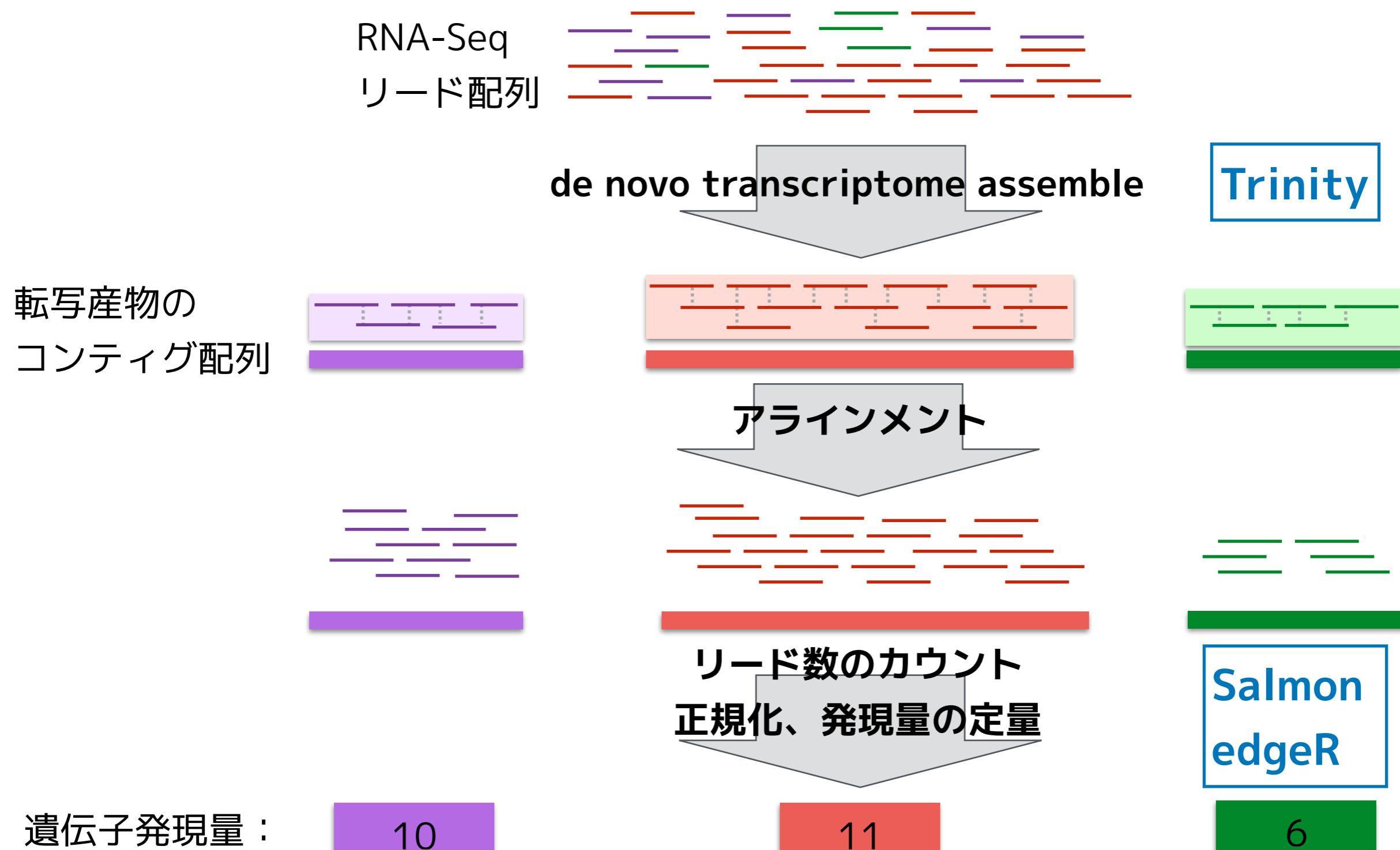
*未公開データ。全データでは解析に時間がかかるため、2番染色体の特定の領域（2Mbp）にアラインメントされるリードだけを事前に抽出しており、演習ではそれを利用する。

de novo transcriptome assemble法を用いた遺伝子発現解析の流れ



まず始めに、RNA-Seqリード配列から転写産物の全長配列を再構築する。

アセンブルされた転写産物配列上にRNA-Seqリードをアラインメントし、転写産物ごとにリード数をカウントすることにより遺伝子発現を定量する。



遺伝子発現量：

10

11

6

Step1. リード配列の準備

De novoトランск립トーム解析用ディレクトリ「denovo」に移動

```
$ cd ~/RNA-Seq/denovo
```

解析用ディレクトリ内のファイルを確認

```
$ ls
sample_info.txt      step2_Trinity.sh  step4_edgeR.sh
step1_prep_fastq.sh  step3_Salmon.sh   step5_eval_assembly.sh
```

シェルスクリプトが5つ用意されており、順番にシェルスクリプトを実行することで解析が進められるようになっている。

```
$ bash ./step1_prep_fastq.sh
$ bash ./step2_Trinity.sh
$ bash ./step3_Salmon.sh
$ bash ./step4_edgeR.sh
$ bash ./step5_eval_assembly.sh
```

演習では各ステップを説明しながら進めていくのでまだ実行しないでください！

Step1. リード配列の準備

Trinityに入力する配列データ（FASTQ）のヘッダーは、リード1の末尾が"/1"、リード2の末尾が"/2"にする必要がある。

ヘッダー末尾に"/1"や"/2"を加えるためのシェルスクリプト

```
$ less step1_prep_fastq.sh
```

```
- step1_prep_fastq.sh -
```

```
DataDir=$HOME/RNA-Seq_whole_genome/data
```

```
# Step1. Add suffix (/1 or /2) to original FASTQ header for Trinity
for DATASET in rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3
do
    echo "converting ${DATASET}_r1 ..."
    zcat $DataDir/${DATASET}_r1.org.fastq.gz | awk '{ if (NR%4==1) { print $1"/1" } else
{ print } }' | gzip -c > ${DATASET}_r1.trinity.fastq.gz
    echo "converting ${DATASET}_r2 ..."
    zcat $DataDir/${DATASET}_r2.org.fastq.gz | awk '{ if (NR%4==1) { print $1"/2" } else
{ print } }' | gzip -c > ${DATASET}_r2.trinity.fastq.gz
done
```

3つの処理から成り、真ん中のAWKスクリプトはオリジナルのFASTQを読み込み、ヘッダー行（行数を4で割って1余る行）の後ろに"/1"や"/2"を付け足している。

Step1. リード配列の準備

シェルスクリプトの実行 (実行時間: 約1分)

```
$ bash ./step1_prep_fastq.sh
```

作成されたリード配列の確認

```
$ ls -lh *.fastq.gz
-rw-rw-r-- 1 guest01 guest01 11M 9月 17 12:27 rice_D_rep1_r1.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 11M 9月 17 12:27 rice_D_rep1_r2.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 11M 9月 17 12:27 rice_D_rep2_r1.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 11M 9月 17 12:27 rice_D_rep2_r2.trinity.fastq.gz
...
-rw-rw-r-- 1 guest01 guest01 16M 9月 17 12:28 rice_N_rep2_r1.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 16M 9月 17 12:28 rice_N_rep2_r2.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 13M 9月 17 12:28 rice_N_rep3_r1.trinity.fastq.gz
-rw-rw-r-- 1 guest01 guest01 13M 9月 17 12:28 rice_N_rep3_r2.trinity.fastq.gz
```

```
$ less rice_D_rep1_r1.trinity.fastq.gz
@MG00HS14:530:C6M7YACXX:8:1101:1210:2214/1 ..... "/1"が付け足されている
GGCCGGAGCATCATCGTCCATTCCACCACCCATGTCGGCACAGCACCCTGGTACATCTTGGCAATAATTGGGTTGCACAGG
CCCTCCCGCTCCTTCATCT
+
@@@FDDADFFD?FGHHIGIIIEE4CGIIIIIFIIGGID;BCA9CGEHHFGCAC@DDDFFDACEEEEE=A@;?8<>C3?
BDBBDBB(5>B>>CCDCEDA
...
```

Step 2. Trinityによるde novo transcriptome assemble



TrinityによってRNA-Seqリードをアセンブルするシェルスクリプト

```
$ less step2_Trinity.sh
```

- step2_Trinity.sh -

```
ToolDir=$HOME/RNA-Seq_whole_genome/tool  
Trinity_bin=$ToolDir/trinityrnaseq-Trinity-v2.8.3  
Samtools_bin=$ToolDir/samtools-1.9/bin  
Jellyfish_bin=$ToolDir/jellyfish-2.2.10/bin  
Salmon_bin=$ToolDir/salmon-0.11.3-linux_x86_64/bin  
Bowtie2_bin=$ToolDir/bowtie2-2.3.4.2-linux-x86_64  
  
export PATH=$Trinity_bin:$Samtools_bin:$Jellyfish_bin:$Salmon_bin:$Bowtie2_bin:$PATH  
  
# Step2. De novo transcriptome assemble by Trinity  
Trinity --seqType fq --max_memory 4G --CPU 1 --SS_lib_type RF \  
--samples_file sample_info.txt --output Trinity_out
```

実行するのコマンドはTrinityひとつだが、内部で様々なツールを使用しており、パスの設定が必要。

入力ファイル等は sample_info.txt 内に記載することで指定
Stranded RNA-Seqライブラリのオプションを指定

Step 2. Trinityによるde novo transcriptome assemble



- sample_info.txt -

```
$ cat sample_info.txt
Day      Day_rep1          rice_D_rep1_r1.trinity.fastq.gz rice_D_rep1_r2.trinity.fastq.gz
Day      Day_rep2          rice_D_rep2_r1.trinity.fastq.gz rice_D_rep2_r2.trinity.fastq.gz
Day      Day_rep3          rice_D_rep3_r1.trinity.fastq.gz rice_D_rep3_r2.trinity.fastq.gz
Night    Night_rep1        rice_N_rep1_r1.trinity.fastq.gz rice_N_rep1_r2.trinity.fastq.gz
Night    Night_rep2        rice_N_rep2_r1.trinity.fastq.gz rice_N_rep2_r2.trinity.fastq.gz
Night    Night_rep3        rice_N_rep3_r1.trinity.fastq.gz rice_N_rep3_r2.trinity.fastq.gz
```

- サンプル条件、ラベル、配列ファイルをタブ区切りで記載。
- edgeRによる発現変動遺伝子の検出時にも同じファイルを利用する。
- 解析全体で1つのsample infoファイルを使い回すことで、サンプルの指定間違いなどの防止にもなる。

Step 2. Trinityによるde novo transcriptome assemble



- シェルスクリプトの実行（実行時間：約20分）

```
$ bash ./step2_Trinity.sh
```

アセンブルされた転写産物（コンティグ）配列の確認

```
$ ls Trinity_out/
```

Trinity.fasta	inchworm.K25.L25.fa.finished	pipeliner.134196.cmds
Trinity.fasta.gene_trans_map	inchworm.kmer_count	read_partitions
Trinity.timing	insilico_read_normalization	recursive_trinity.cmds
both.fa	jellyfish.kmers.fa	
recursive_trinity.cmds.completed		
...		

\$ less Trinity_out/Trinity.fasta

「TRINITY_DN8_c0_g1_i1」は転写産物配列ID。
「TRINITY_DN8_c0_g1遺伝子座のアイソフォーム1」
の配列という意味。

```
>TRINITY_DN11_c0_g1_i4 len=2370 path=[1:0-231 3:232-233 4:234-317 5:318-318 6:319-1595  
7:1596-1745 8:1746-2147 10:2148-2369]  
CCCCCTCTCCTCCTCCCTCCGCGCCGCCATCGCGTCGAGCTGCCCGCGAGATCCGCCGTTGACGGAGGAGGGAGTG  
AGTGAGTAGCAAGAGGAAGAGGAAGAGGATGGCGTCGGGACGGTACATGGCGTACTCGCCTCGCCCTCCACCACCCGCAC  
CCGGCCTCCGCGCCGCCTCCGCCGACCAAGGAGAAGTACCTCGCGGAGCTG...  
...
```

```
$ grep ">" Trinity_out/Trinity.fasta | wc -l
```

1045 アセンブルされた転写産物数

```
$ grep ">" Trinity_out/Trinity.fasta | cut -d "_" -f 1,2,3,4 | sort | uniq | wc -l
```

597 アセンブルされた遺伝子座数

Step 3. Salmonによる遺伝子発現量の定量



Salmonによって各Trinityコンティグの遺伝子発現量を定量するシェルスクリプト

```
$ less step3_Salmon.sh
```

- step3_Salmon.shの前半 -

```
ToolDir=$HOME/RNA-Seq_whole_genome/tool  
Trinity_bin=$ToolDir/trinityrnaseq-Trinity-v2.8.3  
Samtools_bin=$ToolDir/samtools-1.9/bin  
Salmon_bin=$ToolDir/salmon-0.11.3-linux_x86_64/bin  
R_bin=/work/NGSworkshop/tool/R-3.5.1/bin
```

```
export PATH=$Trinity_bin:$Samtools_bin:$Salmon_bin:$R_bin:$PATH
```

```
# Step3. Estimating Transcript Abundance by Salmon
```

```
$Trinity_bin/util/align_and_estimate_abundance.pl --transcripts Trinity_out/Trinity.fasta \  
--seqType fq --SS_lib_type RF --samples_file sample_info.txt \  
--est_method salmon \  
--thread_count 1 --trinity_mode --prep_reference
```

複数の発現量推定の手法が提供されて
いて、**--est_method**で指定可能

alignment_based: RSEM / eXpress
alignment_free: kallisto / **salmon**

Step 3. Salmonによる遺伝子発現量の定量

- step3_Salmon.shのつづき -

Salmonによってサンプルごとに得られた
遺伝子発現量をまとめ、正規化する。

```
# Build Transcript and Gene Expression Matrices
$Trinity_bin/util/abundance_estimates_to_matrix.pl \
--est_method salmon \
--gene_trans_map Trinity_out/Trinity.fasta.gene_trans_map \
--name_sample_by_basedir --out_prefix salmon \
Day_rep1/quant.sf Day_rep2/quant.sf Day_rep3/quant.sf \
Night_rep1/quant.sf Night_rep2/quant.sf Night_rep3/quant.sf
```

- --out_prefix trans_counts : 出力ファイルのprefix
- --est_method : 発現量の定量方法
- --name_sample_by_basedir : 出力結果のサンプル名をディレクトリ名にする
- 最後にSalmonの結果の4つのファイルをスペース区切りで指定

Step 3. Salmonによる遺伝子発現量の定量



シェルスクリプトの実行 (実行時間：約30秒)

```
$ bash ./step3_Salmon.sh
```

Salmonが出力する、サンプルごとの遺伝子発現量の確認

```
$ less Day_rep1/quant.sf
```

Name	Length	EffectiveLength	TPM	NumReads
...				
TRINITY_DN11_c0_g1_i4	2370	2205.908	17.918052	3.400
TRINITY_DN11_c0_g1_i5	1453	1288.908	3271.279807	362.674
TRINITY_DN11_c0_g1_i6	2222	2057.908	0.000000	0.000
TRINITY_DN11_c0_g1_i7	1831	1666.908	688.705191	98.747
TRINITY_DN11_c0_g1_i1	1568	1403.908	389.791814	47.070
TRINITY_DN11_c0_g1_i2	2731	2566.908	0.000000	0.000
...				

TPM: transcripts per kilobase million

Step 3. Salmonによる遺伝子発現量の定量

全サンプルの発現量データがまとめられたファイル
(リードカウントや正規化された発現量)

```
$ ls -lh salmon.*  
-rw-rw-r-- 1 guest01 guest01 38K 9月 17 12:56 salmon.gene.TMM.EXPR.matrix  
-rw-rw-r-- 1 guest01 guest01 42K 9月 17 12:56 salmon.gene.TPM.not_cross_norm  
-rw-rw-r-- 1 guest01 guest01 350 9月 17 12:56 salmon.gene.TPM.not_cross_norm.TMM_info.txt  
-rw-rw-r-- 1 guest01 guest01 522 9月 17 12:56 salmon.gene.TPM.not_cross_norm.runTMM.R  
-rw-rw-r-- 1 guest01 guest01 33K 9月 17 12:56 salmon.gene.counts.matrix  
-rw-rw-r-- 1 guest01 guest01 69K 9月 17 12:56 salmon.isoform.TMM.EXPR.matrix  
-rw-rw-r-- 1 guest01 guest01 74K 9月 17 12:56 salmon.isoform.TPM.not_cross_norm  
-rw-rw-r-- 1 guest01 guest01 350 9月 17 12:56 salmon.isoform.TPM.not_cross_norm.TMM_info.txt  
-rw-rw-r-- 1 guest01 guest01 528 9月 17 12:56 salmon.isoform.TPM.not_cross_norm.runTMM.R  
-rw-rw-r-- 1 guest01 guest01 64K 9月 17 12:56 salmon.isoform.counts.matrix
```

- *.TMM.EXPR.matrix: TMM正規化後のTPM（サンプル間の発現量の分布を補正済）
- *.TPM.not_cross_norm: TMM正規化前のTPM
- *.counts.matrix: リードカウント情報

Step 3. Salmonによる遺伝子発現量の定量

出力された遺伝子発現量データの確認

\$ less salmon.gene.counts.matrix 各コンティグごとのリードカウント数

	Day_rep1	Day_rep2	Day_rep3	Night_rep1	Night_rep2	Night_rep3
TRINITY_DN0_c0_g1	1074.74	1109.72	1177.57	453.39	765.04	465.04
TRINITY_DN0_c0_g2	0.00	11.59	12.91	0.00	11.31	13.18
TRINITY_DN0_c1_g1	3474.15	2265.54	2495.93	4309.03	4032.49	2973.35
TRINITY_DN0_c2_g1	941.11	1136.04	1562.60	196.39	395.61	397.52
TRINITY_DN100_c0_g1	41.01	48.15	22.86	52.64	47.26	62.23
TRINITY_DN100_c0_g2	166.14	153.42	173.19	231.18	251.42	199.13
TRINITY_DN101_c0_g1	14.54	11.67	3.19	25.39	31.85	39.26
TRINITY_DN101_c0_g2	258.58	62.45	95.17	278.95	102.73	125.59
...						

\$ less salmon.gene.TMM(EXPR.matrix) 各コンティグごとのTMM正規化後のTPM

	Day_rep1	Day_rep2	Day_rep3	Night_rep1	Night_rep2	Night_rep3
TRINITY_DN0_c0_g1	8650.120	8005.950	5938.349	3294.978	4060.982	2425.264
TRINITY_DN0_c0_g2	0.000	83.633	65.123	0.000	60.019	68.728
TRINITY_DN0_c1_g1	27961.943	16344.399	12586.696	31315.699	21405.140	15506.694
TRINITY_DN0_c2_g1	7574.590	8195.782	7880.017	1427.220	2099.986	2073.140
TRINITY_DN100_c0_g1	330.076	347.400	115.302	382.563	250.852	324.556
TRINITY_DN100_c0_g2	1337.210	1106.825	873.365	1680.070	1334.571	1038.488
TRINITY_DN101_c0_g1	117.038	84.203	16.100	184.548	169.041	204.728
TRINITY_DN101_c0_g2	2081.190	450.558	479.919	2027.285	545.291	654.959
...						

Step 4. edgeRによる遺伝子発現量の違いの統計検定



edgeRによる発現変動遺伝子を検出するシェルスクリプト

```
$ less step4_edgeR.sh
```

- step4_edgeR.sh -

```
ToolDir=$HOME/RNA-Seq_whole_genome/tool  
Trinity_bin=$ToolDir/trinityrnaseq-Trinity-v2.8.3  
R_bin=/work/NGSworkshop/tool/R-3.5.1/bin ..... RにはedgeRパッケージ  
export PATH=$Trinity_bin:$R_bin:$PATH  
  
# Step 4. Differential Expression Analysis by edgeR  
$Trinity_bin/Analysis/DifferentialExpression/run_DE_analysis.pl \  
--matrix salmon.gene.counts.matrix ..... マージしたカウントデータ  
--method edgeR ..... 手法を選択 (edgeR|DESeq2|voom|ROTS)  
--samples_file sample_info.txt ..... Trinityでも使ったサンプル情報ファイル  
--output edgeR_out ..... 出力先
```

Step 4. edgeRによる遺伝子発現量の違いの統計検定



シェルスクリプトの実行 (実行時間：約5秒)

```
$ bash ./step4_edgeR.sh
```

出力されたDEG解析結果の確認

```
$ less edgeR_out/salmon.gene.counts.matrix.Day_vs_Night.edgeR.DE_results
```

sampleA	sampleB	logFC	logCPM	PValue	FDR
TRINITY_DN20_c0_g1	Day	Night	-3.60910954106426	15.7385010545598	9.76517320797912e-25
TRINITY_DN17_c0_g1	Day	Night	-5.07277658132845	12.9935453393215	1.79369745794944e-18
TRINITY_DN74_c0_g1	Day	Night	2.90252919563421	15.7808172939919	4.74431833680181e-12
TRINITY_DN47_c0_g1	Day	Night	2.70995304895816	12.1237688054613	4.6989024282568e-11
TRINITY_DN123_c0_g3	Day	Night	-2.97773297559472	13.5583202256039	6.13559962140135e-10
TRINITY_DN0_c2_g1	Day	Night	2.06470113525029	12.2558805934621	4.12609113274095e-08
TRINITY_DN91_c0_g1	Day	Night	3.31030585045054	10.7238806061214	4.78398009276254e-08
...					

logFC:log2(fold change), logCPM:log2(counts per million), PValue:P-value, FDR:False Discovery Rate

昼と夜の遺伝子発現量の差の有意差が小さい順に、
転写産物や遺伝子座が出力されている。

De novo assembleの確かしさの検証



1. 元のリードをコンティグに再マッピングすることで、どれぐらいのリードがアセンブルに貢献しているかを見る。
2. コンティグ配列をクエリーとして、既知のタンパク質配列データベースに相同性検索をし、マップ率やカバー率によって、アセンブルの良し悪しを評価する。

上記、1、2を実行するためのシェルスクリプト (step5_eval_assembly.sh) を用意しているが、2番染色体の一部のみを使った演習データで行なっても結果の評価が難しいため、**ここでは全ranscriptomeを対象に解析した結果を示す。**

興味があれば演習用のデータで実行してみても構いません。

シェルスクリプトの実行 (実行時間：約10分)

```
$ bash ./step5_eval_assembly.sh
```

De novo assembleの確からしさの検証1

アセンブルに用いたRNA-Seqリードをコンティグに再マッピングし、どれくらいのリードがアセンブリに貢献しているかを調べる。

- step5_eval_assembly.sh の前半 -

```
ToolDir=$HOME/RNA-Seq/tool
Bowtie2_bin=$ToolDir/bowtie2-2.3.4.2-linux-x86_64
Samtools_bin=$ToolDir/samtools-1.9/bin
BLAST_bin=$ToolDir/ncbi-blast-2.7.1+/bin
Trinity_bin=$ToolDir/trinityrnaseq-Trinity-v2.8.3

export PATH=$Bowtie2_bin:$Samtools_bin:$BLAST_bin:$PATH

### Step5. Evaluation of de novo transcriptome assemblies by Trinity
# check read representation
bowtie2-build Trinity_out/Trinity.fasta Trinity.fasta ..... bowtie2のインデックス作成
bowtie2 -q --no-unal -k 20 -x Trinity.fasta \ ..... 以降はbowtie2によるアラインメント
-1
rice_D_rep1_r1.trinity.fastq.gz, rice_D_rep2_r1.trinity.fastq.gz, rice_D_rep3_r1.trinity.fastq.gz, rice_N_rep1_r1.trinity.fastq.gz, rice_N_rep2_r1.trinity.fastq.gz, rice_N_rep3_r1.trinity.fastq.gz \
-2
rice_D_rep1_r2.trinity.fastq.gz, rice_D_rep2_r2.trinity.fastq.gz, rice_D_rep3_r2.trinity.fastq.gz, rice_N_rep1_r2.trinity.fastq.gz, rice_N_rep2_r2.trinity.fastq.gz, rice_N_rep3_r2.trinity.fastq.gz \
| samtools view -Sb -o bowtie2_read_to_contig.bam
```

詳しくは「Assessing the Read Content of the Transcriptome Assembly」を参照

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/RNA-Seq-Read-Representation-by-Trinity-Assembly>

De novo assembleの確かしさの検証1

Bowtie2によるアラインメントの統計情報は標準エラー出力で確認できる。

```
120885050 reads; of these:  
 120885050 (100.00%) were paired; of these:  
   12576489 (10.40%) aligned concordantly 0 times  
   21693242 (17.95%) aligned concordantly exactly 1 time  
   86615319 (71.65%) aligned concordantly >1 times → 89.6%  
---  
 12576489 pairs aligned concordantly 0 times; of these:  
   1562524 (12.42%) aligned discordantly 1 time  
---  
 11013965 pairs aligned 0 times concordantly or discordantly; of these:  
 22027930 mates make up the pairs; of these:  
   5554496 (25.22%) aligned 0 times  
   739220 (3.36%) aligned exactly 1 time  
   15734214 (71.43%) aligned >1 times  
97.70% overall alignment rate
```

- aligned concordantly exactly 1 time
- aligned concordantly >1 times

を合わせたものが、70~80%になるのが一般的のようです。

De novo assembleの確からしさの検証2



コンティグ配列をクエリーとして、既知のタンパク質配列データベースに相同意検索をし、マップ率やカバー率によって、アセンブルの良し悪しを評価する。

- step5_eval_assembly.sh の後半 -

```
# homology search against rice protein sequences
ln -s ../data/rice_protein.fa
$BLAST_bin/makeblastdb -dbtype prot -in rice_protein.fa
$BLAST_bin/blastx -query Trinity_out/Trinity.fasta -db rice_protein.fa \
  -task blastx-fast -evalue 1e-20 -max_target_seqs 1 -outfmt 6 \
  -num_threads $CPU \
  -out blastx_trinity_to_rice_protein.outfmt6
```

```
perl $Trinity_bin/util/analyze_blastPlus_topHit_coverage.pl
blastx_trinity_to_rice_protein.outfmt6 Trinity_out/Trinity.fasta rice_protein.fa >
blastx_TopHitCoverage.txt
```

イネの全タンパク質配列 (FASTA) の
ファイルのシンボリックリンクを配置
し、BLAST用DBを作成

Trinityのコンティグを
クエリにして、イネの全
タンパク質配列に対して
BLASTX検索

Trinityに付属の
スクリプトで集計

詳しくは「Counting Full Length Trinity Transcripts」を参照

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/Counting-Full-Length-Trinity-Transcripts>

De novo assembleの確からしさの検証2

Trinityのコンティグ配列（148,683本）をクエリーとして、イネの全タンパク質配列（42,229本）に対してBLASTX検索し、イネの各タンパク質にベストヒットするコンティグ配列のCoverage率（タンパク質全長のうちTrinityコンティグとアラインメントがとれた領域の割合）を集計。

#hit_pct_cov_bin	count_in_bin	>bin_below
100	13692	13692
90	2417	16109
80	1536	17645
70	1219	18864
60	1132	19996
50	1120	21116
40	1058	22174
30	1068	23242
20	796	24038
10	134	24172

イネの全タンパク質（42,229本）のうちの約半数（21,116本）が、Coverage>50%という閾値でTrinityのコンティグに対応するものが存在している。

個別の遺伝子について調べてみる（LHY遺伝子の場合）

De novo assembleの確からしさの検証2で使用したスクリプトの結果

(blastx_trinity_to_rice_protein.outfmt6.w_pct_hit_length) を見ると、LHY (Os04t0583900-01)にヒットするコンティグが複数ある。

#qseqid	sseqid	pident	length	mismatch	gapopen	qstart	qend	sstart	send	evalue	bitscore
db_hit_len		pct_hit_len_aligned		hit_descr							
TRINITY_DN376_c0_g1_i5	Os04t0583900-01	99.578	237	1	0	1559	2269	228	464		
4.31e-154	461	237	51.08	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i3	Os04t0583900-01	99.578	237	1	0	1479	2189	228	464		
2.05e-154	461	237	51.08	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i6	Os04t0583900-01	99.578	237	1	0	1757	2467	228	464		
2.49e-153	461	237	51.08	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i1	Os04t0583900-01	99.721	359	1	0	1154	2230	106	464		
0.0	668	359	77.37	Similar to LHY protein.							
TRINITY_DN376_c0_g4_i1	Os04t0583900-01	95.385	65	3	0	204	10	28	92		
1.64e-40	137	65	14.01	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i2	Os04t0583900-01	99.578	237	1	0	1660	2370	228	464		
1.08e-153	461	237	51.08	Similar to LHY protein.							
TRINITY_DN376_c1_g1_i1	Os04t0583900-01	99.099	111	1	0	359	27	354	464		
8.75e-61	192	111	23.92	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i7	Os04t0583900-01	100.000	358	0	0	959	2032	107	464		
0.0	668	358	77.16	Similar to LHY protein.							
TRINITY_DN376_c0_g4_i2	Os04t0583900-01	93.333	45	3	0	144	10	48	92		
2.55e-23	90.9	45	9.70	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i8	Os04t0583900-01	100.000	464	0	0	561	1952	1	464		
0.0	894	464	100.00	Similar to LHY protein.							
TRINITY_DN376_c0_g1_i9	Os04t0583900-01	99.721	359	1	0	1057	2133	106	464		
0.0	668	359	77.37	Similar to LHY protein.							

個別の遺伝子について調べてみる（LHY遺伝子の場合）

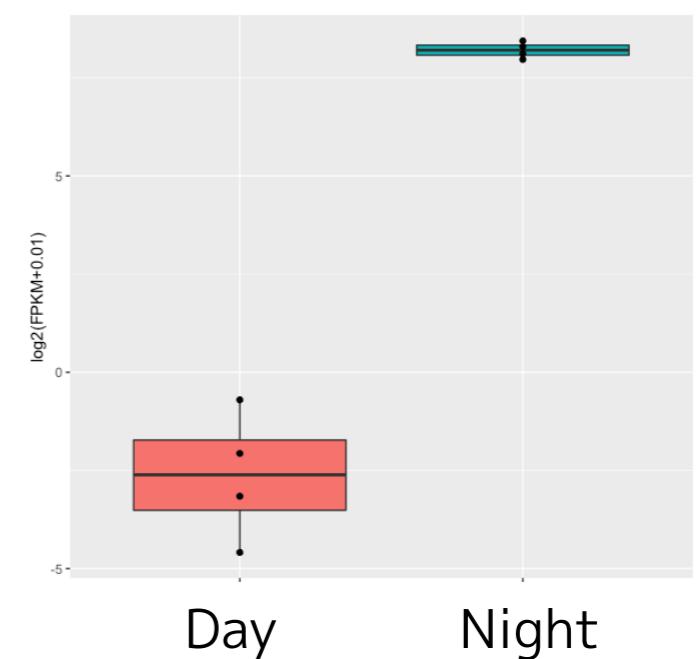
BLASTXの結果から、LHY転写産物（Os04t0583900-01）に最も相同意が高いTrinity contigは「TRINITY_DN376_c0_g1_i8」であることが分かる。

```
#qseqid sseqid pident length mismatch      gapopen qstart qend sstart send
evalue bitscore          db_hit_len   pct_hit_len_aligned hit_descr
TRINITY_DN376_c0_g1_i8 Os04t0583900-01 100.000 464       0        0      561    1952
1      464     0.0      894      464     100.00 Similar to LHY protein.
```

「TRINITY_DN376_c0_g1」遺伝子座について、edgeRの結果（`salmon.gene.counts.matrix.Day_vs_Night.edgeR.DE_results`）を調べてみると昼夜の遺伝子発現量に有意差がみられる（昼<夜）。

	sampleA	sampleB	logFC	logCPM	PValue	FDR
TRINITY_DN376_c0_g1	Day	Night	-9.488	7.318	2.119e-41	4.962e-37

昼にくらべて、夜の遺伝子発現量が有意に高いという結果になっており、リファレンス情報を用いたRNA-Seq解析の結果（右図）と一致する。

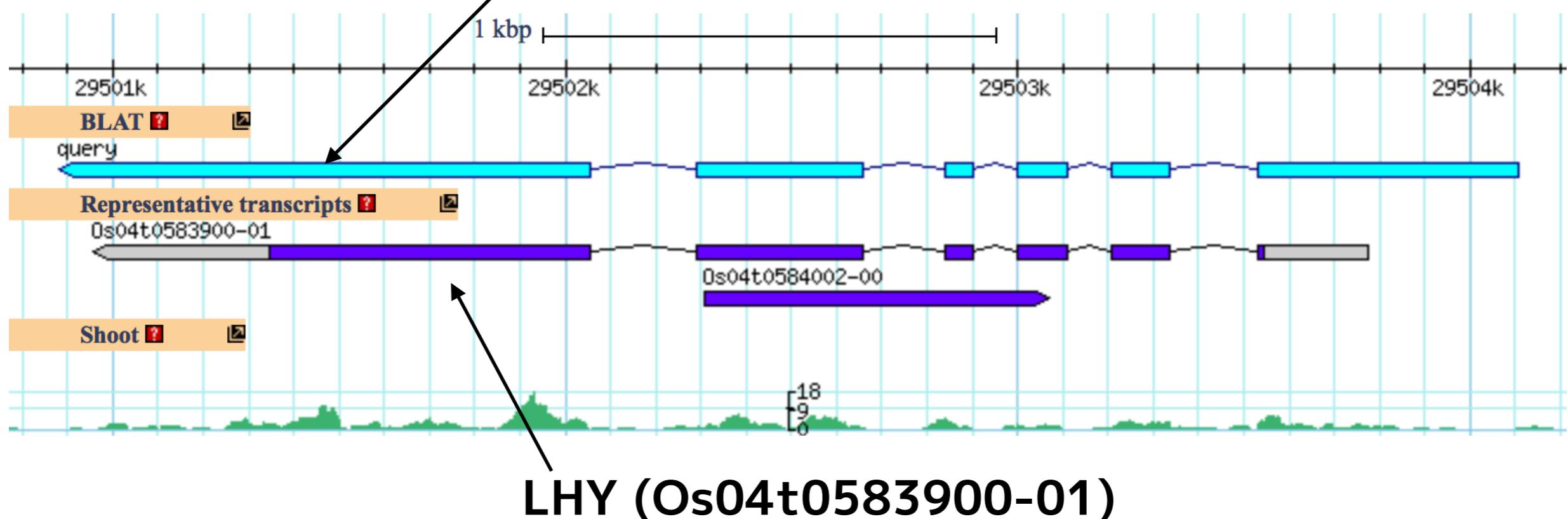


遺伝子全長がきちんとアセンブルされているか？

近縁種のゲノム配列や遺伝子アノテーションが整備されている場合、BLAT等でアラインメントし、マップ率や遺伝子構造を比較することでも評価が可能です。

RAP-DBのBLAT検索によって
イネゲノムにマッピングされたコンティグ配列

TRINITY_DN376_c0_g1_i8 (2,417 bp)



LHY (Os04t0583900-01)

コンティグ配列のアノテーション



得られた転写産物が何者なのかが知りたい！
そのためにはコンティグ配列のアノテーションが必要

TransDecoder (Find Coding Regions Within Transcripts)

TransDecoder identifies candidate coding regions within transcript sequences, such as those generated by de novo RNA-Seq transcript assembly using Trinity, or constructed based on RNA-Seq alignments to the genome using Tophat and Cufflinks.

TransDecoder identifies likely coding sequences based on the following criteria:

- a minimum length open reading frame (ORF) is found in a transcript sequence
- a log-likelihood score similar to what is computed by the GeneID software is > 0 .
- the above coding score is greatest when the ORF is scored in the 1st reading frame as compared to scores in the other 5 reading frames.
- if a candidate ORF is found fully encapsulated by the coordinates of another candidate ORF, the longer one is reported. However, a single transcript can report multiple ORFs (allowing for operons, chimeras, etc).
- **optional** the putative peptide has a match to a Pfam domain above the noise cutoff score.

The software is primarily maintained by [Brian Haas](#) at the [Broad Institute](#) and [Alexie Papanicolaou](#) at the [Commonwealth Scientific and Industrial Research Organisation](#) (CSIRO). It is integrated into other related software such as [Trinity](#), [PASA](#), [EvidenceModeler](#), and [Trinotate](#).

<http://transdecoder.github.io/>

転写産物の塩基配列からタンパク質コード領域（CDS）を予測

Trinotate: Transcriptome Functional Annotation and Analysis

Trinotate



RNA-Seq → Trinity → Transcripts/Proteins → Functional Data → Discovery

Automated Higher Order Biological Analysis

<https://trinotate.github.io/>

- 機能が分かっているホモログ
 - 機能ドメイン
 - 機能分類 (Gene Ontology)
-) を予測

- ▶ リファレンス情報を用いたRNA-Seq解析
- ▶ De novoトランскriプトーム解析
- ▶ **RNA-Seqデータを用いた多型検出**

RNA-Seqデータを用いた多型検出の特徴

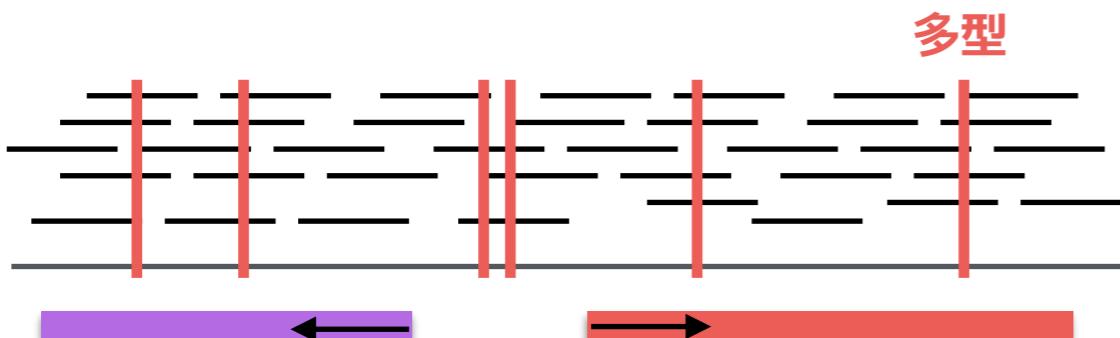
良い点

- ・ 巨大なゲノムでも効率よく多型情報が得られる。
- ・ 遺伝子発現量と多型情報が同時に得られる。
- ・ 転写領域は進化的に保存されており、進化的に距離の離れた種の比較でも効率よく多型情報が得られる。

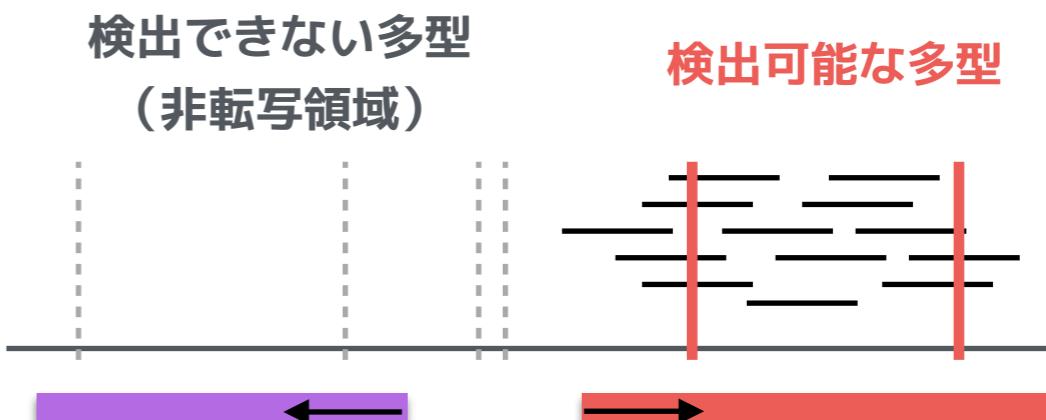
悪い点

- ・ 低発現、または発現していない遺伝子の多型情報は得られない。
- ・ 遺伝子発現解析と同時にできるが、転写開始点上流の発現調節領域の多型情報は得られない。

ゲノムリシーケンスデータ



RNA-Seqデータ



目的：RNA-Seqデータを用いて日本晴とコシヒカリの間のゲノム配列の違い（多型）を調べる

日本晴
(リファレンスゲノム)

VS



コシヒカリ (RNA-Seq)



12:00 PM

rice_N_rep1-3

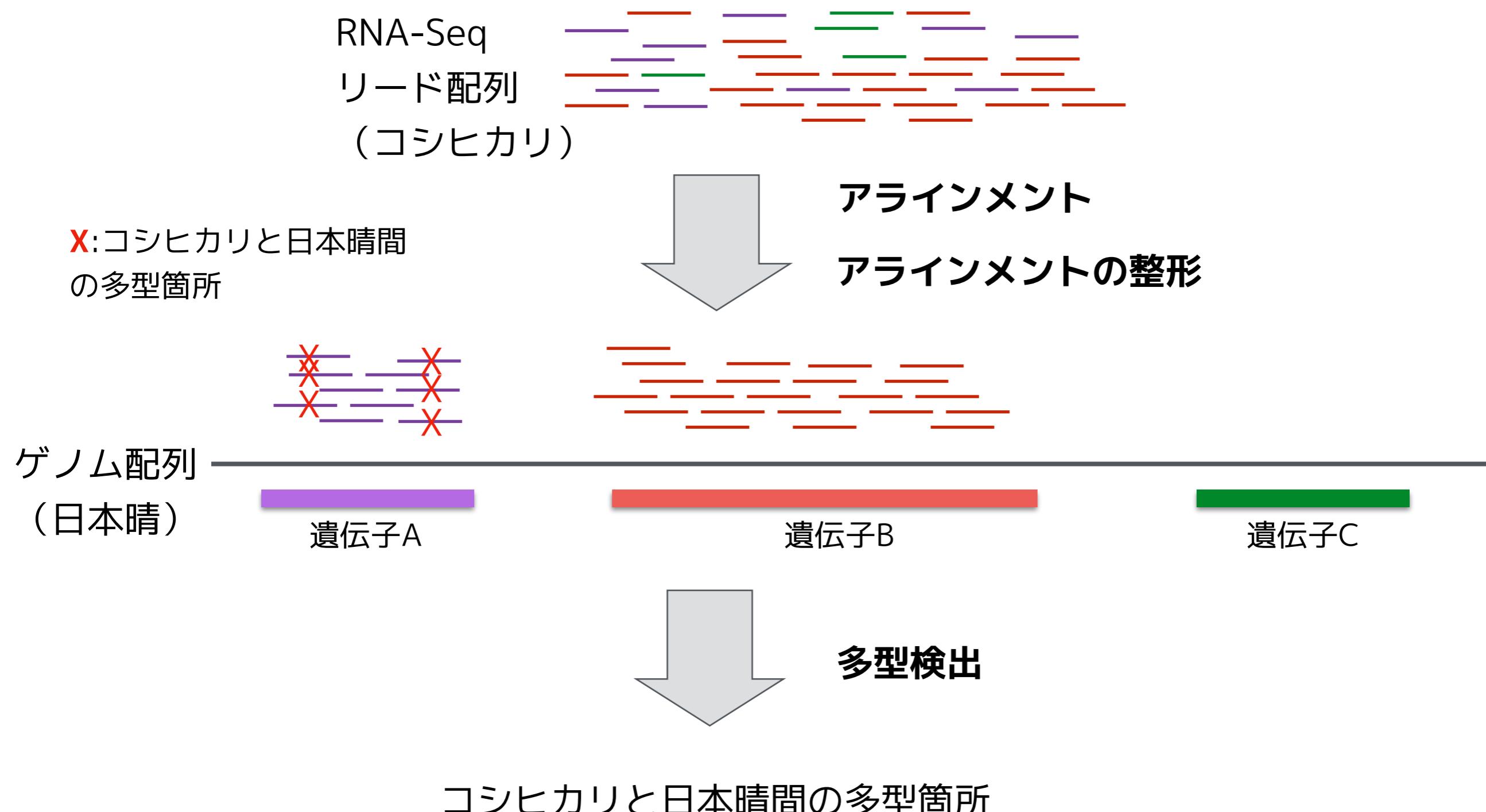
00:00 AM

- 田んぼで栽培する、イネ（コシヒカリ）の葉身のサンプル。
- 3つの異なる生育ステージ（3反復）において、**昼（12時）**と**夜（0時）**にサンプリング（2条件）。
- Illumina社の**Stranded mRNA-Seq**法でライブラリ調製。
- Illumina社のHiSeq2000による、**101bpのPaired-endシーケンシング**。

*全て未公開データ。全データでは解析に時間がかかるため、2番染色体の特定の領域（2Mbp）にアラインメントされるリードだけを事前に抽出しており、演習ではそれを利用する。

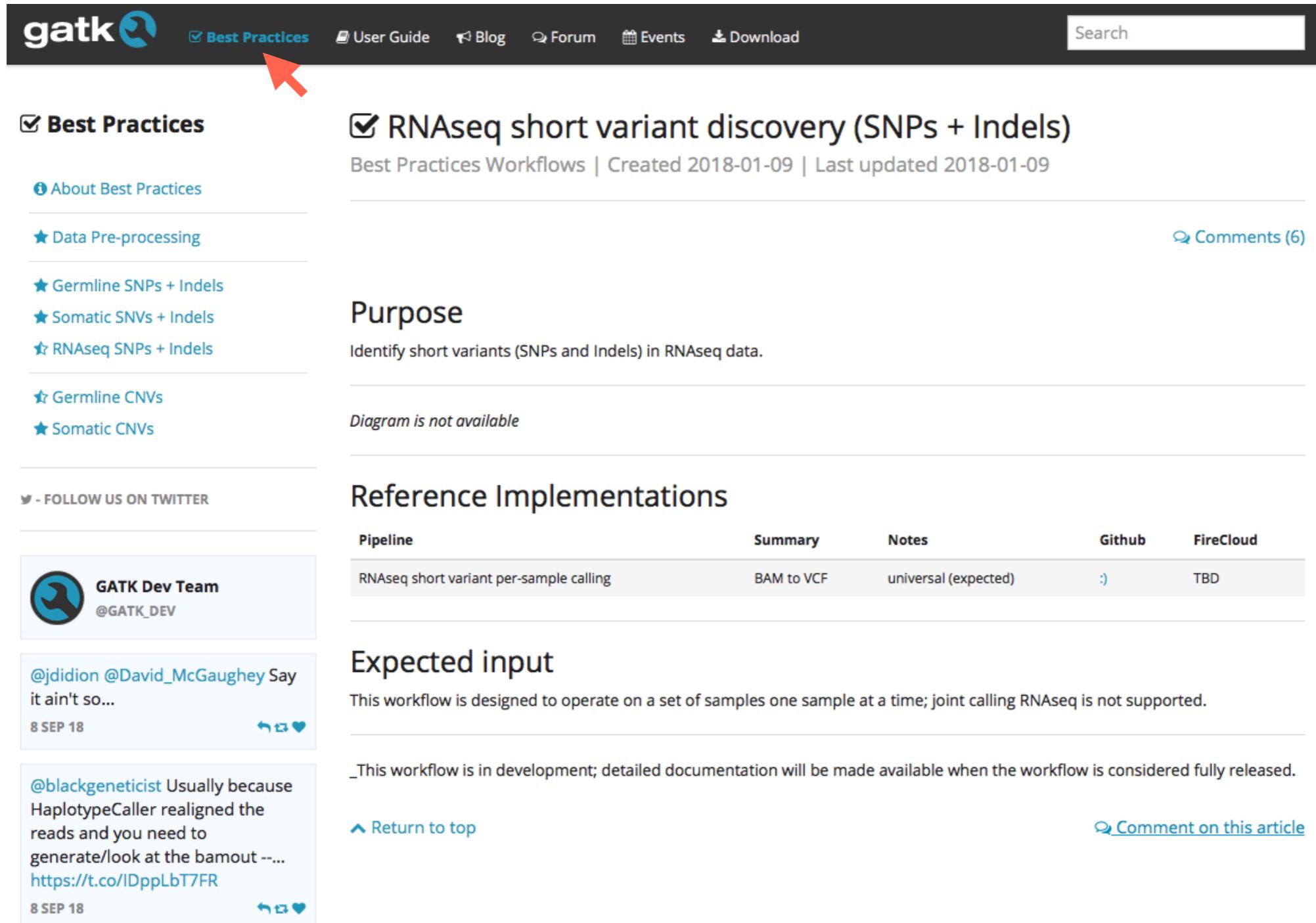
RNA-Seqデータを用いた多型検出の流れ

リファレンスとは異なる種や品種由来のRNA-Seqリードをゲノム配列にアラインメント（マッピング）し、種間や品種間の遺伝子配列の違い（多型情報）を得る。



GATKのウェブサイトのBest Practices

<https://software.broadinstitute.org/gatk/best-practices/>



The screenshot shows the GATK Best Practices website. A red arrow points to the "Best Practices" link in the top navigation bar. The main content area displays the "RNAseq short variant discovery (SNPs + Indels)" workflow, which was created on 2018-01-09 and last updated on 2018-01-09. It includes sections for Purpose, Reference Implementations, and Expected input. The Reference Implementations table has columns for Pipeline, Summary, Notes, Github, and FireCloud. The Pipeline row lists "RNAseq short variant per-sample calling", "BAM to VCF", "universal (expected)", and links for Github and FireCloud. The Expected input section notes that the workflow is designed for one sample at a time and is in development. There are also sections for "About Best Practices" and "Follow Us on Twitter".

Pipeline	Summary	Notes	Github	FireCloud
RNAseq short variant per-sample calling	BAM to VCF	universal (expected)	Github	TBD

最新版のGATK4系のワークフローは開発、検証中・・・

GATK3系版のRNA-Seqデータによる多型検出方法をアレンジ

<https://software.broadinstitute.org/gatk/documentation/article?id=4067>

gatk Best Practices User Guide Blog Forum Events Download Search

User Guide GATK 3 ▾

- Quick Start Guide
- Best Practices
- Tool Documentation
- Methods and Algorithms**
- Tutorials
- Dictionary
- Presentations
- Frequently Asked Questions
- Solutions to Problems
- Bugs & Feature Requests
- Version History
- Pipelining Options
- GATK on FireCloud

- FOLLOW US ON TWITTER

GATK Dev Team @GATK_DEV

Best Practices for Variant Discovery in RNAseq

Methods and Algorithms | Created 2014-04-16 | Last updated 2015-05-16

This article is part of the Best Practices documentation. See <http://www.broadinstitute.org/gatk/guide/best-practices-for-the-full-documentation-set>.

This is our recommended workflow for calling variants in RNAseq data from single samples, in which all steps are performed per-sample. In future we will provide cohort analysis recommendations, but these are not yet available.

```

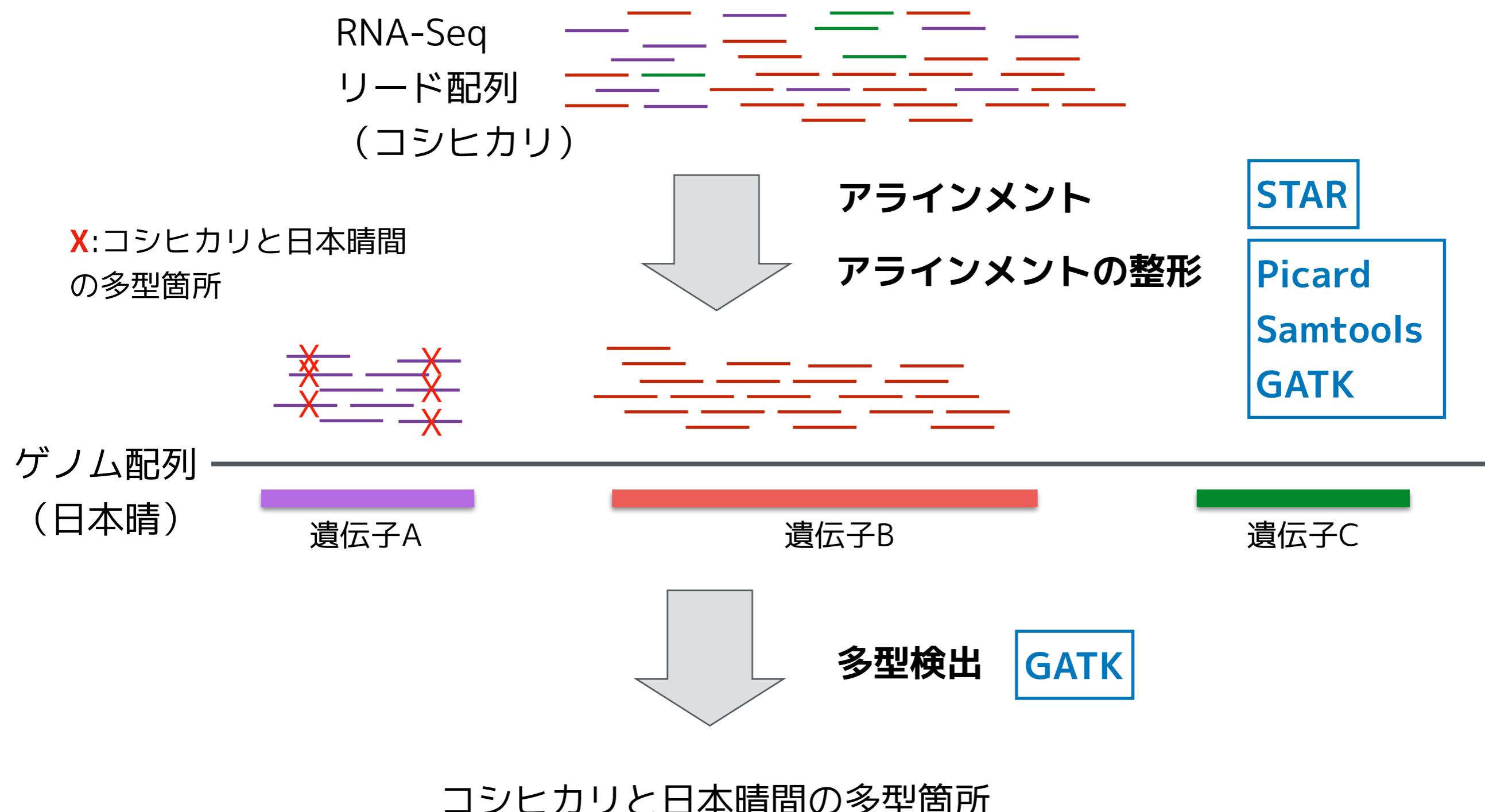
graph LR
    subgraph DATA_CLEANUP [DATA CLEANUP]
        A[Raw RNAseq Reads] --> B[Map to Reference  
STAR 2-pass]
        B --> C[Mark Duplicates & Sort (Picard)]
        C --> D[Split'N'Trim  
+ReassignMappingQuality]
        D --> E[Indel Realignment]
        E --> F[Base Recalibration]
        F --> G[Analysis -Ready RNAseq Reads]
    end
    subgraph VARIANT_DISCOVERY [VARIANT DISCOVERY]
        G --> H[Variant Calling  
HC in RNAseq mode]
        H --> I[Raw Variants  
SNPs  
Indels]
        I --> J[Variant Filtering  
RNAseq-specific settings]
        J --> K[Filtered Variants  
SNPs  
Indels]
    end
    subgraph EVALUATION [EVALUATION]
        K --> L[Variant Annotation]
        L --> M[Phasing]
        M --> N[Variant Evaluation  
look good?]
        N --> O[troubleshoot]
        N --> P[use in project]
    end

```

GATK3系版のワークフローを参考にして考えた、
GATK4を用いて多型検出する方法を紹介します。

RNA-Seqデータを用いた多型検出の流れ

リファレンスとは異なる種や品種由来のRNA-Seqリードをゲノム配列にアラインメント（マッピング）し、種間や品種間の遺伝子配列の違い（多型情報）を得る。



Step1. STARのためのインデックス作成

RNA-Seqデータを用いた多型検出用ディレクトリ「varcall」に移動

```
$ cd ~/RNA-Seq/varcall
```

解析用ディレクトリ内のファイルを確認

```
$ ls
step1_make_STAR_index.sh      step3_BAM_Processing.sh
step2_STAR.sh                  step4_GATK.sh
```

シェルスクリプトが5つ用意されており、順番にシェルスクリプトを実行することで解析が進められるようになっている。

```
$ bash ./step1_make_STAR_index.sh
$ bash ./step2_STAR.sh
$ bash ./step3_BAM_Processing.sh
$ bash ./step4_GATK.sh
```

演習では各ステップを説明しながら進めていくのでまだ実行しないでください！

Step1. STARのためのインデックス作成

STAR用のゲノム配列のインデックスを作成するシェルスクリプト

```
$ less step1_make_STAR_index.sh
```

- step1_make_STAR_index.sh -

```
DataDir=$HOME/RNA-Seq_whole_genome/data  
ToolDir=$HOME/RNA-Seq_whole_genome/tool  
STAR_bin=$ToolDir/STAR-2.6.1b/bin/Linux_x86_64
```

```
export PATH=$STAR_bin:$PATH
```

```
### Step1. Make genome index for STAR
```

```
GenomeDir=STAR_index  
mkdir $GenomeDir
```

```
STAR --runMode genomeGenerate \  
--genomeFastaFiles $DataDir/genome.fa \  
--genomeDir $GenomeDir \  
--sjdbGTFfile $DataDir/annotation.gtf \  
--sjdbOverhang 100 --runThreadN 1
```

..... インデックス作成モードを指定してSTARを実行
..... ゲノム配列 (FASTA) の指定
..... インデックス出力ディレクトリの指定
..... 遺伝子アノテーション (GTF) の指定
..... sjdbOverhangは (リード長-1) を指定

パス等の設定

インデックスの出力

ディレクトリの作成

Step1. STARのためのインデックス作成

シェルスクリプトの実行 (実行時間：約1分)

```
$ bash ./step1_make_STAR_index.sh
```

STAR_indexディレクトリ中に作成されたインデックスファイルの確認

```
$ ls -lh STAR_index/
合計 1.8G
-rw-rw-r-- 1 guest01 guest01 35M 9月 17 12:14 Genome
-rw-rw-r-- 1 guest01 guest01 285M 9月 17 12:14 SA
-rw-rw-r-- 1 guest01 guest01 1.5G 9月 17 12:14 SAindex
-rw-rw-r-- 1 guest01 guest01 9 9月 17 12:13 chrLength.txt
-rw-rw-r-- 1 guest01 guest01 6 9月 17 12:13 chrName.txt
-rw-rw-r-- 1 guest01 guest01 15 9月 17 12:13 chrNameLength.txt
-rw-rw-r-- 1 guest01 guest01 11 9月 17 12:13 chrStart.txt
-rw-rw-r-- 1 guest01 guest01 41K 9月 17 12:14 exonGeTrInfo.tab
-rw-rw-r-- 1 guest01 guest01 20K 9月 17 12:14 exonInfo.tab
-rw-rw-r-- 1 guest01 guest01 3.8K 9月 17 12:14 geneInfo.tab
-rw-rw-r-- 1 guest01 guest01 644 9月 17 12:14 genomeParameters.txt
-rw-rw-r-- 1 guest01 guest01 26K 9月 17 12:14 sjdbInfo.txt
-rw-rw-r-- 1 guest01 guest01 26K 9月 17 12:14 sjdbList.fromGTF.out.tab
-rw-rw-r-- 1 guest01 guest01 26K 9月 17 12:14 sjdbList.out.tab
-rw-rw-r-- 1 guest01 guest01 18K 9月 17 12:14 transcriptInfo.tab
```

Step2. STARによるRNA-Seqリードのアラインメント



STARによってRNA-Seqリードをゲノム配列にアラインメントするシェルスクリプト

```
$ less step2_STAR.sh
```

```
- step2_STAR.sh -
```

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
STAR_bin=$ToolDir/STAR-2.6.1b/bin/Linux_x86_64
```

```
export PATH=$STAR_bin:$PATH
```

```
### Step2. Alignment of RNA-Seq reads by STAR
```

```
GenomeDir=STAR_index
```

```
Read1_list=$DataDir/rice_D_rep1_r1.org.fastq.gz,$DataDir/  
rice_D_rep2_r1.org.fastq.gz,$DataDir/rice_D_rep3_r1.org.fastq.gz,$DataDir/  
rice_N_rep1_r1.org.fastq.gz,$DataDir/rice_N_rep2_r1.org.fastq.gz,$DataDir/  
rice_D_rep3_r1.org.fastq.gz
```

```
Read2_list=$DataDir/rice_D_rep1_r2.org.fastq.gz,$DataDir/  
rice_D_rep2_r2.org.fastq.gz,$DataDir/rice_D_rep3_r2.org.fastq.gz,$DataDir/  
rice_N_rep1_r2.org.fastq.gz,$DataDir/rice_N_rep2_r2.org.fastq.gz,$DataDir/  
rice_D_rep3_r2.org.fastq.gz
```

```
STAR --genomeDir $GenomeDir --readFilesCommand "gunzip -c" \  
--readFilesIn $Read1_list $Read2_list \  
--alignIntronMin 20 --alignIntronMax 10000 \  
--outSAMtype BAM SortedByCoordinate \  
--runThreadN 1
```

パス等の設定

全サンプルをまとめて
変数に格納

リードファイルの圧縮形式に応じて指定

STARによる
アラインメントを実行

Step2. STARによるRNA-Seqリードのアラインメント



シェルスクリプトの実行 (実行時間：約1分)

```
$ bash ./step2_STAR.sh
```

作成されたアラインメントファイルの確認

```
$ ls -lhtr
合計 133M
...
-rw-rw-r-- 1 guest01 guest01 133M 9月 17 12:17 Aligned.sortedByCoord.out.bam
-rw-rw-r-- 1 guest01 guest01 59K 9月 17 12:17 SJ.out.tab
-rw-rw-r-- 1 guest01 guest01 246 9月 17 12:17 Log.progress.out
-rw-rw-r-- 1 guest01 guest01 25K 9月 17 12:17 Log.out
-rw-rw-r-- 1 guest01 guest01 1.9K 9月 17 12:17 Log.final.out
```

すべてのサンプルをまとめて1サンプルとしてアラインメントしたので、1つの BAMファイル（Aligned.sortedByCoord.out.bam）が出力されている。

Step3. アライメントデータの加工

PicardやGATKでアライメントデータを加工し、下流の解析に適したものにするシェルスクリプト。

```
$ less step3_BAM_Processing.sh
```

- step3_BAM_Processing.sh の前半 -

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
Picard_bin=$ToolDir/picard-2.18.12  
Samtools_bin=$ToolDir/samtools-1.9  
GATK_bin=$ToolDir/gatk-4.0.8.1  
  
Export PATH=$Samtools_bin:$GATK_bin:$PATH  
  
### Step3. Processing of alignment (BAM) file  
java -jar $Picard_bin/picard.jar AddOrReplaceReadGroups \  
I=Aligned.sortedByCoord.out.bam \  
O=Aligned.sortedByCoord.RG.out.bam \  
SO=coordinate \  
RGID=Koshihikari RGLB=TruSeq_RNA_stranded RGPL=illumina RGPU=HiSeq2000  
RGSM=Koshihikari
```

パス等の設定

アライメントデータにRG (read group) 情報を追加した上で、位置でソートする。

Step3. アライメントデータの加工

- step3_BAM_Processing.shのつづき -

```
java -jar $Picard_bin/picard.jar MarkDuplicates \
I=Aligned.sortedByCoord.RG.out.bam \
O=Aligned.sortedByCoord.RG.MD.out.bam \
CREATE_INDEX=true \
VALIDATION_STRINGENCY=SILENT \
M=Aligned.sortedByCoord.RG.MD.metrics
```

PCRによって重複した（冗長になった）リードを取り除く

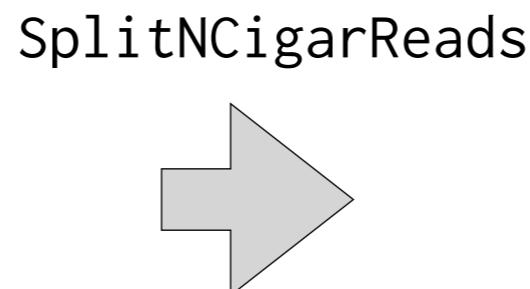
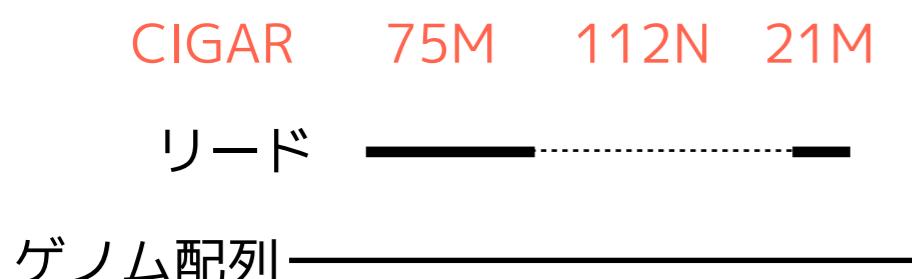
```
samtools faidx $DataDir/genome.fa
```

```
java -jar $Picard_bin/picard.jar CreateSequenceDictionary \
R=$DataDir/genome.fa \
O=$DataDir/genome.dict
```

ゲノム配列のindexやdictionaryの作成

```
gatk SplitNCigarReads \
-R $DataDir/genome.fa \
-I Aligned.sortedByCoord.RG.MD.out.bam \
-O Aligned.sortedByCoord.RG.MD.SplitN.out.bam
```

アライメント中のイントロン領域情報を除く



Step3. アラインメントデータの加工

シェルスクリプトの実行 (実行時間：約2分)

```
$ bash ./step3_BAM_Processing.sh
```

作成されたアラインメントファイルの確認

```
$ ls -lhtr
合計 562M
...
-rw-rw-r-- 1 guest01 guest01 118M 9月 17 12:18 Aligned.sortedByCoord.RG.out.bam
-rw-rw-r-- 1 guest01 guest01 120M 9月 17 12:19 Aligned.sortedByCoord.RG.MD.out.bam
-rw-rw-r-- 1 guest01 guest01 23K 9月 17 12:19 Aligned.sortedByCoord.RG.MD.out.bai
-rw-rw-r-- 1 guest01 guest01 2.8K 9月 17 12:19 Aligned.sortedByCoord.RG.MD.metrics
-rw-rw-r-- 1 guest01 guest01 193M 9月 17 12:20 Aligned.sortedByCoord.RG.MD.SplitN.out.bam
-rw-rw-r-- 1 guest01 guest01 22K 9月 17 12:20 Aligned.sortedByCoord.RG.MD.SplitN.out.bai
```

各ステップの出力ファイルが複数存在するが、最終アラインメントは「Aligned.sortedByCoord.RG.MD.SplitN.out.bam」である。

Step4. GATK HaplotypeCallerによる多型検出

GATK HaplotypeCallerによってRNA-Seqリードのアラインメントを元に多型(SNP、InDel)を検出するシェルスクリプト

```
$ less step4_GATK.sh
```

```
- step4_GATK.sh -
```

```
DataDir=$HOME/RNA-Seq/data  
ToolDir=$HOME/RNA-Seq/tool  
GATK_bin=$ToolDir/gatk-4.0.8.1
```

```
export PATH=$GATK_bin:$PATH
```

```
### Step4. Detection of variations by GATK HaplotypeCaller
```

```
gatk HaplotypeCaller -R $DataDir/genome.fa -I  
Aligned.sortedByCoord.RG.MD.SplitN.out.bam \  
-stand-call-conf 20 --dont-use-soft-clipped-bases \  
-O variation.vcf.gz
```

HaplotypeCallerによる多型検出

```
gatk VariantFiltration -R $DataDir/genome.fa -V variation.vcf.gz \  
-window 20 -cluster 3 \  
--filter-name "QD" --filter "QD < 2.0" \  
--filter-name "FS" --filter "FS > 60.0" \  
-O variation.filter.vcf.gz
```

複数の条件での多型のフィルタリング

フィルタリングで指定しているQDやFSについては以下のページが参考になる

<https://software.broadinstitute.org/gatk/documentation/article?id=11069>

Step4. GATK HaplotypeCallerによる多型検出

シェルスクリプトの実行 (実行時間：約2分)

```
$ bash ./step4_GATK.sh
```

作成された多型情報ファイルの確認

```
$ ls -lhtr
合計 562M
...
-rw-rw-r-- 1 guest01 guest01 11K 9月 17 12:23 variation.vcf.gz
-rw-rw-r-- 1 guest01 guest01 839 9月 17 12:23 variation.vcf.gz.tbi
-rw-rw-r-- 1 guest01 guest01 11K 9月 17 12:24 variation.filter.vcf.gz
-rw-rw-r-- 1 guest01 guest01 840 9月 17 12:24 variation.filter.vcf.gz.tbi
```

基本的にはゲノムリシーケンスデータによる変異検出と同じで、多型情報はVCFファイル(gz圧縮)として出力されている。

検出された多型情報 (variation.filter.vcf.gz) の確認



VCF形式で記載された多型情報

```
> less variation.filter.vcf.gz
...
1 chr02 30473056 . T A 419.77 PASS
AC=1;AF=0.500;AN=2;BaseQRankSum=-0.312;DP=210;ExcessHet=3.0103;FS=16.873;MLEAC=1;MLEAF=0.
500;MQ=60.00;MQRankSum=0.000;QD=2.00;ReadPosRankSum=-6.091;SOR=2.491 GT:AD:DP:GQ:PL
0/1:173,37:210:99:448,0,9241
2 chr02 30473059 . T A 53.77 QD
AC=1;AF=0.500;AN=2;BaseQRankSum=1.180;DP=164;ExcessHet=3.0103;FS=13.389;MLEAC=1;MLEAF=0.5
00;MQ=60.00;MQRankSum=0.000;QD=0.33;ReadPosRankSum=-5.911;SOR=3.304 GT:AD:DP:GQ:PL
0/1:147,17:164:82:82,0,9231
3 chr02 30473609 . G A 1050.77 PASS
AC=2;AF=1.00;AN=2;DP=31;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=33.90;SO
R=0.756 GT:AD:DP:GQ:PL 1/1:0,31:31:93:1079,93,0
4 chr02 30474736 . T C 7195.77 PASS
AC=2;AF=1.00;AN=2;DP=242;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=29.73;S
OR=0.815 GT:AD:DP:GQ:PL 1/1:0,242:242:99:7224,726,0
...
```

PASS: フィルタリングをパスした多型

IGVでアラインメントと多型情報の可視化

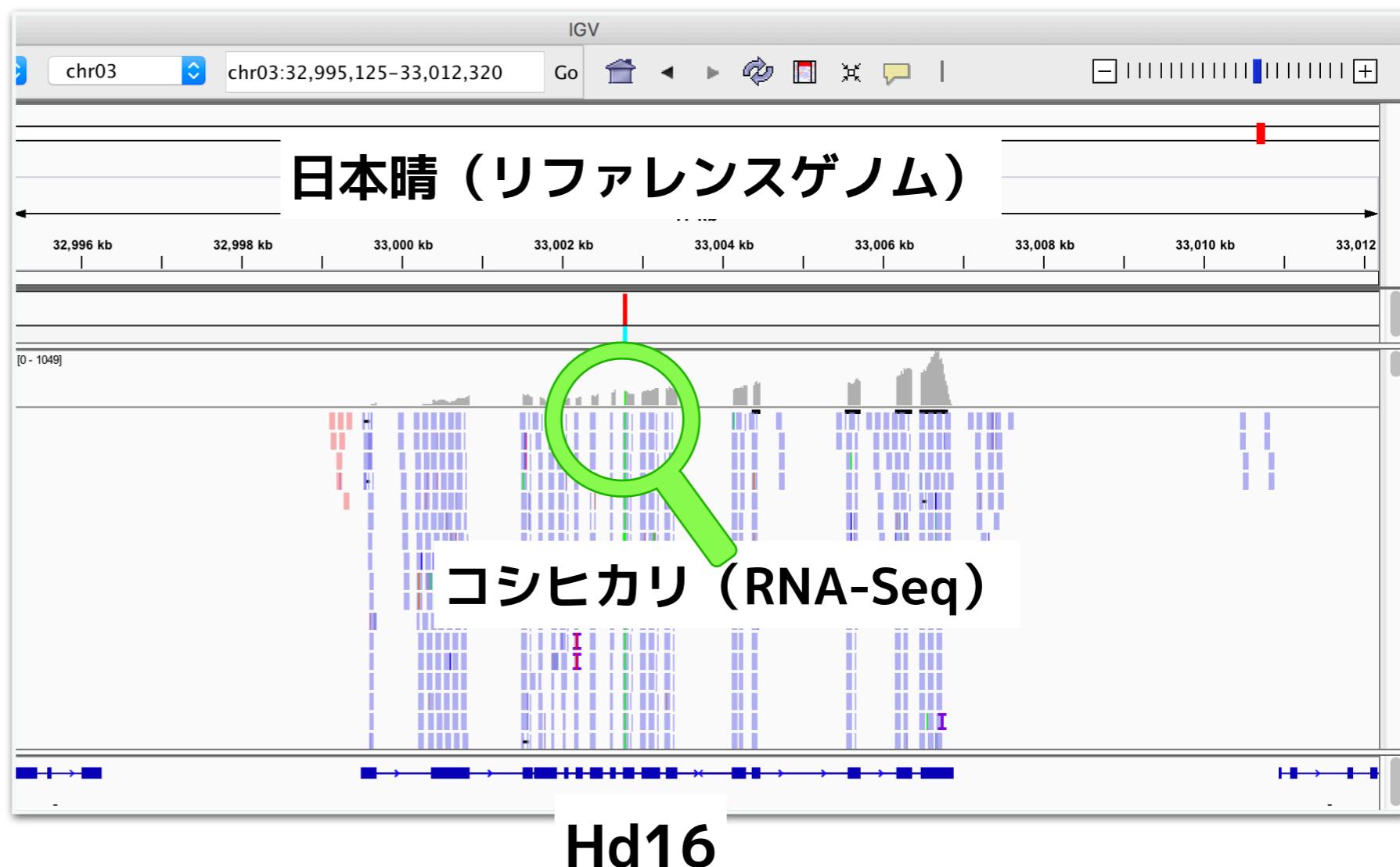
アラインメント情報 (Aligned.sortedByCoord.RG.MD.SplitN.out.bam) と 多型情報 (variation.filter.vcf.gz) は IGV で閲覧することができる。



↓ : 前のスライドで示したVCFファイル中の4つの多型

既知の品種間多型は検出されているか？

Hd16遺伝子上にある、日本晴とコシヒカリの開花期の違いに関わる既知のFNP*



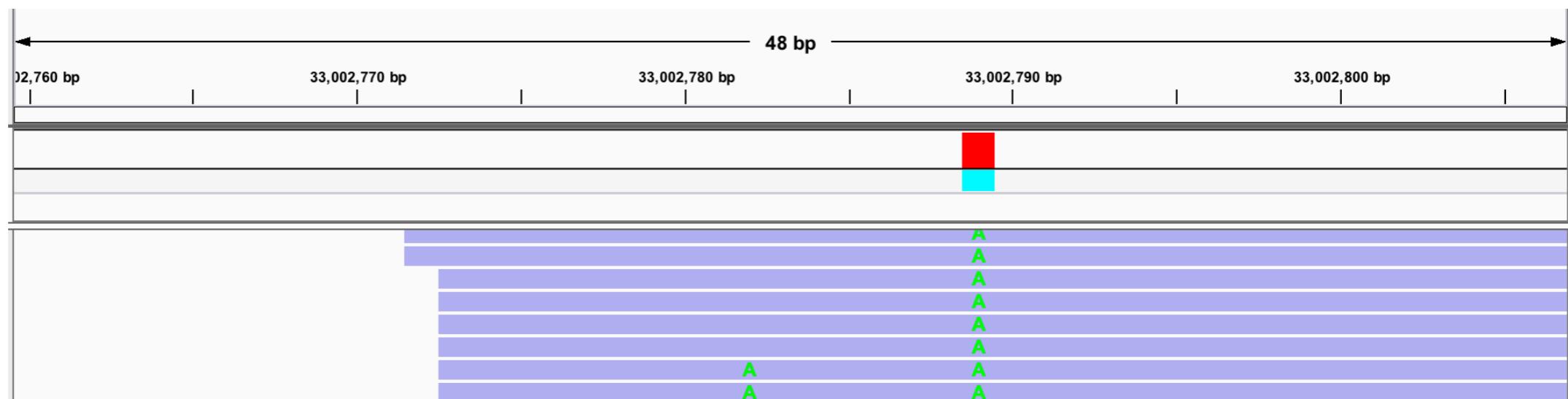
*FNP=Functional Nucleotide Polymorphism

ローカルPC上(workshop/RNA-Seq/varcall)にゲノムワイドな解析結果から3番染色体の情報を抜き出したアラインメントとVCFファイルがあるのでIGVで開いて見てみよう。

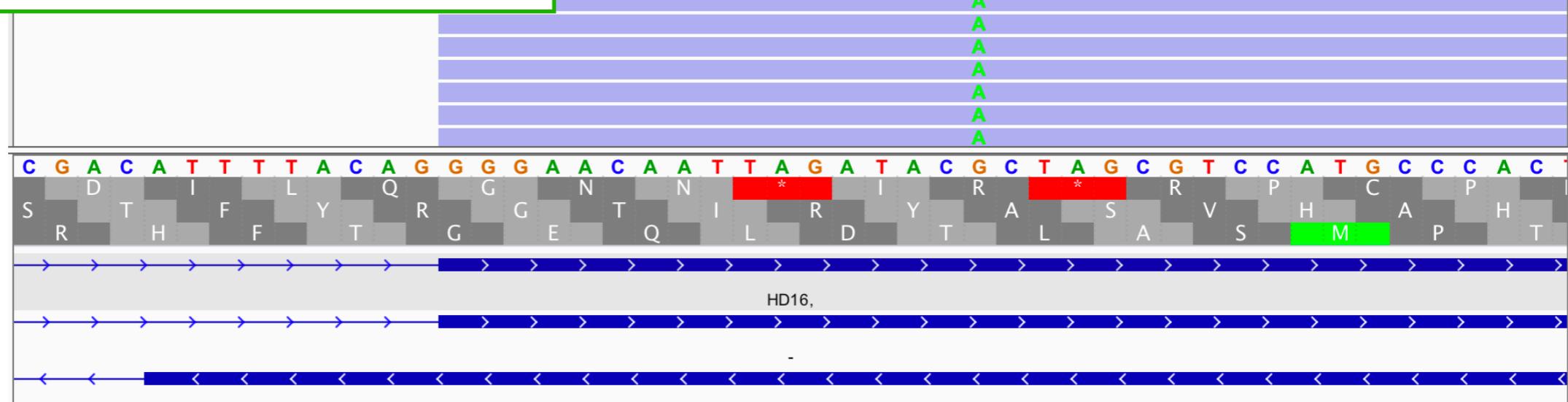
アラインメントとVCFファイルの双方で確認

VCFファイル中の多型情報

```
chr03    33002789      .      G      A      3117.77 PASS
AC=2;AF=1.00;AN=2;DP=92;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;
QD=33.89;SOR=0.782          GT:AD:DP:GQ:PL  1/1:0,92:92:99:3146,276,0
```



BAMファイル中のアラインメント 情報をIGVで可視化



日本晴 (G) とコシヒカリ (A) の多型が確認された

参考図書・ウェブサイト

次世代シークエンサーDRY解析教本（細胞工学別冊）



(監修) 清水 厚志, 坊農 秀雅 発売日： 2015/10/15

NGSデータの解析環境の構築から実行コマンドまで非常に細かく書かれている。



坊農秀雅 (著) 発売日： 2017/9/29



- バイオインフォマティクス全般についてかかれた入門書
- 解析でよく使われるデータベース、ツール、ファイル形式、専門用語などについて書かれている。

その他の参考書籍



NGSアプリケーション RNA-Seq実験ハンドブック～発現解析からncRNA、シングルセルまであらゆる局面を網羅! (実験医学別冊)

鈴木 穂 (編集) 発売日： 2016/3/28



次世代シークエンス解析スタンダード～NGSのポテンシャルを活かしきる WET&DRY (実験医学別冊)
二階堂 愛 (編集) 発売日： 2014/8/23



進展が早い分野なので、最新の情報はネットで入手するのが一番。

まずはGoogleで検索！



Google 検索結果

Trimmomatic インストール

すべて 動画 画像 ニュース ショッピング もっと見る 設定 ツール 約 333 件 (0.31 秒)

Trimmomatic | FASTQ クリーニングツール - bioinformatics - biopapyrus
<https://bi.biopapyrus.jp/rnaseq/qc/trimmomatic.html>

Trimmomatic はアダプターの除去のみならず、リードの末端から一定数の塩基をトリムしたりする、簡単なクオリティフィルタリングも行える。... インストール。Trimmomatic は Java によって書かれている。Trimmomatic 配布ウェブサイトからバイナリファイル (.jar ...)

Bash on Ubuntu on WindowsでSingle-cell RNA sequence解析 ③NGS ...
<https://qiita.com/bioinformatics/items/0d1f3e0a2a2a2a2a2a2a>

2017/05/21 - 前回の続きから : trimmomaticのインストール、実行：シングルエンドの場合、シーケンスデータのクオリティチェックが終わったら、次に低品質なリードやアダプター配列と思しき配列を除去、トリミングしなくてはいけません。このような処理ツールの ...

NGS Surfer's Wiki | Trimmomaticインストールログ
<https://cell-innovation.nig.ac.jp/wiki/tiki-index.php?...Trimmomaticインストールログ>

2016/09/23 - インストール手順は、以下のように \$ cd /work_path \$ wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.36.zip \$ unzip Trimmomatic-0.36.zip \$ cp -r Trimmomatic-0.36 /tool_path/ ...

アダプター配列の除去 cutadapt - The Cat Way
<http://catway.jp/bioinformatics/qc/removeseq.html>

インストール: まず、http://hannonlab.cshl.edu/fastx_toolkit/download.html から pre-compiled binary 版を落として、/usr/local/bin など ... アダプター配列のトリミングする Trimmomatic (Paired-end対応) 概要: paired-end に対応したアダプター配列をトリムする ...

0から始めるNGSデータ解析メモ : Trimmomatic
<http://ngsdamemorandum.blogspot.com/2014/07/trimmomatic.html>

2014/07/24 - だいぶ慣れてきたので、インストールの仕方よりもツールの使い方をメインに記載するかな。アダプタートリマーツールであり、クリーニングも出来る Trimmomatic (<http://www.usadellab.org/cms/?page=trimmomatic>) 使い方としては、 ...

Trimmomatic - 井上 潤
http://www.geocities.jp/ancientfishtree/Trimmomatic_ji.html

2013/11/25 - Q20といったオプションにより、read の両端にあるクオリティーの低い配列を取り除いてくれます。さらに、この処理によって短くなった read を取り除くことも可能です。アダプター配列も取り除いてくれるようです。インストールすると、メーカごとの ...

deep sequenceされたウィルスのアセンブルツール sparNA - macで ...
<http://kazumaxneo.hatenablog.com/?page=1512063534>

2017/11/30 - Trimmomaticでのクオリティトリミング->ホストゲノムへのアライメント->アライメントされなかったリードの抽出、の順番で解析される。公式サイト、<https://bitbucket.org/biobakery/kneadata/wiki/Home>。インストール、依存。Trimmomatic ...

クオリティトリミングツール sickle - macでインフォマティクス
<http://kazumaxneo.hatenablog.com/entry/2017/08/24/002643>

2017/08/24 - Trimmomaticと同様、ペアリードの順番が破壊されないよう、ペアの数を同じに崩して出力できる (orphanリードは別出力)。インストール GitHub GitHub - najoshi/sickle: Windowed Adaptive Trimming for fastq files using quality brewで ...

- 最近では日本語でもかなり情報が充実してきている。
- エラーが出て解析ツールがうまく動かない場合は、エラーメッセージをコピペして検索すると、同じ状況に陥った人が解決方法を示してくれていることが多い。

参考ウェブサイト

進展が早い分野なので、最新の情報はネットで入手するのが一番。

親切な人がツールのインストールや解析方法などを記事にしてくれている

- **RNA-Seq Blog** <http://www.rna-seqblog.com>
- **Qiita** <https://qiita.com>
- **Hatena Blog** <http://hatenablog.com>

同じような問題で困っている人がいたり、解決方法が報告されているかも

- **SEQanswers** <http://seqanswers.com>
- **stackoverflow** <https://stackoverflow.com>

初心者向けにツールやデータベースの使い方を動画で紹介してくれている

- **TogoTV** <http://togotv.dbcls.jp>

解析ツールの取得

<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

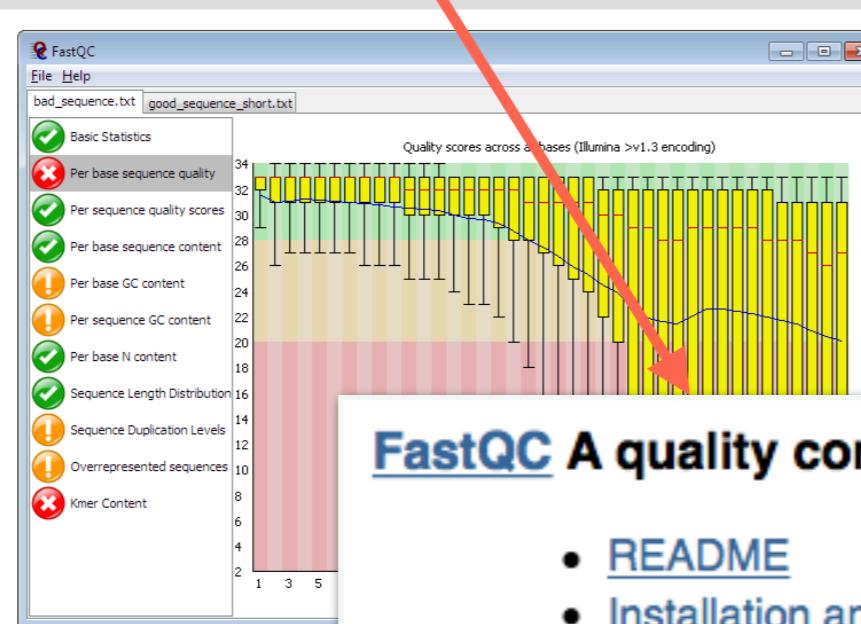
Babraham Bioinformatics

About | People | Services | Projects | Training | Publications

FastQC

Function	A quality control tool for high throughput sequence data.
Language	Java
Requirements	A suitable Java Runtime Environment The Picard BAM/SAM Libraries (included in download)
Code Maturity	Stable. Mature code, but feedback is appreciated.
Code Released	Yes, under GPL v3 or later .
Initial Contact	Simon Andrews

[Download Now](#)



[FastQC A quality control application for high throughput sequence data](#)

- [README](#)
- [Installation and setup instructions](#)
- [Release Notes](#) Please read these before using the program.
- [FastQC v0.11.7 \(Win/Linux zip file\)](#)
- [FastQC v0.11.7 \(Mac DMG image\)](#)
- [Source Code for the latest FastQC release](#)

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

USADELLAB.org

Home Research Education Service & Software
Supporting Info About Us NGS, DE and other things

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 0.38: [binary](#), [source](#) and [manual](#)

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

<https://ccb.jhu.edu/software/hisat2/index.shtml>

HISAT2

graph-based alignment of next generation sequencing reads to a population of genomes

 JOHNS HOPKINS UNIVERSITY
CENTER FOR COMPUTATIONAL BIOLOGY
C C B

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes (as well as to a single reference genome). Based on an extension of BWT for graphs [Sirén et al. 2014], we designed and implemented a graph FM index (GFM), an original approach and its first implementation to the best of our knowledge. In addition to using one global GFM index that represents a population of human genomes, HISAT2 uses a large set of small GFM indexes that collectively cover the whole genome (each index representing a genomic region of 56 Kbp, with 55,000 indexes needed to cover the human population). These small indexes (called local indexes), combined with several alignment strategies, enable rapid and accurate alignment of sequencing reads. This new indexing scheme is called a Hierarchical Graph FM index (HGFM).

HISAT2 2.1.0 release 6/8/2017

- This major version includes the first release of HISAT-genotype, which currently performs HLA typing, DNA fingerprinting analysis, and CYP typing on whole genome sequencing (WGS) reads. We plan to extend the system so that it can analyze not just a few genes, but a whole human genome. Please refer to [the HISAT-genotype website](#) for more details.
- HISAT2 can be directly compiled and executed on Windows system using Visual Studio, thanks to [Nigel Dyer](#).
- Implemented --new-summary option to output a new style of alignment summary, which is easier to parse for programming purposes.
- Implemented --summary-file option to output alignment summary to a file in addition to the terminal (e.g. stderr).
- Fixed discrepancy in HISAT2's alignment summary.
- Implemented --no-template-len-adjustment option to disable automatic template length adjustment for RNA-seq reads.

HISAT2 2.0.5 release 11/4/2016

Version 2.0.5 is a minor release with the following changes.

- Due to a policy change (HTTP to HTTPS) in using SRA data ('--sra-option'), users are strongly encouraged to use this version. As of 11/9/2016, NCBI will begin a permanent redirect to HTTPS, which means the previous versions of HISAT2 no longer works with '--sra-acc' option soon.
- Implemented -I and -X options for specifying minimum and maximum fragment lengths. The options are valid only when used with --no-spliced-alignment, which is used for the alignment of DNA-seq reads.
- Fixed some cases where reads with SNPs on their 5' ends were not properly aligned.
- Implemented --no-softclip option to disable soft-clipping.

 OSI certified

Site Map

[Home](#)
[Manual](#)
[FAQ](#)

News and Updates

New releases and related tools will be announced through the Bowtie [mailing list](#).

Getting Help

Please use hisat2.genomics@gmail.com for private communications only. Please do not email technical questions to HISAT2 contributors directly.

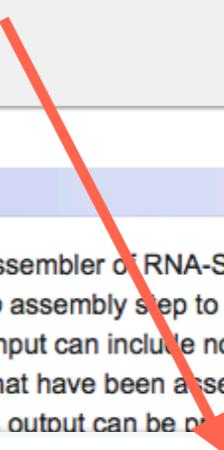
Releases

version 2.1.0	6/8/2017
Source code	
Linux x86_64 binary	
Mac OS X x86_64 binary	
Windows binary	

Please cite:
Kim D, Langmead B and Salzberg SL.
HISAT: a fast spliced aligner with low memory requirements. *Nature Methods* 2015

StringTie

<https://ccb.jhu.edu/software/stringtie/>



The screenshot shows the official website for StringTie, a transcript assembly and quantification tool for RNA-Seq data. The header includes the StringTie logo, the title "StringTie Transcript assembly and quantification for RNA-Seq", and the Johns Hopkins University Center for Computational Biology (CCB) logo. The navigation bar offers links to Home, Manual, FAQ, Overview, News, Obtaining and installing StringTie (which is highlighted with a red underline), Licensing and contact Information, and Publications. The main content area features an "Overview" section describing StringTie's capabilities and its ability to identify differentially expressed genes. Below the overview is a large section titled "Obtaining and installing StringTie". This section provides instructions for downloading the software, mentioning precompiled binary and source package options, and details about specific releases like v1.3.4. A sidebar on the left contains news items, including a recent release note for 2/25/2018.

StringTie

Transcript assembly and quantification for RNA-Seq

JOHNS HOPKINS UNIVERSITY
CENTER FOR COMPUTATIONAL BIOLOGY
CCB

Home Manual FAQ CCB » Software » StringTie

Overview

StringTie is a fast and highly efficient assembler of RNA-Seq alignments into potential transcripts. It uses a novel network flow algorithm as well as an optional *de novo* assembly step to assemble and quantitate full-length transcripts representing multiple splice variants for each gene locus. Its input can include not only the alignments of raw reads used by other transcript assemblers, but also alignments longer sequences that have been assembled from those reads. In order to identify differentially expressed genes between experiments, StringTie's output can be processed by specialized software like Ballgown, Cuffdiff or other programs (DESeq2, edgeR, etc.).

News

2/25/2018 - v1.3.4 release

- GTF/GFF parser adjustments (for transcripts) are no longer taken
- v log output now includes more
- minor performance optimization

Obtaining and installing StringTie

The current version of StringTie can be downloaded as precompiled binary or as a source package:

- [stringtie-1.3.4d.tar.gz](#) : source package
- [stringtie-1.3.4d.Linux_x86_64.tar.gz](#) : Linux x86_64 binary package
- [stringtie-1.3.4d.OSX_x86_64.tar.gz](#) : Apple OS X binary package (for OS X v10.7 and above)

In order to build and install StringTie from the source package the following steps can be taken:

1. Unpack the downloaded StringTie source archive in a directory of your choice, e.g.:

```
cd ~/src/
tar xvfz ~/Downloads/stringtie-VER.tar.gz
```

http://www.htslib.org

Samtools

Home Download ▾ Workflows ▾ Documentation ▾ Support ▾

Samtools

Samtools is a suite of programs for interacting with high-throughput sequencing data. It consists of three separate repositories:

Samtools	Reading/writing/editing/indexing/viewing SAM/BAM/CRAM format
BCFtools	Reading/writing BCF2/VCF/gVCF and short indel sequence variation
HTSlib	A C library for reading/writing htslib

Samtools and BCFtools both use HTSlib internally, but they do not depend on htslib so they can be built independently.

Download

Source code releases can be downloaded from [GitHub](#) or [Sourceforge](#):

[Source release details](#)

Current releases

Samtools and BCFtools are distributed as individual packages. The code uses HTSlib internally, but these source packages contain their own copies of htslib so they can be built independently.

HTSlib is also distributed as a separate package which can be installed if you are writing your own programs against the HTSlib API. HTSlib also provides the `bgzip`, `htsfile`, and `tabix` utilities, so you may also want to build and install HTSlib to get these utilities, or see the additional instructions in [INSTALL](#) to install them from a samtools or bcftools source package.

Download current source releases: [!\[\]\(576ca37522b33960cc8e9d8e91d25c68_img.jpg\) samtools-1.9](#) [!\[\]\(9547f5817d92d898ce9c531448e605c3_img.jpg\) bcftools-1.9](#) [!\[\]\(41cb92af7bfd472172a0bef4c29299bb_img.jpg\) htslib-1.9](#)

See also release notes for [samtools](#), [bcftools](#), and [htslib](#).

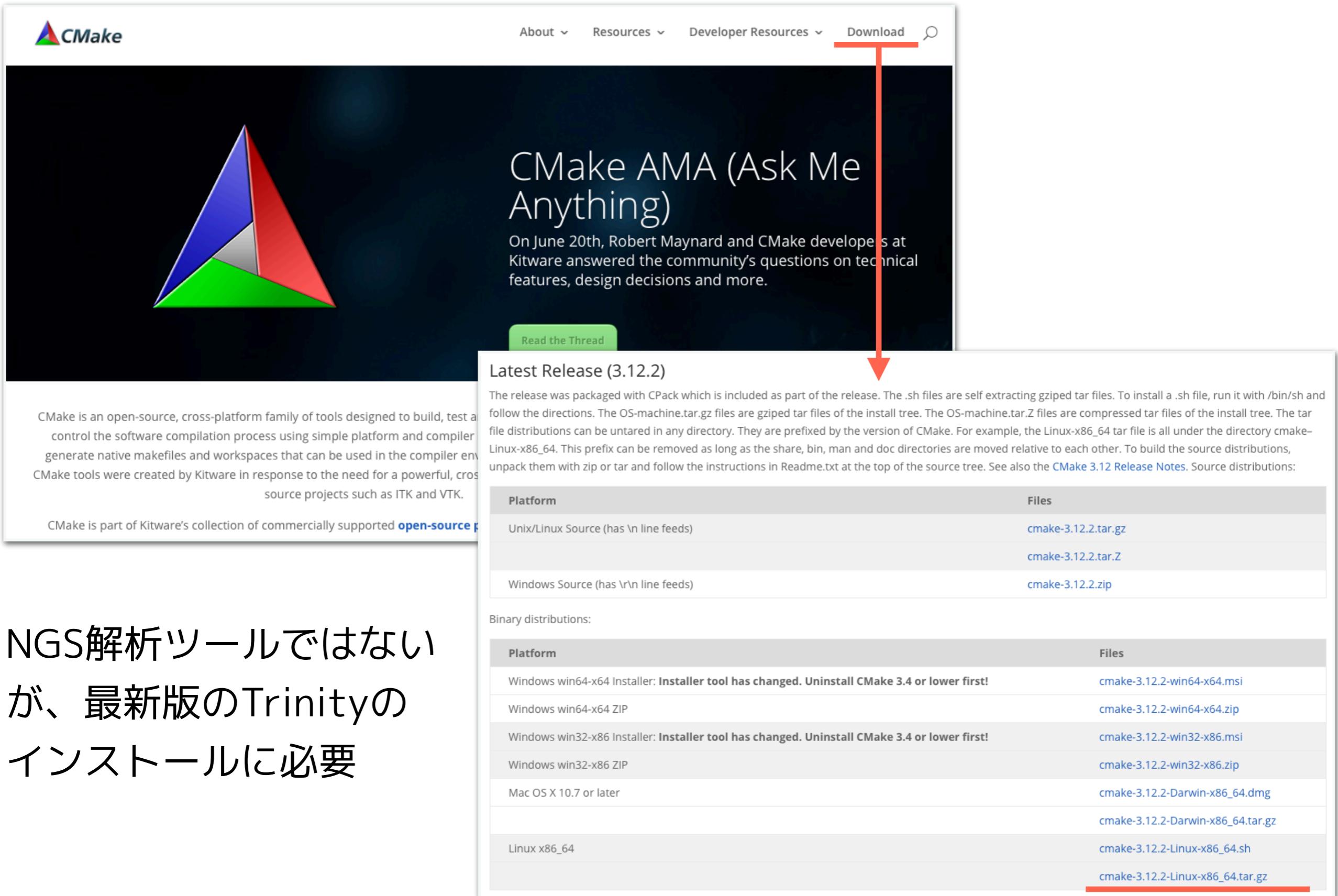
New releases are announced on the [samtools mailing lists](#) and by [@htslib](#) on Twitter. Previous releases are available from the [samtools GitHub organisation](#) (see [samtools](#), [bcftools](#), or [htslib](#) releases) or from the [samtools Sourceforge project](#).

Building and installing

Building each desired package from source is very simple:

```
cd samtools-1.x    # and similarly for bcftools and htslib
./configure --prefix=/where/to/install
make
make install
```

<https://cmake.org>



The screenshot shows the CMake website's download page. At the top, there are navigation menus for About, Resources, Developer Resources, and Download. A red arrow points from the 'Download' menu item to the 'Latest Release (3.12.2)' section. The main content area features a large image of the Trinity logo (a red, green, and blue pyramid) and a section titled 'CMake AMA (Ask Me Anything)' with a link to 'Read the Thread'. Below this, the 'Latest Release (3.12.2)' section is highlighted. It contains a detailed description of the release, mentioning CPack packaging and various file formats like .sh, tar.gz, and tar.Z. Below this, there are two tables: one for 'Platform' distributions and one for 'Binary distributions'. The 'Platform' table lists Unix/Linux Source and Windows Source with their corresponding file names. The 'Binary distributions' table lists various Windows and Mac OS X installers and source files.

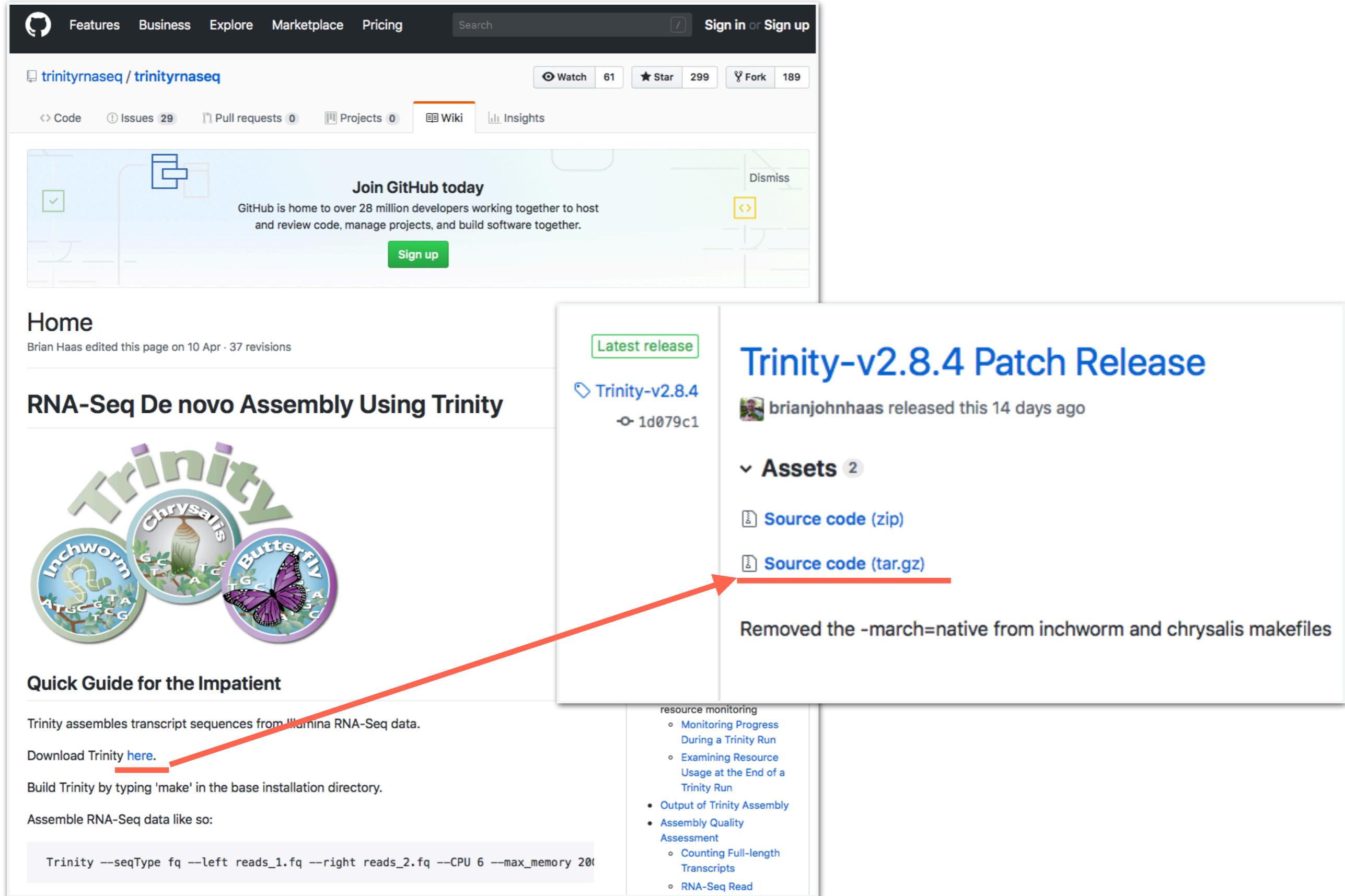
Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.12.2.tar.gz cmake-3.12.2.tar.Z
Windows Source (has \r\n line feeds)	cmake-3.12.2.zip

Platform	Files
Windows win64-x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.12.2-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.12.2-win64-x64.zip
Windows win32-x86 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.12.2-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.12.2-win32-x86.zip
Mac OS X 10.7 or later	cmake-3.12.2-Darwin-x86_64.dmg cmake-3.12.2-Darwin-x86_64.tar.gz
Linux x86_64	cmake-3.12.2-Linux-x86_64.sh cmake-3.12.2-Linux-x86_64.tar.gz

NGS解析ツールではない
が、最新版のTrinityの
インストールに必要

Trinity

<https://github.com/trinityrnaseq/trinityrnaseq/wiki>



The screenshot shows the GitHub wiki page for the Trinity repository. At the top, there's a banner for joining GitHub. Below it, the main content area has a section titled "Home" with a note from Brian Haas about edits. A large image on the left features the word "Trinity" in green with three circular icons: "Inchworm" (green worm), "Chrysalis" (caterpillar), and "Butterfly" (purple butterfly). To the right, a box highlights the "Trinity-v2.8.4" release, which was posted 14 days ago by brianjohnhaas. It includes two "Source code" download links: one for zip and one for tar.gz. A red arrow points from the "Download Trinity here." link in the wiki to the "Source code (tar.gz)" link in the release box. Another red arrow points from the "Build Trinity by typing 'make'" instruction to the "Source code (tar.gz)" link. The bottom right corner contains a sidebar with resource monitoring and assembly quality assessment links.

Join GitHub today

Dismiss

Sign up

Home

Brian Haas edited this page on 10 Apr · 37 revisions

RNA-Seq De novo Assembly Using Trinity



Quick Guide for the Impatient

Trinity assembles transcript sequences from Illumina RNA-Seq data.

Download Trinity [here](#).

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

Assemble RNA-Seq data like so:

```
Trinity --seqType fq --left reads_1.fq --right reads_2.fq --CPU 6 --max_memory 200
```

Latest release

Trinity-v2.8.4
1d079c1

brianjohnhaas released this 14 days ago

Assets 2

Source code (zip)

Source code (tar.gz)

Removed the -march=native from inchworm and chrysalis makefiles

resource monitoring

- Monitoring Progress During a Trinity Run
- Examining Resource Usage at the End of a Trinity Run

- Output of Trinity Assembly
- Assembly Quality Assessment
 - Counting Full-length Transcripts
 - RNA-Seq Read

137

Bowtie2

<http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>



Bowtie 2

Fast and sensitive read alignment



JOHNS HOPKINS
UNIVERSITY

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

>> Version 2.3.4.3 - September 17, 2018

- Fixed an issue causing `bowtie2-build` and `bowtie2-inspect` to output incomplete help text.
- Fixed an issue causing `bowtie2-align` to crash.
- Fixed an issue preventing `bowtie2` from processing paired and/or unpaired FASTQ reads together with interleaved FASTQ reads.

>> Version 2.3.4.2 - August 07, 2018

- Fixed issue causing `bowtie2` to fail in `--fast-local` mode.
- Fixed issue causing `--soft-clipped-unmapped-tlen` to be a positional argument.
- New option `--trim-to N` causes `bowtie2` to trim reads longer than `N` bases to exactly `N` bases. Can trim from either 3' or 5' end, e.g. `--trim-to 5:30` trims reads to 30 bases, truncating at the 5' end.
- Updated "Building from source" manual section with additional instructions on installing TBB.
- Several other updates to manual, inclu

Site Map

[Home](#)
[News archive](#)
[Manual](#)
[Getting started](#)
[Frequently Asked Questions](#)
[Tools that use Bowtie](#)

Latest Release

install with [bioconda](#)

Bowtie2 2.3.4.3	09/17/18
<small>Please cite: Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. <i>Nature Methods</i>. 2012, 9:357-359.</small>	

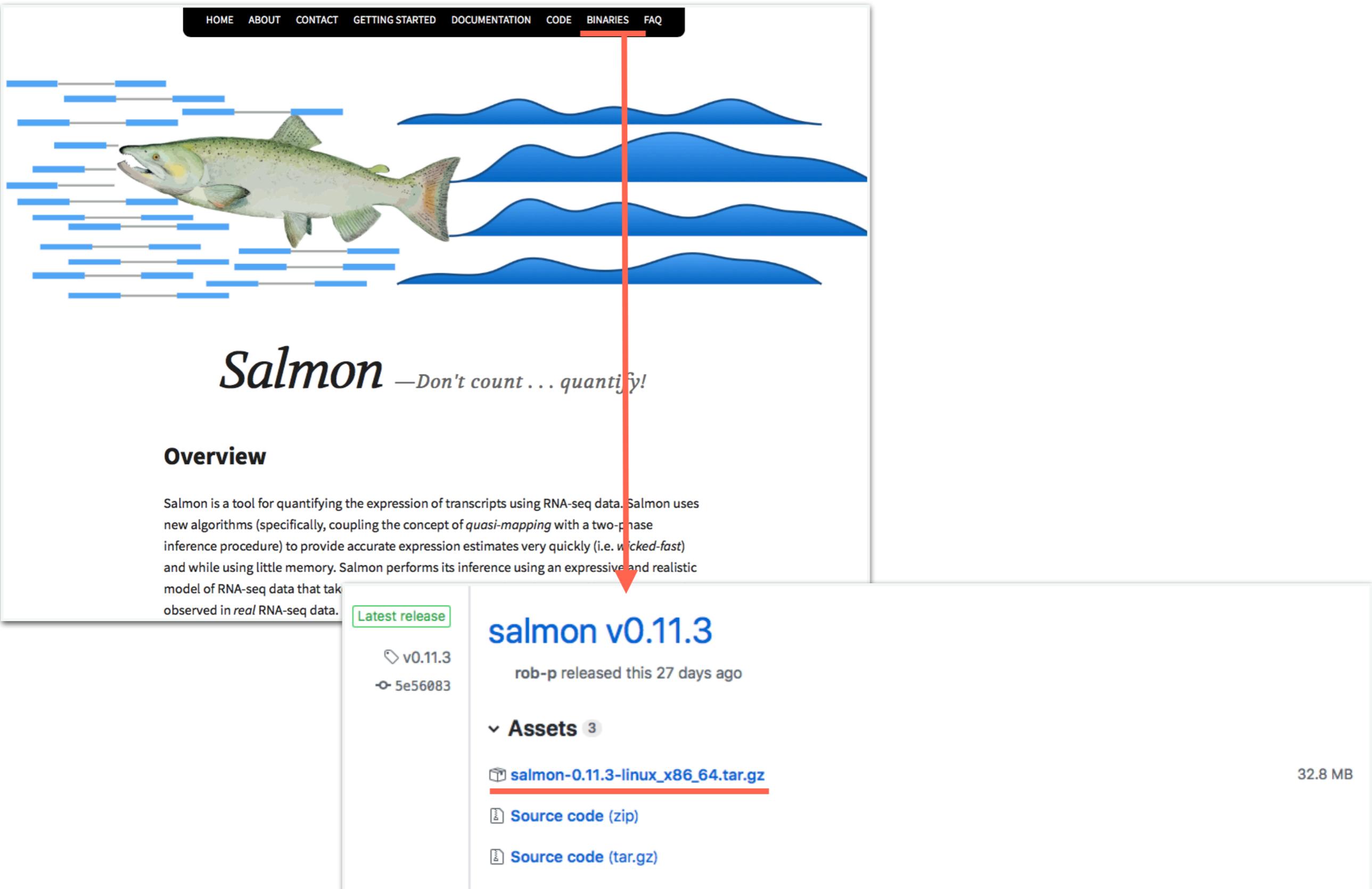
 [Download Latest Version](#)
`bowtie2-2.3.4.3-macos-x86_64.zip` (23.0 MB)

[Home](#) / [bowtie2](#) / 2.3.4.3

Name	Modified	Size	Downloads / Week
Parent folder			
bowtie2-2.3.4.3-source.zip	2018-09-17	17.2 MB	287 
bowtie2-2.3.4.3-linux-x86_64.zip	2018-09-17	66.4 MB	393 
bowtie2-2.3.4.3-macos-x86_64.zip	2018-09-17	23.0 MB	132 
Totals: 3 Items		106.6 MB	812



<https://combine-lab.github.io/salmon/>



The screenshot shows the Salmon GitHub release page for version v0.11.3. A red arrow points from the top navigation bar's "BINARIES" tab down to the "Assets" section of the release details.

Salmon —Don't count . . . quantify!

Overview

Salmon is a tool for quantifying the expression of transcripts using RNA-seq data. Salmon uses new algorithms (specifically, coupling the concept of *quasi-mapping* with a two-phase inference procedure) to provide accurate expression estimates very quickly (i.e. *wicked-fast*) and while using little memory. Salmon performs its inference using an expressive and realistic model of RNA-seq data that takes into account features commonly observed in real RNA-seq data.

Latest release

v0.11.3 · rob-p · 27 days ago · 5e56083

salmon v0.11.3

rob-p released this 27 days ago

Assets (3)

- salmon-0.11.3-linux_x86_64.tar.gz (32.8 MB)
- Source code (zip)
- Source code (tar.gz)

Jellyfish

<http://www.genome.umd.edu/jellyfish.html>

Home | About Us | Contact Us

The University of Maryland Genome Assembly Group Developing Methods for Improving Genome Assembly

JELLYFISH

Newest version released July, 1st 2015 is [Jellyfish 2.2.3](#).

Jellyfish mer counter

What is it?

Jellyfish is a tool for fast, memory-efficient counting of k-mers in DNA. A k-mer is a substring of length k, and counting the occurrences of all such substrings is a central step in many analyses of DNA sequence. JELLYFISH can count k-mers quickly by using an efficient encoding of a hash table and by exploiting the "compare-and-swap" CPU instruction to increase parallelism.

Jellyfish is a command-line program that reads FASTA and multi-FASTA files containing DNA sequences. It outputs its k-mer counts in a binary format, which can be translated into a human-readable text format using the "jellyfish dump" command. See the documentation below for more details.

Jellyfish is distributed as source code under the [GPL license](#). Jellyfish is developed on Linux 64-bit (x86_64). It requires gcc version 4.4 or newer to compile. It is reported to compile on Linux with the clang compiler version 3.0, MacOS X (Intel 64 bit) with gcc version 4.7 and Microsoft Windows 7 with cygwin and gcc.

The current version is version 2.0. The older version 1.1 is still available from the [CBCB group at the University of Maryland](#). The current version does not have any limitation on the size of k-mers, unlike v1.1 which has a limit up to k <= 31. The support for Quake has been dropped in the new version, use version 1.1 with the --no-quake option. The documentation gives some information on how to use Jellyfish and the differences between the new and old versions.

Contact
 For any questions or comments, contact [Guillaume Marçais](#) or [Carl Kingsford](#).

Source

- [Latest source and binaries](#).
- [User guide](#).
- [Git](#).

Version 1.1.12

 gmarcais released this on 1 May · [426 commits](#) to master since this release

Assets 5

 jellyfish-1.1.12.tar.gz	1.05 MB
 jellyfish-linux	5.63 MB
 jellyfish-macosx	434 KB
 Source code (zip)	
 Source code (tar.gz)	

This is an update to version 1 of Jellyfish. It does not bring any new features, only fixes to compile on newer environment.

In addition to the source release jellyfish-1.1.12.tar.gz, we provide binaries for linux and for macosx.

<https://blast.ncbi.nlm.nih.gov/Blast.cgi>

NIH U.S. National Library of Medicine NCBI National Center for Biotechnology Information Sign in to NCBI

BLAST®

Basic Local Alignment Search Tool

BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance. [Learn more](#)

N E W S

A new version (1.4.0) of the BLAST RNA-seq mapping tool, Magic-BLAST, is now available

Tue, 21 Aug 2018 16:00:00 EST [More BLAST news...](#)

Web BLAST

Nucleotide BLAST nucleotide ▶ nucleotide

blastx translated nucleotide ▶ protein

tblastn protein ▶ translated nucleotide

Protein BLAST protein ▶ protein

BLAST Genomes

Enter organism common name, species ID, or taxon ID

Human Mouse

BLAST®

Download BLAST Software and Databases

BLAST+ executables

Do you have difficulties running high volume BLAST searches? Do you have proprietary sequence data to search and can your own server? Do you have your own research pipeline? Have security or IP concerns about sending searches outside of your organization? If you answered yes to any of these questions, read on!

The NCBI provides a suite of command-line tools to run BLAST called BLAST+. This allows users to perform BLAST searches on their own server without size, volume and database restrictions. BLAST+ can be used with a command line so it can be integrated directly into your workflow.

What are the next steps?

Download and install BLAST+. Installers and source code are available from <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/>. Download the databases you need, (see database section below), or create your own. Start searching.

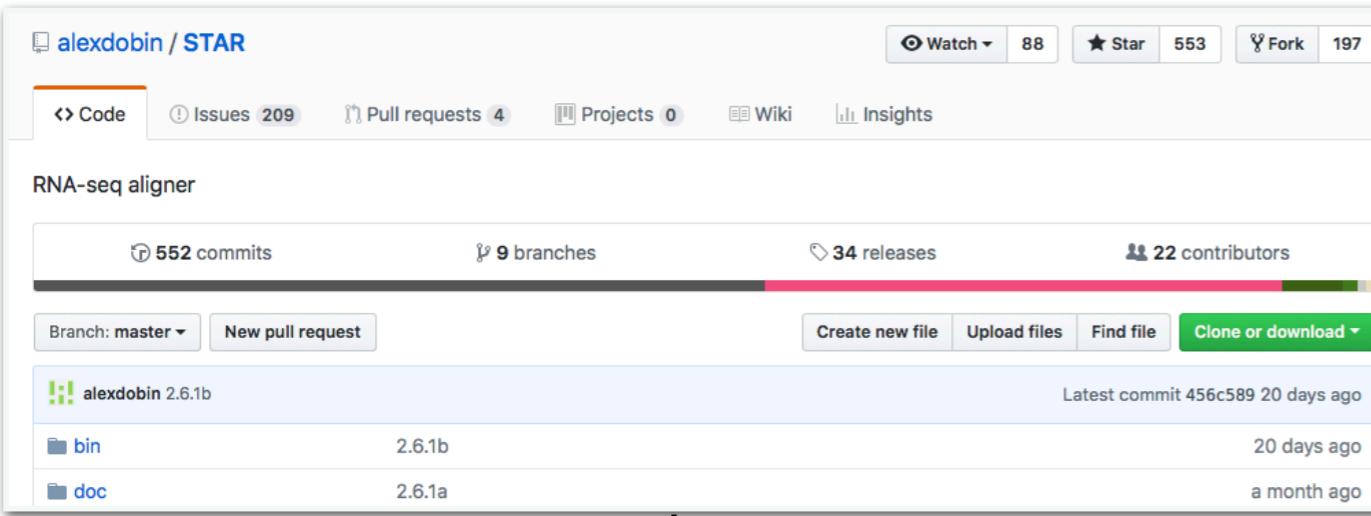
For more details, please see the [BLAST+ user manual](#), the [BLAST Help manual](#), the [BLAST releases notes](#), and the article in BMC Bioinformatics (PubMed link). See our [versioning policy](#).

The BLAST+ suite is the currently supported package. The older C toolkit executables are no longer supported. See our [versioning policy](#).

We are always listening and welcome your feedback at [BLAST Support Center](#).

- [ChangeLog](#)
- [ncbi-blast-2.7.1+-1.src.rpm](#)
- [ncbi-blast-2.7.1+-1.src.rpm.md5](#)
- [ncbi-blast-2.7.1+-1.x86_64.rpm](#)
- [ncbi-blast-2.7.1+-1.x86_64.rpm.md5](#)
- [ncbi-blast-2.7.1+-src.tar.gz](#)
- [ncbi-blast-2.7.1+-src.tar.gz.md5](#)
- [ncbi-blast-2.7.1+-src.zip](#)
- [ncbi-blast-2.7.1+-src.zip.md5](#)
- [ncbi-blast-2.7.1+-win64.exe](#)
- [ncbi-blast-2.7.1+-win64.exe.md5](#)
- ncbi-blast-2.7.1+-x64-linux.tar.gz**
- [ncbi-blast-2.7.1+-x64-linux.tar.gz.md5](#)
- [ncbi-blast-2.7.1+-x64-macosx.tar.gz](#)
- [ncbi-blast-2.7.1+-x64-macosx.tar.gz.md5](#)
- [ncbi-blast-2.7.1+-x64-win64.tar.gz](#)
- [ncbi-blast-2.7.1+-x64-win64.tar.gz.md5](#)
- [ncbi-blast-2.7.1+.dmg](#)
- [ncbi-blast-2.7.1+.dmg.md5](#)

<https://github.com/alexdobin/STAR>



alexdobin / STAR

Code Issues Pull requests Projects Wiki Insights

RNA-seq aligner

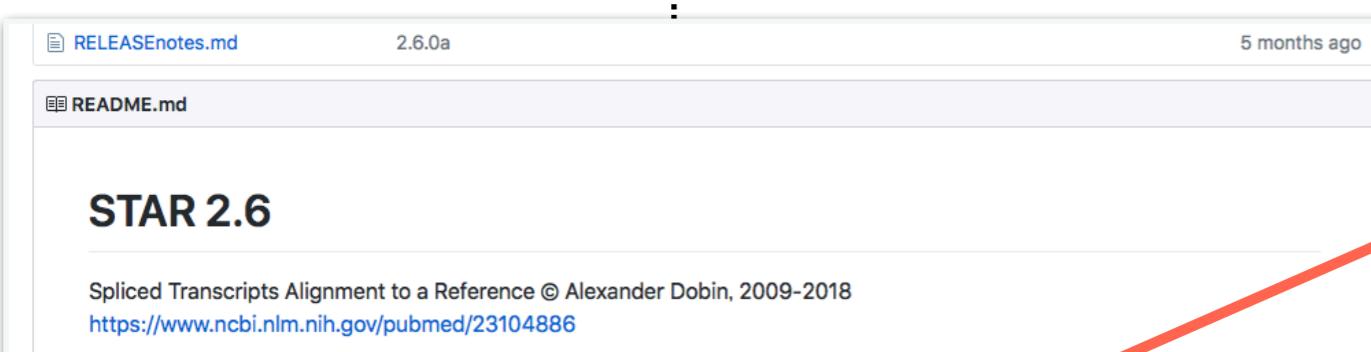
552 commits 9 branches 34 releases 22 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

alexdobin 2.6.1b Latest commit 456c589 20 days ago

bin 2.6.1b 20 days ago

doc 2.6.1a a month ago



RELEASEnotes.md 2.6.0a 5 months ago

README.md

STAR 2.6

Spliced Transcripts Alignment to a Reference © Alexander Dobin, 2009-2018
<https://www.ncbi.nlm.nih.gov/pubmed/23104886>

COMPILING FROM SOURCE

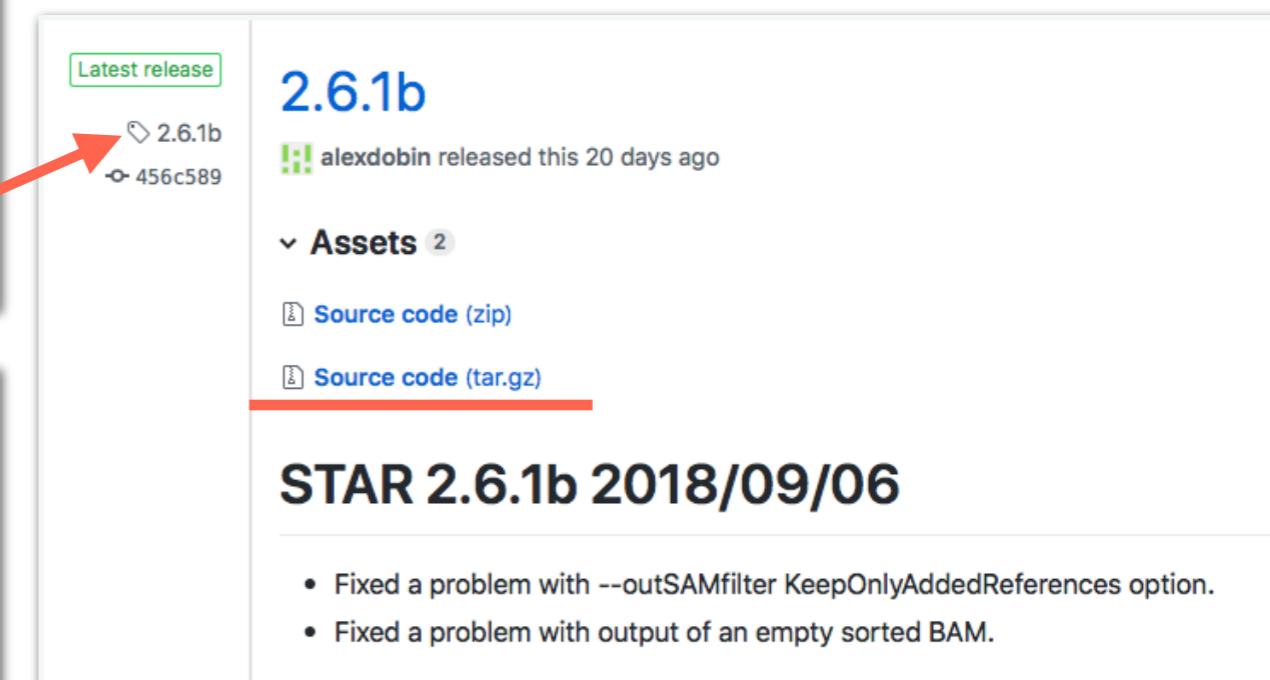
Download the latest [release from](#) and uncompress it

```
# Get latest STAR source from releases
wget https://github.com/alexdobin/STAR/archive/2.6.0a.tar.gz
tar -xzf 2.6.0a.tar.gz
cd STAR-2.6.0a

# Alternatively, get STAR source using git
git clone https://github.com/alexdobin/STAR.git
```

Compile under Linux

```
# Compile
cd STAR/source
make STAR
```



Latest release 2.6.1b 456c589

2.6.1b

alexdobin released this 20 days ago

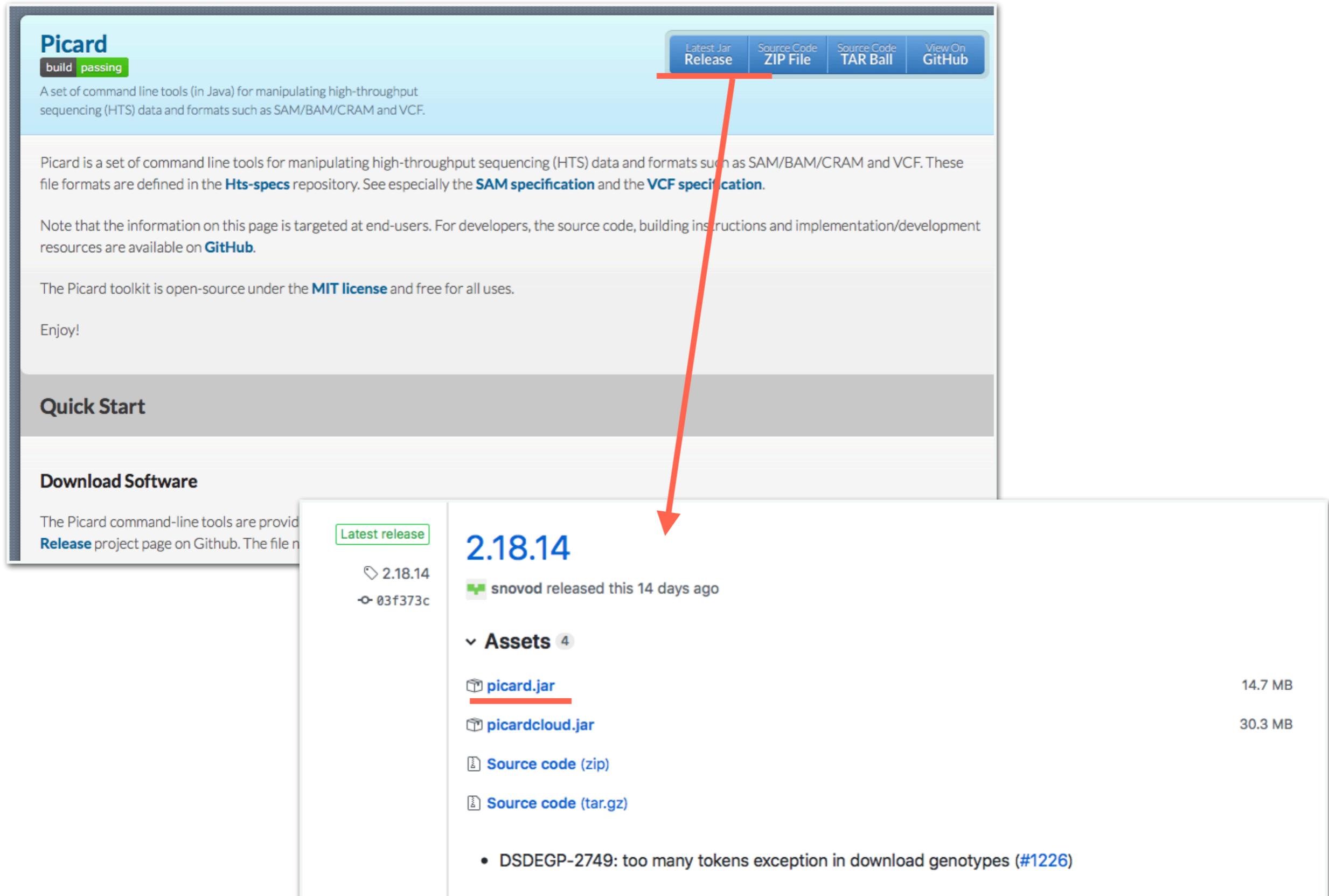
Assets 2

[Source code \(zip\)](#) [Source code \(tar.gz\)](#)

STAR 2.6.1b 2018/09/06

- Fixed a problem with --outSAMfilter KeepOnlyAddedReferences option.
- Fixed a problem with output of an empty sorted BAM.

<https://broadinstitute.github.io/picard/>



The screenshot shows the GitHub release page for Picard version 2.18.14. A red arrow points from the 'Latest Jar Release' button in the top navigation bar down to the 'picard.jar' download link in the assets section.

Picard
build passing

A set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF.

Picard is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF. These file formats are defined in the [Hts-specs](#) repository. See especially the [SAM specification](#) and the [VCF specification](#).

Note that the information on this page is targeted at end-users. For developers, the source code, building instructions and implementation/development resources are available on [GitHub](#).

The Picard toolkit is open-source under the [MIT license](#) and free for all uses.

Enjoy!

Quick Start

Download Software

The Picard command-line tools are provided via the [Latest release](#) project page on Github. The file names are:

Latest release

2.18.14
03f373c

2.18.14

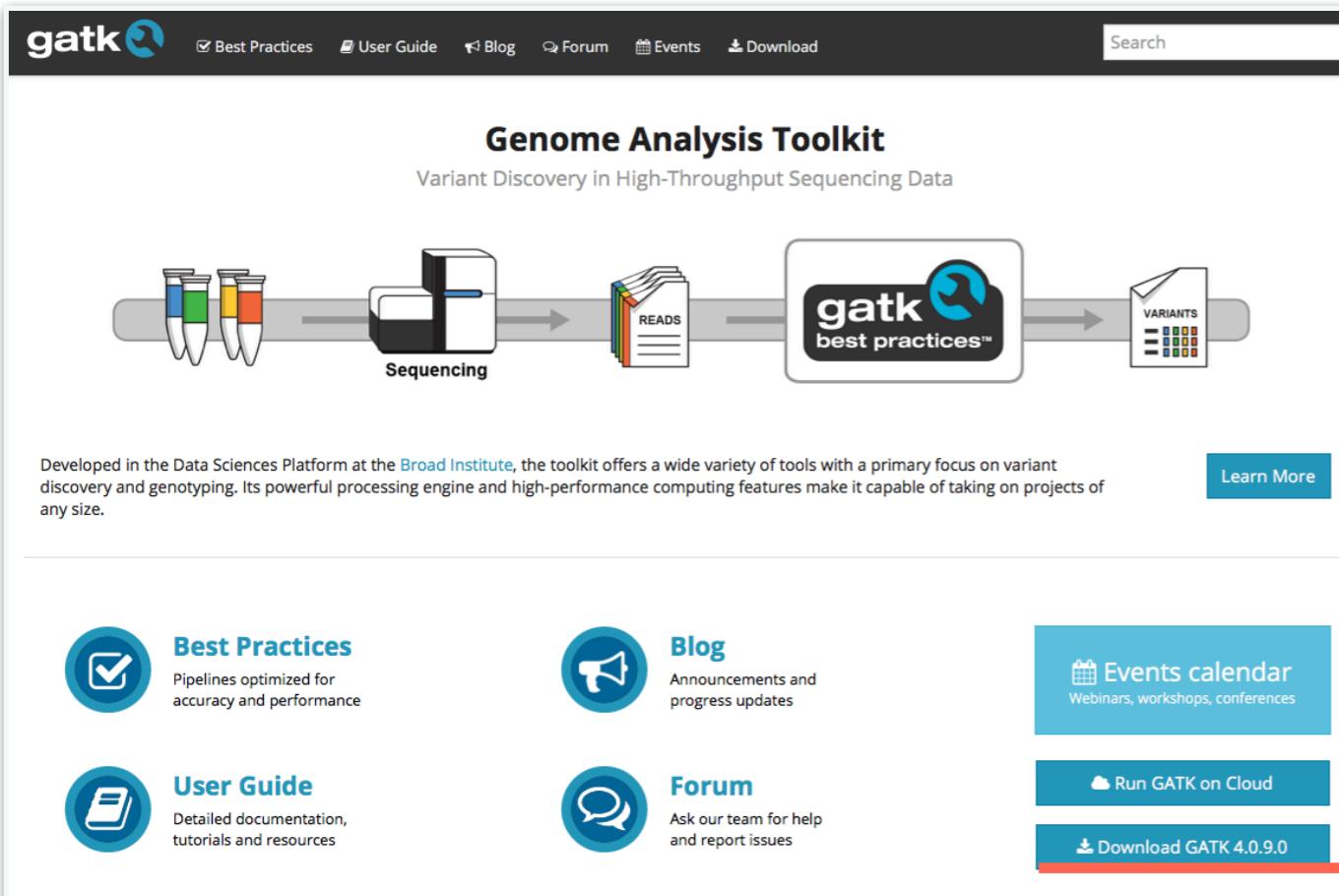
snovod released this 14 days ago

Assets 4

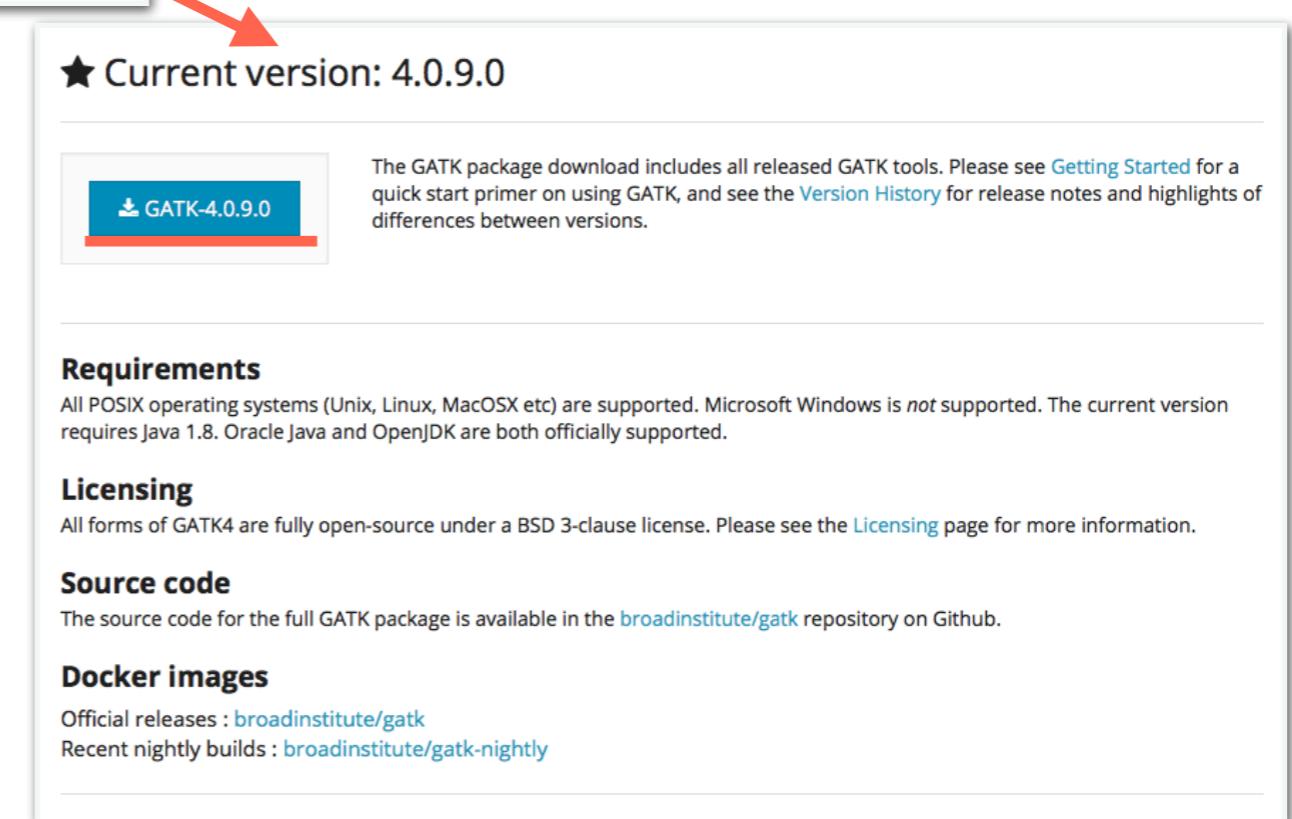
- [picard.jar](#)
- [picardcloud.jar](#)
- [Source code \(zip\)](#)
- [Source code \(tar.gz\)](#)

• DSDEGP-2749: too many tokens exception in download genotypes (#1226)

<https://software.broadinstitute.org/gatk/>



The screenshot shows the GATK homepage. At the top, there's a navigation bar with links for Best Practices, User Guide, Blog, Forum, Events, Download, and a search bar. Below the navigation is a main title "Genome Analysis Toolkit" with a subtitle "Variant Discovery in High-Throughput Sequencing Data". A central diagram illustrates the workflow: "Sequencing" leads to "READS", which then goes through the "gatk best practices™" step to produce "VARIANTS". Below this, a text box states: "Developed in the Data Sciences Platform at the Broad Institute, the toolkit offers a wide variety of tools with a primary focus on variant discovery and genotyping. Its powerful processing engine and high-performance computing features make it capable of taking on projects of any size." A "Learn More" button is located to the right. At the bottom of the page, there are links for Best Practices, Blog, Events calendar, Run GATK on Cloud, and Download GATK 4.0.9.0.



The screenshot shows the "Download GATK 4.0.9.0" page. It starts with a heading "★ Current version: 4.0.9.0". Below it is a download button labeled "GATK-4.0.9.0". To the right, a text block explains: "The GATK package download includes all released GATK tools. Please see [Getting Started](#) for a quick start primer on using GATK, and see the [Version History](#) for release notes and highlights of differences between versions." Below this are sections for Requirements, Licensing, Source code, and Docker images, each with their respective descriptions.

★ Current version: 4.0.9.0

[GATK-4.0.9.0](#)

The GATK package download includes all released GATK tools. Please see [Getting Started](#) for a quick start primer on using GATK, and see the [Version History](#) for release notes and highlights of differences between versions.

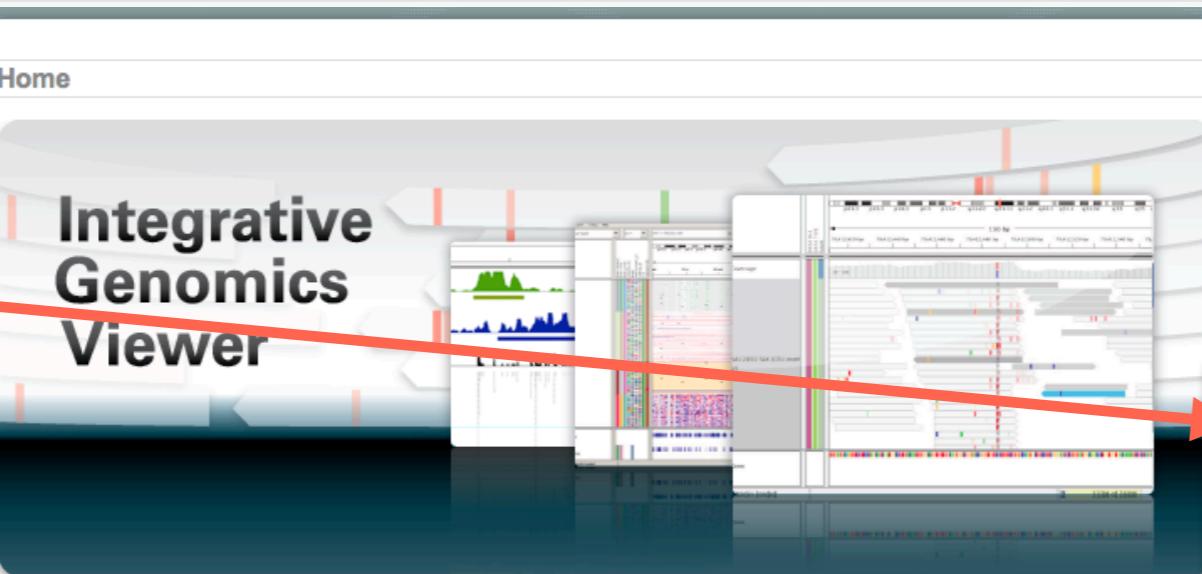
Requirements
All POSIX operating systems (Unix, Linux, MacOSX etc) are supported. Microsoft Windows is *not* supported. The current version requires Java 1.8. Oracle Java and OpenJDK are both officially supported.

Licensing
All forms of GATK4 are fully open-source under a BSD 3-clause license. Please see the [Licensing](#) page for more information.

Source code
The source code for the full GATK package is available in the [broadinstitute/gatk](#) repository on Github.

Docker images
Official releases : [broadinstitute/gatk](#)
Recent nightly builds : [broadinstitute/gatk-nightly](#)

<https://software.broadinstitute.org/software/igv/>



A red arrow points from the IGV software interface on the left to the "Downloads" section on the right.

Downloads

Integrative Genomics Viewer - IGV 2.4

NOTE: IGV 2.4.x releases require Java 8. For Java 10 see the [development snapshot build](#).

Install IGV

Download IGV Mac App

Download and unzip the Mac App Archive, then double-click the IGV application to run it. The application can be moved to the Applications folder, or anywhere else



Download IGV on Windows

Download and unzip the Archive, then double-click the igv.bat file to run IGV. See [readme.txt](#) to run IGV from the command line



For high DPI screens: Use Java 10 and the [development snapshot build](#) of IGV.

Download IGV to run on Linux / MacOS command line

Download and unzip the Archive. See the downloaded [readme.txt](#) for further instructions.



.shや.batファイルを編集することで使用メモリ量を変更できる。データ量が多い場合にはこちらがおすすめ。

RやRStudioのコンソールで以下のコマンドを入力、実行する。

Rのパッケージの場合

tidyverse

```
> install.packages("tidyverse")
```

R/Bioconductorのパッケージの場合

edgeR, ballgown, genefilter

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("edgeR")
> biocLite("ballgown")
> biocLite("genefilter")
```

*使用する際にはlibrary()関数でパッケージを読み込む