

# RNA-Seqによる 大量発現データ解析の基礎・応用

農研機構 基盤技術研究本部  
高度分析研究センター ゲノム情報大規模解析ユニット  
川原 善浩

# 本日の演習の流れ

9時00分-12時00分

リファレンスゲノム情報を用いたRNA-Seq解析



13時00分-15時00分

De novoトランскриプトーム解析

遺伝子機能アノテーション

適当に休憩を挟みながら進めていきます。

ご質問がありましたら、途中でも構いませんので声を掛けてください。

# RNA-Seqとは

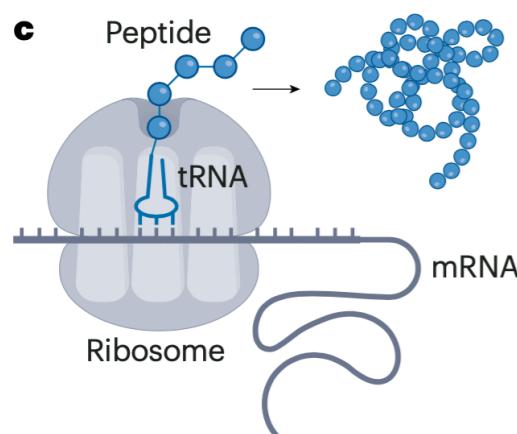
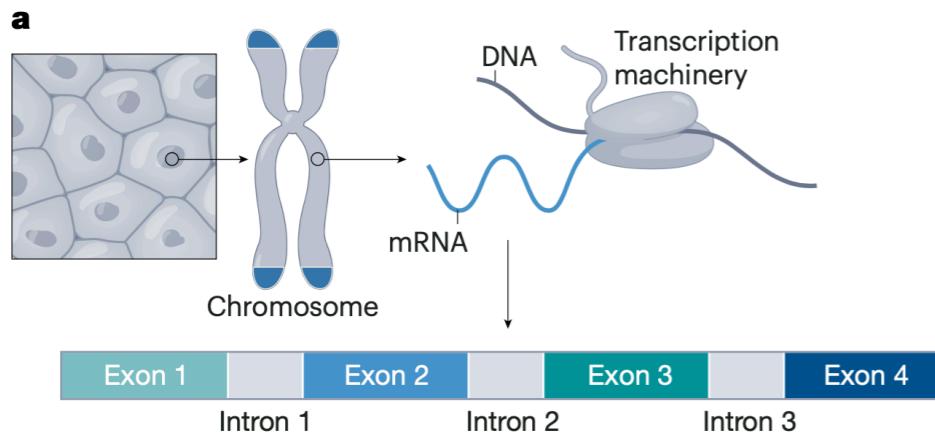


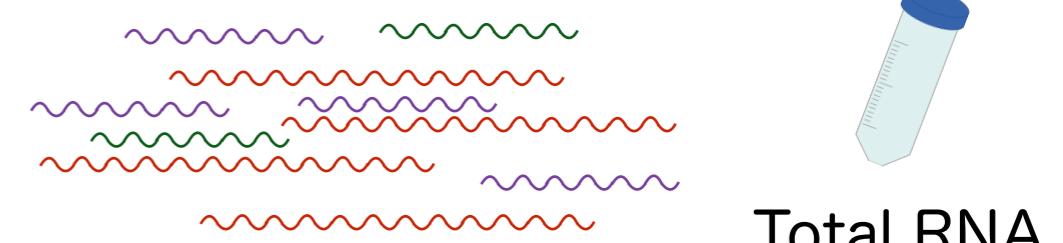
Fig. 1 from Amaral et al. 2023



## データ解析

遺伝子発現プロファイル解析や発現している  
遺伝子配列の取得など、様々な目的に利用可能

## サンプリング、RNA抽出



Total RNA

## ライブラリ調製、シーケンシング



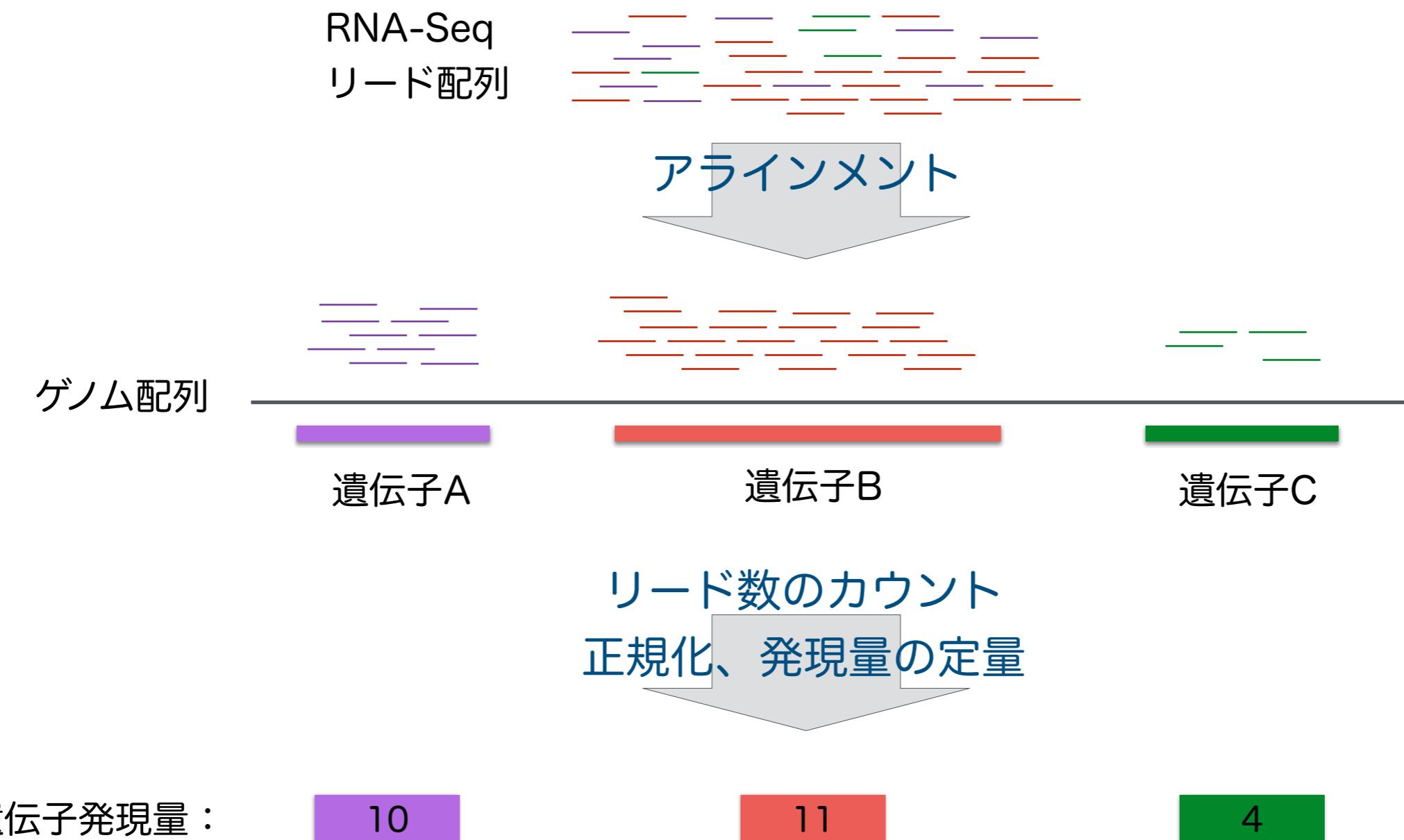
リード配列

ATGCGCATCGTATCGATGATGCGCATCGT  
ATCGATGCGCATCGTATCGATGATGCGC  
ATCGTATCGATGCGCATCGTATCGATGCG  
CATCGTAGGGGGTCGATGCGCATCGTAT  
CGCGCATCGTATCGATGCGATGCGCATCGT  
ATCGATGCGCATCGTATCGATGCGCATCG



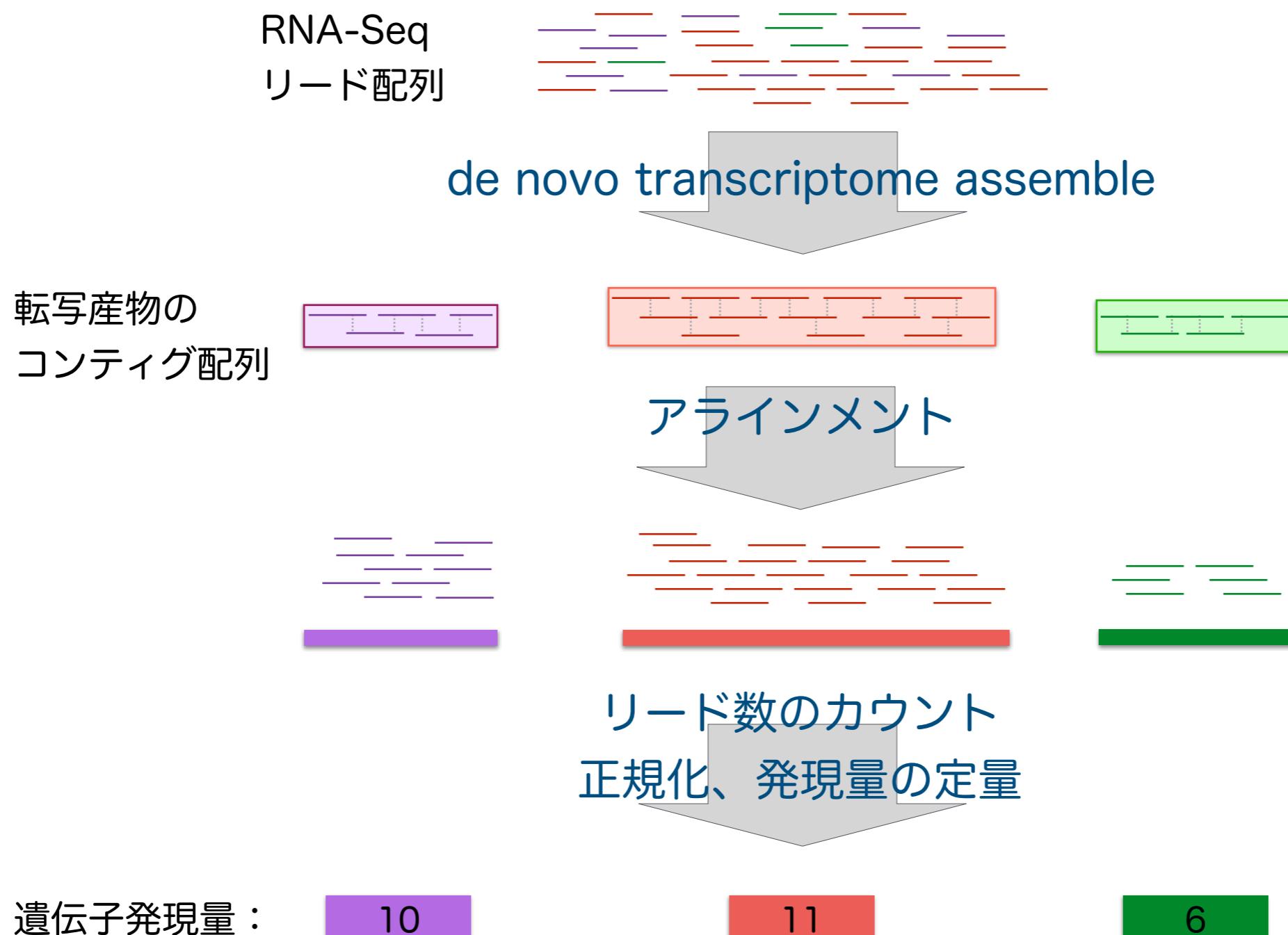
# 手法1：ゲノム配列や遺伝子アノテーションを用いた遺伝子発現解析

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。



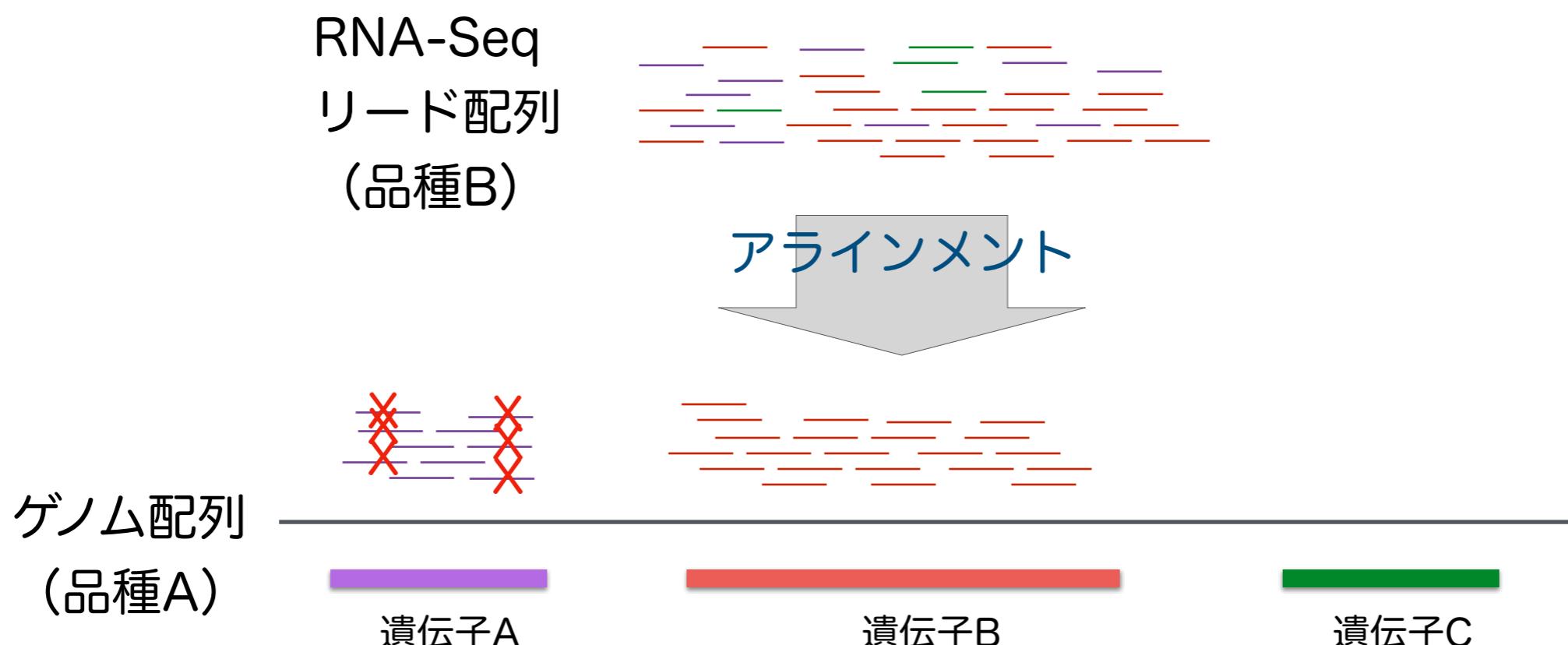
## 手法2：de novo transcriptome assemble法による遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物配列を再構築する。  
その後、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して遺伝子発現を定量する。



# 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リードをリファレンス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、つまり種間や品種間の転写領域上の多型情報を得ることができる。



X:品種AとBでゲノム配列が不一致する箇所

今回の演習では行いません

# 過去の講義資料を公開しています

[https://github.com/YoshiKawahara/NGS\\_HandsOnWorkshop](https://github.com/YoshiKawahara/NGS_HandsOnWorkshop)

YoshiKawahara / NGS\_HandsOnWorkshop Public

Code Issues Pull requests Actions Projects Security Insights

master 1 branch 1 tag

Go to file Code About

Hands-on workshop on NGS data analysis @ NARO

YoshiKawahara Delete RNA-Seq\_analysis\_2022/Desktop/RNA-Seq/useref direc... aff25db on Nov 17, 2022 58 commits

RNA-Seq\_analysis\_2018 Delete NGS\_Workshop\_RNA-Seq\_2018\_public.pdf 4 years ago

RNA-Seq\_analysis\_2019 Add files via upload 4 years ago

RNA-Seq\_analysis\_2021 Add files via upload 2 years ago

RNA-Seq\_analysis\_2022 Delete RNA-Seq\_analysis\_2022/Desktop/RNA-Seq/useref directory

README.md Update README.md

README.md

Presentation, scripts and data for NGS Hands-on workshop @ NARO(Tsukuba)

過去の資料は[NGS\\_HandsOnWorkshop\\_Archive](#)に移動しました。

Files

master

RNA-Seq\_analysis\_2018

RNA-Seq\_analysis\_2019

RNA-Seq\_analysis\_2021

RNA-Seq\_analysis\_2022

NGS\_HandsOnWorkshop / RNA-Seq\_analysis\_2022 / NGS\_Workshop\_RNA-Seq\_2022\_public.pdf

16 MB

本日の流れ

9時00分-12時00分  
「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-15時00分  
「RNA-Seqによる大量発現データ解析の応用」

NARO

適当に質問タイム、休憩を挟みながら・・・

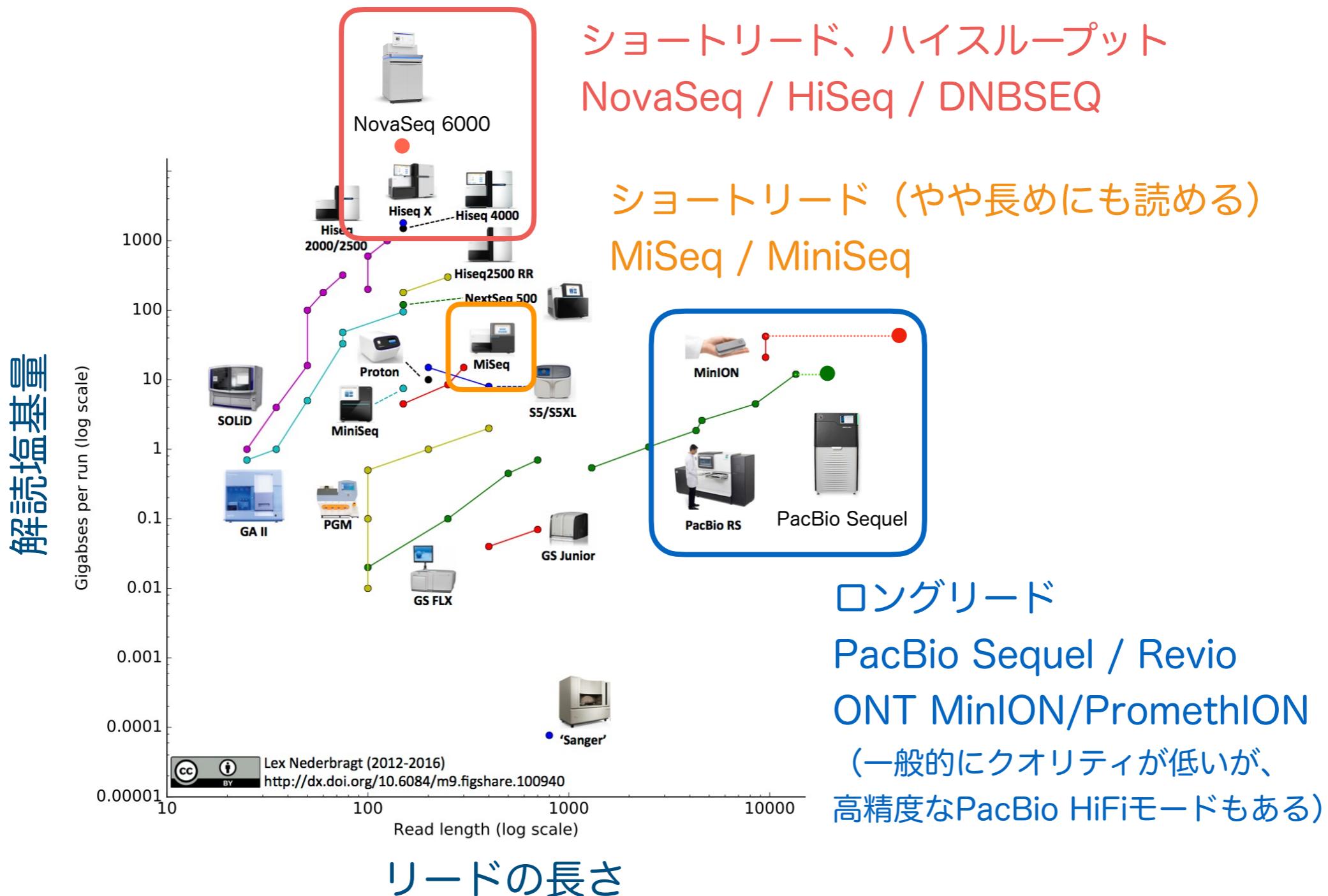
- リファレンス情報を用いたRNA-Seq解析
- RNA-Seqデータを用いた多型検出

の演習をおこないます。

目的やデータ解析方法に応じて、  
「サンプリング」  
「ライブラリ調製」  
「シーケンシング」  
について適切な方法を考えましょう。

「データが出てから考える」では手遅れになることも

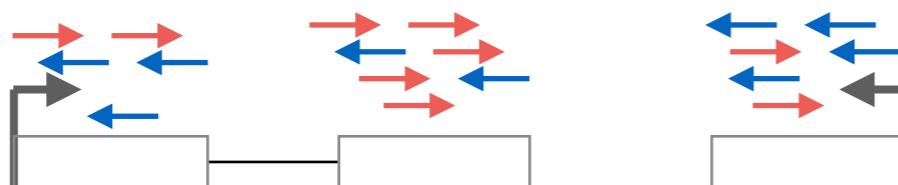
# シークンサーにも色々な種類がある



引用：<https://flxlexblog.wordpress.com/2016/07/08/developments-in-high-throughput-sequencing-july-2016-edition/>  
を一部改変

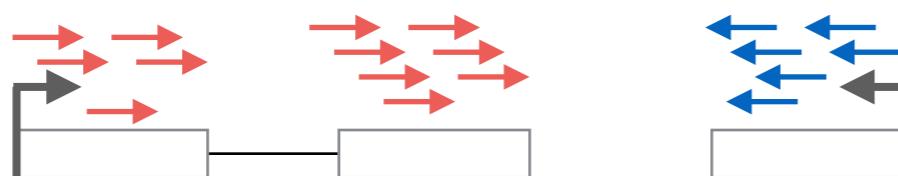
# ライブラリ調製やシークエンシング方法にもいろいろある

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。

cDNA断片の片側だけを読むか両端を読むかの違い。ペアリード間の距離 (cDNA断片長) の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。

シーケンシング

single read  
(SR/SE)

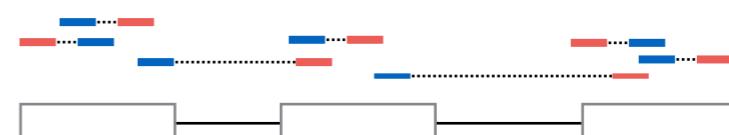


cDNA 断片

アラインメント



paired-end read  
(PE)



# 目的にあったシークエ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

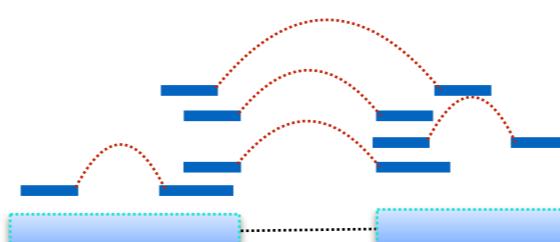
HiSeq, single-read, 50bp



遺伝子構造も発現量も  
どちらも知りたい

リード数と遺伝子構造の  
情報をバランスよく得る

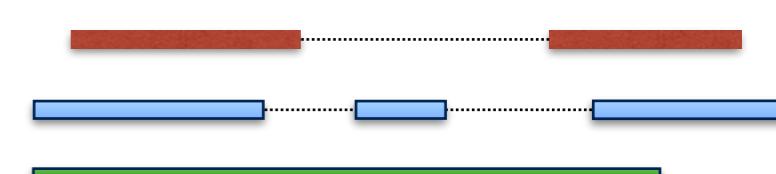
HiSeq, paired-end, 100/150bp



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  
転写産物の全長配列を得る

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
- ▶ 遺伝子機能アノテーション

# 演習用データの準備

以下の要領で演習用データ（/work/NGSworkshop\_23/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。アカウント名（guest29）は各人で異なります。

```
$ 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/guest29
$ cp /work/NGSworkshop_23/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
...
-rw-rw-r-- 1 guest29 guest29 237M 9月 3 14:17 RNA-Seq.tar.gz
...
$ tar xfz RNA-Seq.tar.gz ..... 解析用データを解凍、展開
$ ls -lh
...
drwxrwxr-x 6 guest29 guest29 105 9月 3 14:11 RNA-Seq
-rw-rw-r-- 1 guest29 guest29 237M 9月 3 14:17 RNA-Seq.tar.gz
...
$ ls RNA-Seq ..... 解析用ディレクトリの中身を確認
annot data denovo install_tools.sh tool useref
```

## 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アセンブルによるRNA-Seq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
- annot : RNA-Seqデータを用いた遺伝子アノテーション解析用のディレクトリ
- tool : インストールされた解析ツール群が置かれたディレクトリ
- install\_tools.sh : 解析ツール群をインストールするためのシェルスクリプト (事前に実行済み)

# 解析ツールの取得、インストール

## 解析サーバ (Linux)

### リファレンスゲノム情報を用いたRNA-Seq解析

1. FastQC (v0.12.1)
2. Trimmomatic (v0.39)
3. HISAT2 (v2.2.1)
4. StringTie (v2.2.1)
5. Samtools (v1.18)
6. gffcompare (v0.12.6)
7. gffread (v0.12.7)

### De novoトランскriプトーム解析

1. Trinity (v2.15.1)
2. BLAST+ (v2.14.1)
3. Bowtie2 (v2.5.1)

### 遺伝子機能アノテーション

1. TransDecoder (v5.7.1)
2. Trinotate (v4.0.2)

## ローカル環境 (Win/Mac)

### データの可視化、発現変動解析

1. IGV (v2.16.2)
  2. R (v4.3.1)
  3. RStudio Desktop (v2023.09.0+463)
- R/Bioconductor
4. DESeq2 (v1.40.2)
  5. その他のパッケージ (tidyverse, apeglm, pheatmap, RColorBrewer)

- ・バージョンが変わると解析結果が変わることが多い
- ・新しいバージョンの方がバグが修正されていたり、新機能が追加されてたりする。
- ・自分がどのバージョンを使って解析しているのかをちゃんと把握することが大事。
- ・論文のM&Mには使用したツールとそのバージョンを必ず記載しましょう（データの再現性）。

# 解析ツールの取得、インストール

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。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の「Docker」や「Singularity」を用いてインストールできるものもある。

例) Samtoolsのウェブサイト (<https://www.htslib.org/download/>)

Samtools

Home Download ▾ Workflows ▾ Documentation ▾ Support ▾

## Current releases ダウンロード

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:

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
```

インストール方法の説明

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
```

# 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
```

- install\_tools.sh -

```
#!/bin/bash
```

```
echo START: `date`
```

ツールインストールの開始時間を出力

```
mkdir tool
```

```
cd tool
```

```
ToolDir=`pwd`
```

```
##### for "useref" analysis
```

#で始まるのはコメント行

```
### install FastQC
```

FastQCのダウンロード

```
wget --no-check-certificate https://www.bioinformatics.babraham.ac.uk/projects/fastqc/  
fastqc_v0.12.1.zip
```

```
unzip fastqc_v0.12.1.zip
```

解凍・展開

```
chmod +x FastQC/fastqc
```

fastqcプログラムに実行権限を与える

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 -

```
### install Trimmomatic  
wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip  
unzip Trimmomatic-0.39.zip ..... Trimmomaticのダウンロード  
  
### install HISAT2  
wget -O hisat2-2.2.1-Linux_x86_64.zip https://cloud.biohpc.swmed.edu/index.php/s/oTtGWbWjaxsQ2Ho/download  
unzip hisat2-2.2.1-Linux_x86_64.zip ..... HISAT2のダウンロード  
  
### install StringTie  
wget http://ccb.jhu.edu/software/stringtie/dl/stringtie-2.2.1.Linux_x86_64.tar.gz  
tar xfz stringtie-2.2.1.Linux_x86_64.tar.gz ..... StringTieのダウンロード  
..... 解凍・展開
```

\*.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 xfj」)。

# 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 -

```
### install Samtools  
wget https://github.com/samtools/samtools/releases/download/1.18/samtools-1.18.tar.bz2  
tar xfj samtools-1.18.tar.bz2 ..... Samtoolsのダウンロード  
cd samtools-1.18 ..... 解凍・展開  
make ..... Samtoolsディレクトリに移動  
make prefix=`pwd` install ..... コンパイル  
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
```

```
### install gffcompare  
wget https://github.com/gperteau/gffcompare/releases/download/v0.12.6/  
gffcompare-0.12.6.Linux_x86_64.tar.gz  
tar xfz gffcompare-0.12.6.Linux_x86_64.tar.gz ..... gffcompareのダウンロード  
..... 解凍・展開
```

```
### install gffread  
wget https://github.com/gperteau/gffread/releases/download/v0.12.7/  
gffread-0.12.7.Linux_x86_64.tar.gz  
tar xfz gffread-0.12.7.Linux_x86_64.tar.gz ..... gffreadのダウンロード  
..... 解凍・展開
```

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 -

```
##### for "denovo" analysis  
### install Trinity  
wget https://data.broadinstitute.org/Trinity/TRINITY_SINGULARITY/trinityrnaseq.v2.15.1.simg  
  
# install Bowtie2  
wget --no-check-certificate https://sourceforge.net/projects/bowtie-bio/files/  
bowtie2/2.5.1/bowtie2-2.5.1-linux-x86_64.zip  
unzip bowtie2-2.5.1-linux-x86_64.zip ..... 解凍・展開  
  
# install BLAST+  
wget https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.14.1+-x64-  
linux.tar.gz  
tar xfz ncbi-blast-2.14.1+-x64-linux.tar.gz ..... 解凍・展開
```

Singularityのイメージファイル (.simgや.sif) はダウンロードするだけで良い。

使用方法は後ほど説明します。

TrinityのSingularityイメージのダウンロード

Bowtie2のダウンロード

解凍・展開

NCBI BLAST+のダウンロード

解凍・展開

# 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 -

```
##### for "annot" analysis  
### install TransDecoder  
wget https://data.broadinstitute.org/Trinity/CTAT_SINGULARITY/MISC/TransDecoder/  
transdecoder.v5.7.1.simg  
  
### install Trinotate  
wget https://data.broadinstitute.org/Trinity/TRINOTATE_SINGULARITY/trinotate.v4.0.2.simg
```

TransDecoderのSingularityイメージのダウンロード

TrinotateのSingularityイメージのダウンロード

# ツールのインストール：バイオインフォ解析の壁

個々のツールのインストールはbiocondaなどを使うことでかなり簡単にできるようになった。

Trinityを使えるようにするには、以下のツール類をインストールし、互いに連携するようにパスや環境変数を設定する必要がある。

- Trinity
- bowtie2
- jellyfish
- salmon
- samtools
- それぞれのツールは指定されているバージョンでないと動かないこともある。
- OSの種類やバージョンによってもインストール方法が異なることがある。

ハマるとかなり大変！

## 解析環境（ホストOS）

- Trinity v.0.0.0
- bowtie2 v.0.0.0
- bowtie2 v.X.X.X
- jellyfish v.0.0.0
- salmon v.0.0.0
- samtools v.0.0.0
- samtools v.X.X.X
- HISAT2 v.0.0.0
- samtools v.△.△.△
- StringTie v.0.0.0
- StringTie v.X.X.X

## ホストOS

### Trinity解析環境

- Trinity v.0.0.0
- bowtie2 v.0.0.0
- jellyfish v.0.0.0
- salmon v.0.0.0
- samtools v.0.0.0

### 別の解析環境

- bowtie2 v.X.X.X
- HISAT2 v.0.0.0
- StringTie v.0.0.0
- samtools v.X.X.X

コンテナ型仮想環境

# コンテナ型仮想環境を用いた解析環境の構築



## Docker

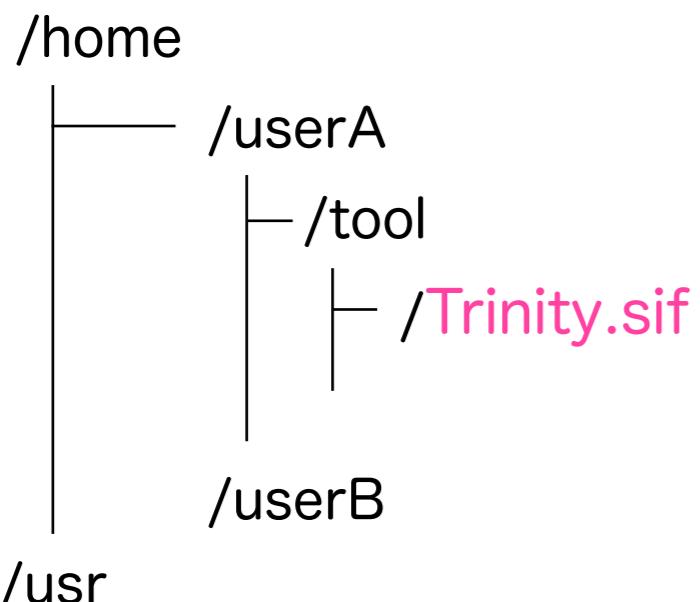
- ・コンテナエンジンを常駐サービスとして起動する必要がある
- ・root（管理者）権限が必要



## Singularity

- ・ユーザ権限で利用可能
- ・使用する際にユーザがコマンドで起動可能
- ・dockerfileが利用可能

## ホストOS

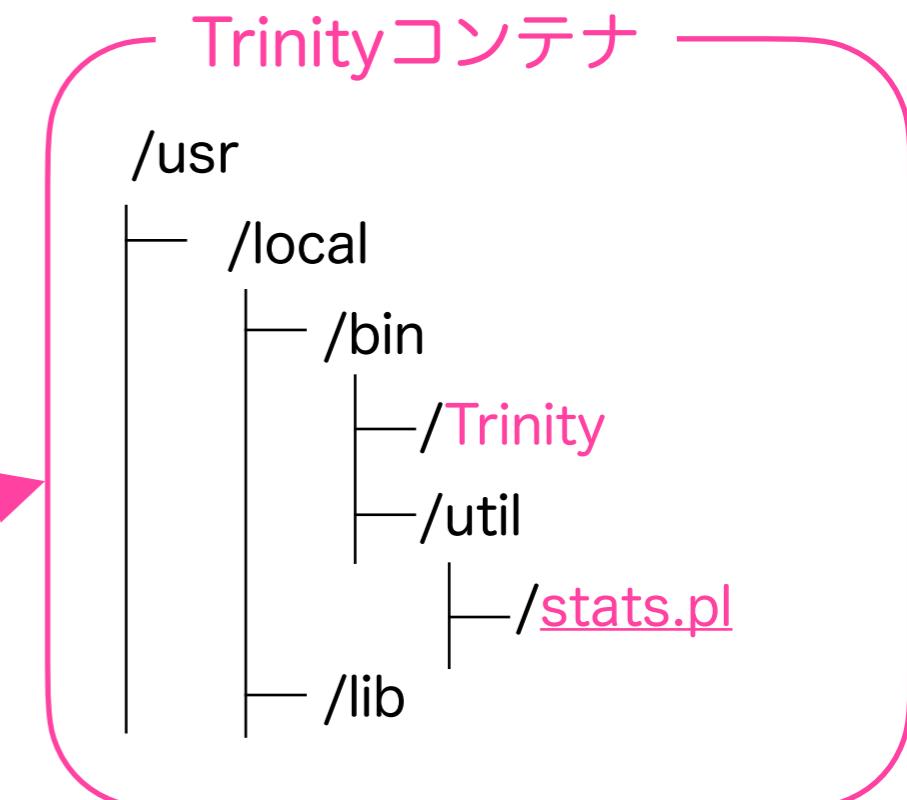


Singularity コンテナ起動し、コンテナ内のプログラムを実行

```
$ singularity exec -e Trinity.sif Trinity  
--output ..
```

```
$ singularity exec -e Trinity.sif /usr/  
local/bin/util/stats.pl
```

インストールはSingularityイメージファイル (.sif/.simg)  
をダウンロードして置くだけ



# 解析ツールのダウンロードとインストール

事前に実行済みですので、演習中は実行しないでください

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2時間)

1. 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標準エラー出力をターミナル画面に表示する。

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

2. 以下のようなコマンドを実行すると、標準出力と標準エラー出力を合わせてターミナル画面に表示しながら、ファイルにも書き出してくれる。こうしておくとあとでログを見直すことができ便利。

```
$ bash ./install_tools.sh 2>&1 | tee install_tools.log
```

- ・ 一連のコマンドが書かれたシェルスクリプトを作成すると、コマンドを連續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ディレクトリ以下にインストールされる。

\$ ls tool		
bowtie2-2.5.1-linux-x86_64	ncbi-blast-2.14.1+	
bowtie2-2.5.1-linux-x86_64.zip	ncbi-blast-2.14.1+-x64-linux.tar.gz	
: (省略)		
hisat2-2.2.1	trinityrnaseq.v2.15.1.simg	
hisat2-2.2.1-Linux_x86_64.zip	trinotate.v4.0.2.simg	

例えば、HISAT2は「tool/hisat2-2.2.1」以下にインストールされている。

\$ ls tool/hisat2-2.2.1		
AUTHORS	hisat2-build-s	hisat2-repeat
example	hisat2-build-s-debug	hisat2-repeat-debug
extract_exons.py	hisat2_extract_exons.py	hisat2_simulate_reads.py
extract_splice_sites.py	hisat2_extract_snps_haplotypes_UCSC.py	LICENSE
hisat2	hisat2_extract_snps_haplotypes_VCF.py	MANUAL
hisat2-align-l	hisat2_extract_splice_sites.py	MANUAL.markdown
hisat2-align-l-debug	hisat2-inspect	NEWS
hisat2-align-s	hisat2-inspect-l	scripts
hisat2-align-s-debug	hisat2-inspect-l-debug	TUTORIAL
hisat2-build	hisat2-inspect-s	VERSION
hisat2-build-l	hisat2-inspect-s-debug	
hisat2-build-l-debug	hisat2_read_statistics.py	

- ・ 様々なドキュメントやプログラム、スクリプトが存在している。
- ・ ターミナルの設定によっては実行形式のファイルやディレクトリ等が色分けされている。

# 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/hisat2-2.2.1/hisat2 -h ..... -hオプションをつけてHISAT2を実行
HISAT2 version 2.2.1 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).
  <SRA accession number>   Comma-separated list of SRA accession numbers, e.g. --sra-
  acc SRR353653,SRR353654.
  <sam>     File for SAM output (default: stdout)
  : (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（可能性が高い）

# 演習

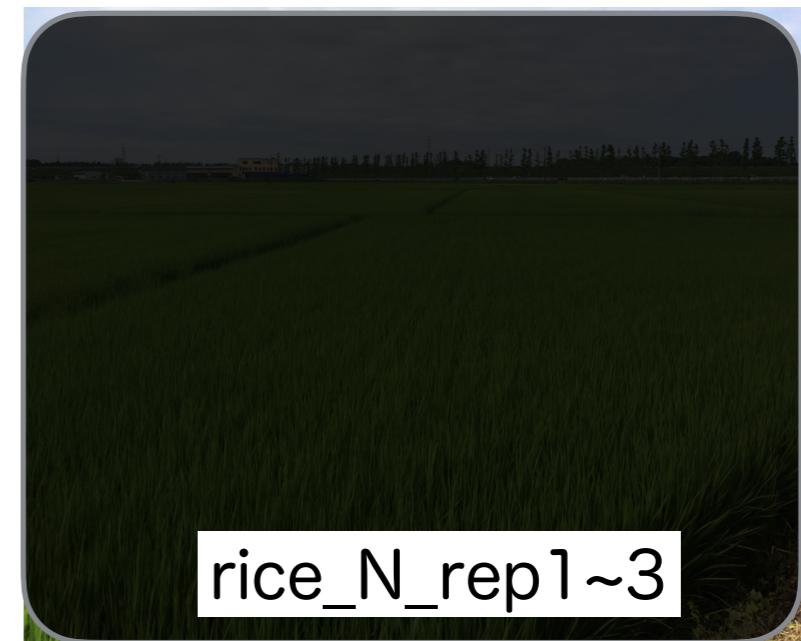
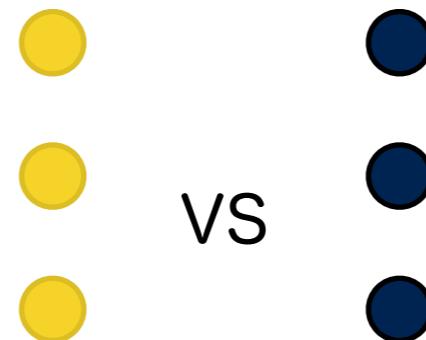
- ▶ リファレンス情報を用いたRNA-Seq解析
- ▶ De novoトランスクリプトーム解析
- ▶ 遺伝子機能アノテーション

# 演習の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



2:00 PM



2:00 AM

- 人工気象器で栽培しているイネ（日本晴）の苗の葉身のサンプル。
- 昼（14 時）と夜（2 時）にサンプリング（2 条件）。
- それぞれ3 個体からサンプリング（3 反復）
- Illumina社のStranded mRNA-Seq法でライブラリ調製。
- Illumina社のHiSeq2000による、100bpのPaired-endシーケンシング。

\*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。

# 使用するデータ

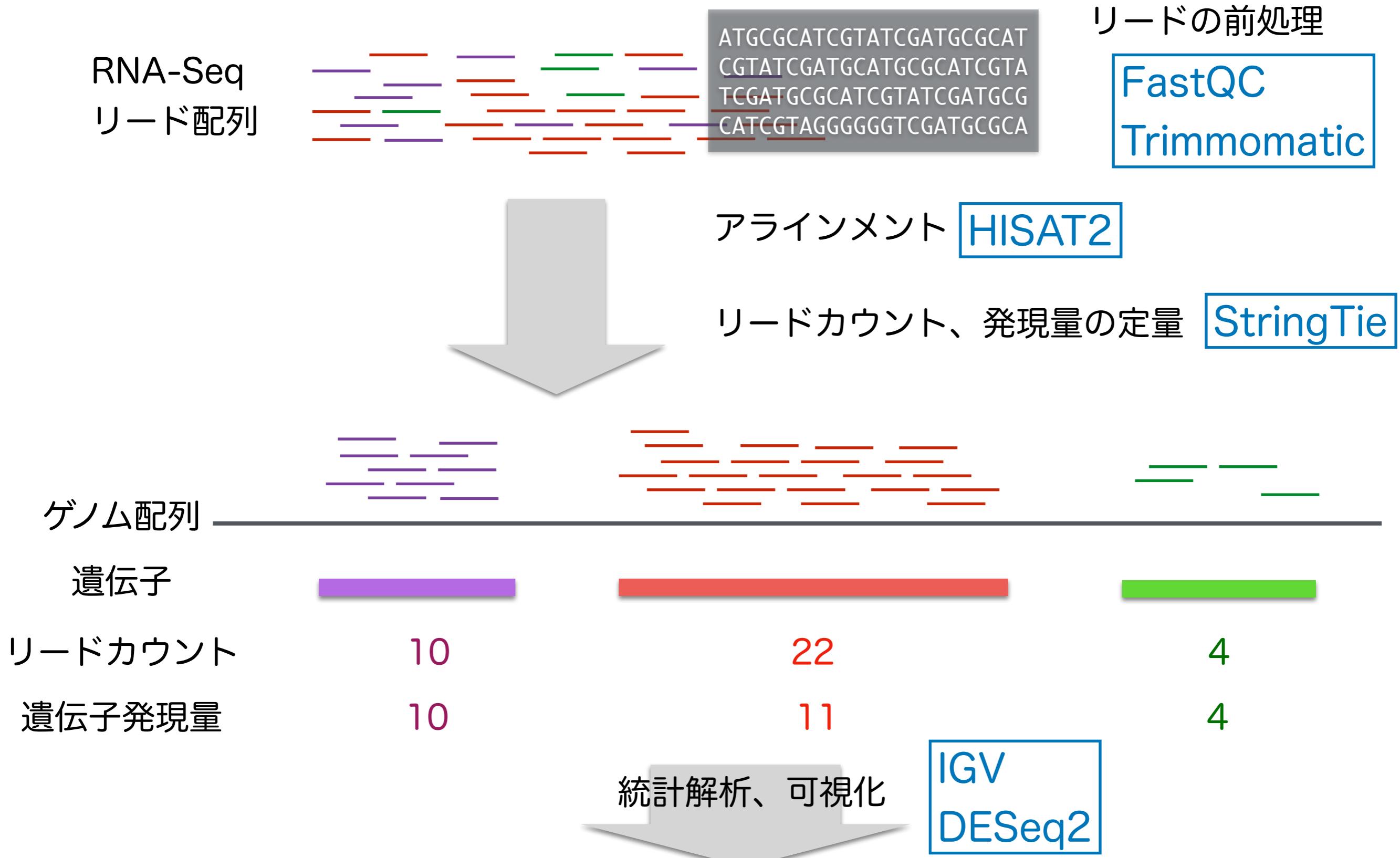
```
$ cd ..... ホームディレクトリに移動  
$ cd RNA-Seq ..... RNA-Seqディレクトリに移動  
$ ls data ..... dataディレクトリの中身を確認  
annotation.gtf ..... rice_D_rep2_r2.org.fastq.gz rice_N_rep2_r1.org.fastq.gz  
genome.fa ..... rice_D_rep3_r1.org.fastq.gz rice_N_rep2_r2.org.fastq.gz  
rice_D_rep1_r1.org.fastq.gz ..... rice_D_rep3_r2.org.fastq.gz 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)
- ・ 既知のイネ遺伝子のアミノ酸配列 (rice\_protein.fa)
- ・ mRNA-Seqリード配列 (\*.fastq.gz)

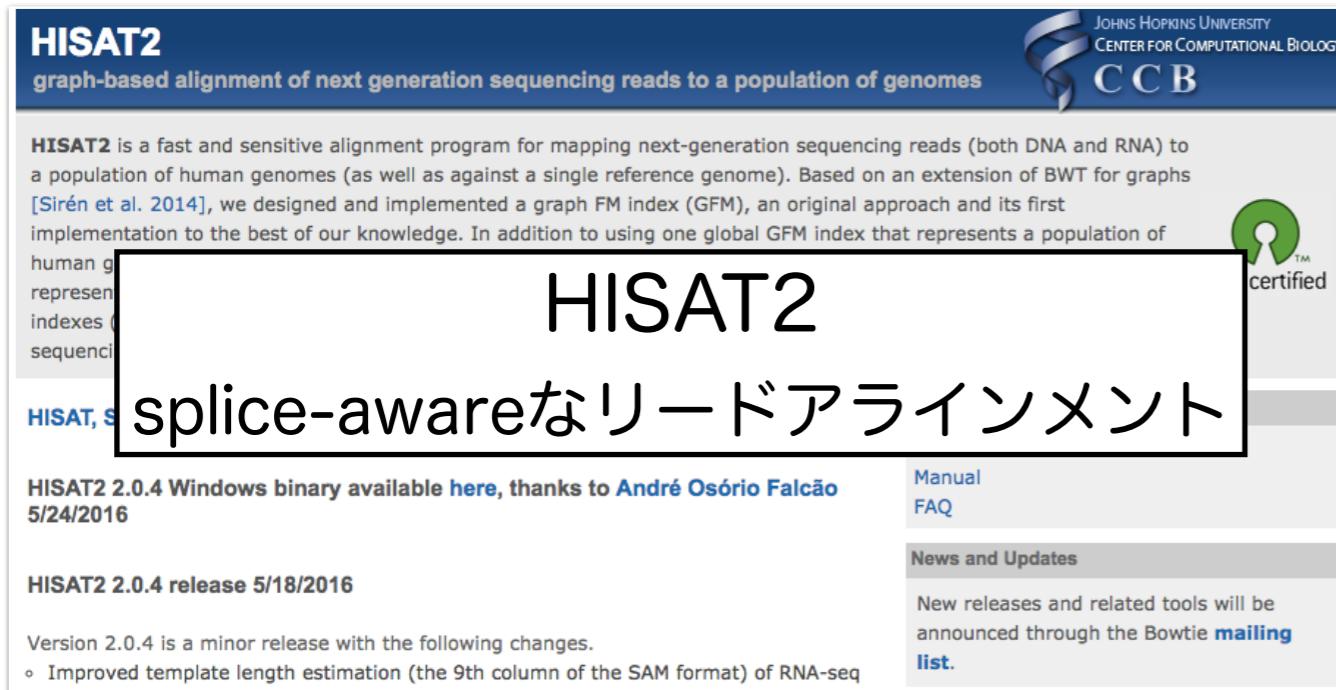


# リファレンス情報を用いたRNA-Seq解析の流れ



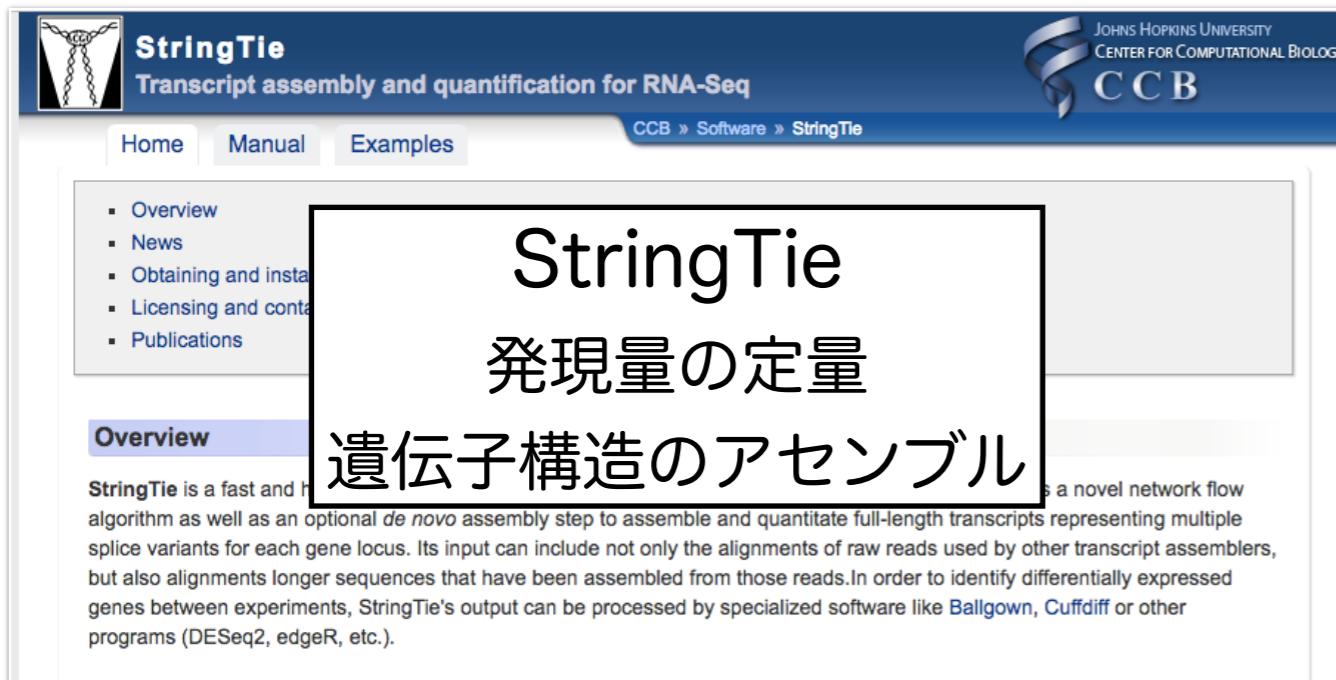
2サンプル間比較によるDifferentially expressed gene (DEG)の検出、結果の可視化など

# リファレンス情報を用いたRNA-Seq解析のためのツール



The screenshot shows the HISAT2 software homepage. At the top, it says "HISAT2 graph-based alignment of next generation sequencing reads to a population of genomes". Below that is the Johns Hopkins University Center for Computational Biology (CCB) logo. A green circular badge indicates "certified". The main title "HISAT2" is prominently displayed, followed by "splice-awareなリードアラインメント". Below this, it says "HISAT2 2.0.4 Windows binary available here, thanks to André Osório Falcão 5/24/2016". It also mentions "HISAT2 2.0.4 release 5/18/2016". A note states: "Version 2.0.4 is a minor release with the following changes." and "Improved template length estimation (the 9th column of the SAM format) of RNA-seq". On the right side, there are links for "Manual", "FAQ", and "News and Updates".

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



The screenshot shows the StringTie software homepage. At the top, it says "StringTie Transcript assembly and quantification for RNA-Seq". Below that is the Johns Hopkins University Center for Computational Biology (CCB) logo. The main title "StringTie" is prominently displayed, followed by "発現量の定量 遺伝子構造のアセンブル". On the left, there's a sidebar with links for "Home", "Manual", "Examples", "Overview", "News", "Obtaining and installing", "Licensing and contacting", and "Publications". The "Overview" section is currently selected. The text in this section describes StringTie as a fast and highly accurate transcript assembler that uses a novel network flow algorithm.

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

## Tuxedo suite tools

(TopHat, Cufflinksの後継プログラム)

Pertea et al. *Nature Protocol* 2016 11(9)

### PROTOCOL

## Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown

Mihaela Pertea<sup>1,2</sup>, Daehwan Kim<sup>1</sup>, Geo M Pertea<sup>1</sup>, Jeffrey T Leek<sup>3</sup> & Steven L Salzberg<sup>1–4</sup>

<sup>1</sup>Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, Maryland, USA. <sup>2</sup>Department of Computer Science, Whiting School of Engineering, Johns Hopkins University, Baltimore, Maryland, USA. <sup>3</sup>Department of Biostatistics, Bloomberg School of Public Health, Johns Hopkins University, Baltimore, Maryland, USA. <sup>4</sup>Department of Biomedical Engineering, Johns Hopkins University, Baltimore, Maryland, USA. Correspondence should be addressed to S.L.S. (salzberg@jhu.edu).

## ballgown

platforms all rank 131 / 1741 post 0 / 0 / in Bioc 5 years  
build ok updated before release dependencies 6 8

DOI: [10.18129/B9.bioc.ballgown](https://doi.org/10.18129/B9.bioc.ballgown) [f](#) [t](#)

Flexible, isoform-level differential expression analysis

Bioconductor version: Rel

Tools for statistical analysis and visualization of transcript s

Author: Jack Fu [aut], Alyssa C. Frazee [aut, cre], Leonardo Collado-Torres [aut], Andrew E. Jaffe [aut], Jeffrey T. Leek [aut, ths]

Maintainer: Jack Fu <jmfp at jhsph.edu>

Citation (from within R, enter `citation("ballgown")`):

Fu J, Frazee AC, Collado-Torres L, Jaffe AE, Leek JT (2019). *ballgown: Flexible, isoform-level differential expression analysis*. R package version 2.16.0.

<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 for each transcript. This allows HISAT2 to align reads to transcripts with high sensitivity even if they have low coverage. The alignment results are stored in a compressed SAM format.

**HISAT2**  
splice-awareなリードアラインメント

HISAT2 2.0.4 Windows binary available [here](#), thanks to André Osório Falcão  
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

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

**StringTie**  
Transcript assembly and quantification for RNA-Seq

JOHNS HOPKINS UNIVERSITY  
CENTER FOR COMPUTATIONAL BIOLOGY  
CCB

Home Manual Examples CCB » Software » StringTie

**StringTie**  
発現量の定量  
遺伝子構造のアセンブル

**Overview**

StringTie is a fast and highly accurate transcript assembler. 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.).

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

発現解析は「DESeq2」を使います



## DESeq2

platforms all rank 25 / 2230 support 198 / 204 in Bioc 10.5 years  
build ok updated since release dependencies 68

DOI: [10.18129/B9.bioc.DESeq2](https://doi.org/10.18129/B9.bioc.DESeq2)

Differential gene expression analysis based on the negative binomial distribution

Bioconductor version: Release

Estimate variance-mean dependence differential expression base

Author: Michael Love [aut, ctb], Wolfgang Huber [aut, ctb], RADIANT EU FP7 [fnd], NIH NHGRI [fnd], CZI [fnd]

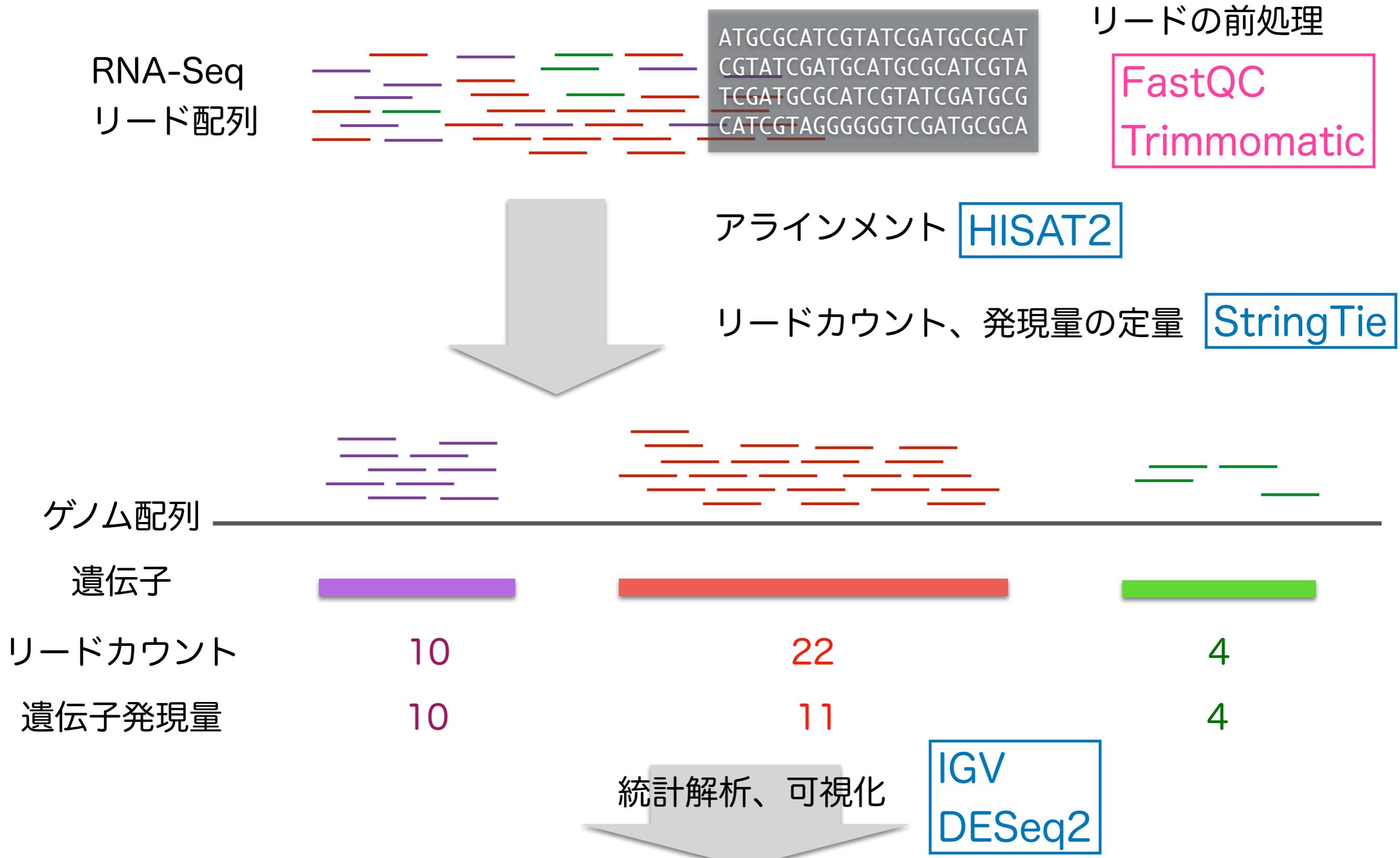
Maintainer: Michael Love <[michaelisaiahlove@gmail.com](mailto:michaelisaiahlove@gmail.com)>

Citation (from within R, enter `citation("DESeq2")`):

Love MI, Huber W, Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." *Genome Biology*, **15**, 550. doi:[10.1186/s13059-014-0550-8](https://doi.org/10.1186/s13059-014-0550-8).

<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>

# リファレンス情報を用いたRNA-Seq解析の流れ



# Step 1. 前処理 - 配列データのQC

質の悪いデータからは良い結果は得られない。

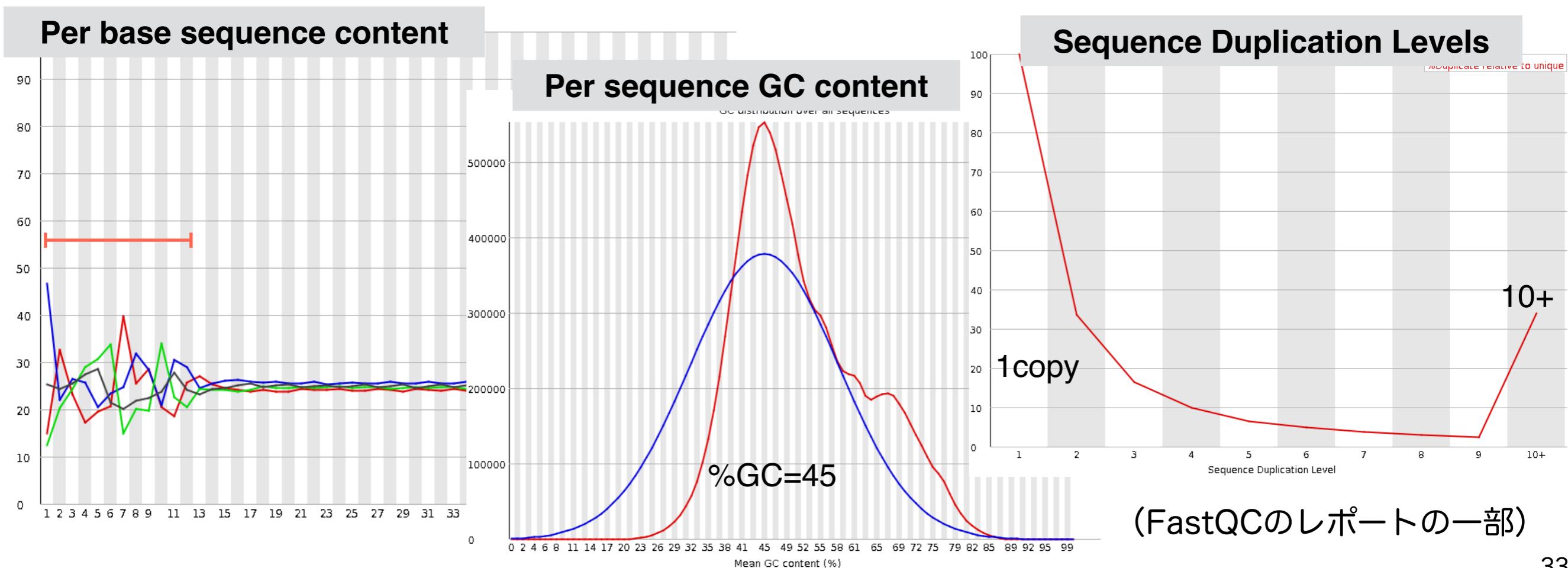
FastQCなどのQCツールを使い、リード数や解読塩基量、クオリティ、塩基出現頻度の偏り、配列長などを事前によくチェックする！

## 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。



# 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、そのアンマッ普リード（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  step4_StringTie-abundance_estimation.sh
step2_HISAT2-build.sh   step5_StringTie_assemble_merge_gffcompare.sh
step3_HISAT2.sh         step6_make_CountMatrix.sh
```

シェルスクリプトが6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
$ bash ./step5_StringTie_assemble_merge_gffcompare.sh
$ bash ./step6_make_count_matrix.sh
```

演習では各ステップの説明をしながら順番に進めていきます。

# Step 1. 前処理 - 配列データのクリーニング

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 -

```
#!/bin/bash
WorkingDir=$HOME/RNA-Seq
DataDir=$WorkingDir/data
ToolDir=$WorkingDir/tool
FastQC_bin=$ToolDir/FastQC
Trimmomatic_bin=$ToolDir/Trimmomatic-0.39
Trimmomatic_program=$Trimmomatic_bin/trimmomatic-0.39.jar

export PATH=$FastQC_bin:$PATH
```

データや各ツールの  
コマンドが置かれている  
ディレクトリを変数に格納

```
CPU=2
echo START: `date`
```

使用するCPU数を変数に格納  
解析の開始時刻を出力

以降の説明では省略します

```
### Step 1: Preprocessing by FastQC and Trimmomatic
FASTQC_OUTDIR_BEFORE=FastQC_before_preprocess
FASTQC_OUTDIR_AFTER=FastQC_after_preprocess
```

FastQCの出力ディレクトリを作成

```
mkdir $FASTQC_OUTDIR_BEFORE $FASTQC_OUTDIR_AFTER
```

# Step 1. 前処理 - 配列データのクリーニング

- step1\_preprocessing.shのつづき -

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。

```
for DATASET in rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3
do
    # FastQC before Trimmomatic
    fastqc \
        --threads $CPU --nogroup --format fastq \
        --outdir $FASTQC_OUTDIR_BEFORE \
        $DataDir/${DATASET}_r*.org.fastq.gz
    # Trimmomatic
    java -Xmx4G -Xms2G -jar $Trimmomatic_program \
        PE -threads $CPU -phred33 -trimlog Trimmomatic_${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
    fastqc \
        --threads $CPU --nogroup --format fastq \
        --outdir $FASTQC_OUTDIR_AFTER \
        ${DATASET}_r*.pe.fastq.gz
done
```

----- 繰り返し開始 -----

----- 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。

前処理前のFastQCの実行  
(r\*でリード1と2を合せて指定)

Trimmomaticの実行

前処理後のFastQCの実行

\ (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。

# Step 1. 前処理 - 配列データのクリーニング

シェルスクリプトの実行

(実行時間：約1分)

```
$ bash ./step1_preprocessing.sh 2>&1 | tee step1_preprocessing.log
```

出力ファイルの確認

```
$ ls
FastQC_after_preprocess      rice_D_rep3_r2.unpe.fastq.gz  step1_preprocessing.log
FastQC_before_preprocess     rice_N_rep1_r1.pe.fastq.gz   step1_preprocessing.sh
rice_D_rep1_r1.pe.fastq.gz    rice_N_rep1_r1.unpe.fastq.gz  step2_HISAT2-build.sh
rice_D_rep1_r1.unpe.fastq.gz  rice_N_rep1_r2.pe.fastq.gz   step3_HISAT2.sh
rice_D_rep1_r2.pe.fastq.gz    rice_N_rep1_r2.unpe.fastq.gz  step4_StringTie-abundance_estimation.sh
rice_D_rep1_r2.unpe.fastq.gz  rice_N_rep2_r1.pe.fastq.gz
step5_StringTie_assemble_merge_gffcompare.sh
rice_D_rep2_r1.pe.fastq.gz    rice_N_rep2_r1.unpe.fastq.gz  step6_make_CountMatrix.sh
rice_D_rep2_r1.unpe.fastq.gz   rice_N_rep2_r2.pe.fastq.gz   Trimmomatic_rice_D_rep1.log
rice_D_rep2_r2.pe.fastq.gz    rice_N_rep2_r2.unpe.fastq.gz  Trimmomatic_rice_D_rep2.log
rice_D_rep2_r2.unpe.fastq.gz  rice_N_rep3_r1.pe.fastq.gz   Trimmomatic_rice_D_rep3.log
rice_D_rep3_r1.pe.fastq.gz    rice_N_rep3_r1.unpe.fastq.gz  Trimmomatic_rice_N_rep1.log
rice_D_rep3_r1.unpe.fastq.gz   rice_N_rep3_r2.pe.fastq.gz   Trimmomatic_rice_N_rep2.log
rice_D_rep3_r2.pe.fastq.gz    rice_N_rep3_r2.unpe.fastq.gz  Trimmomatic_rice_N_rep3.log
```

- 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）

# 前処理結果の確認

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.log
...
Input Read Pairs: 14177689 Both Surviving: 13911912 (98.13%) Forward Only Surviving:
265716 (1.87%) Reverse Only Surviving: 61 (0.00%) Dropped: 0 (0.00%)
...
```

- ・ 前処理の結果、13,911,912 ペア (98.13%) 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

シーケ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

ポリア精製や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。

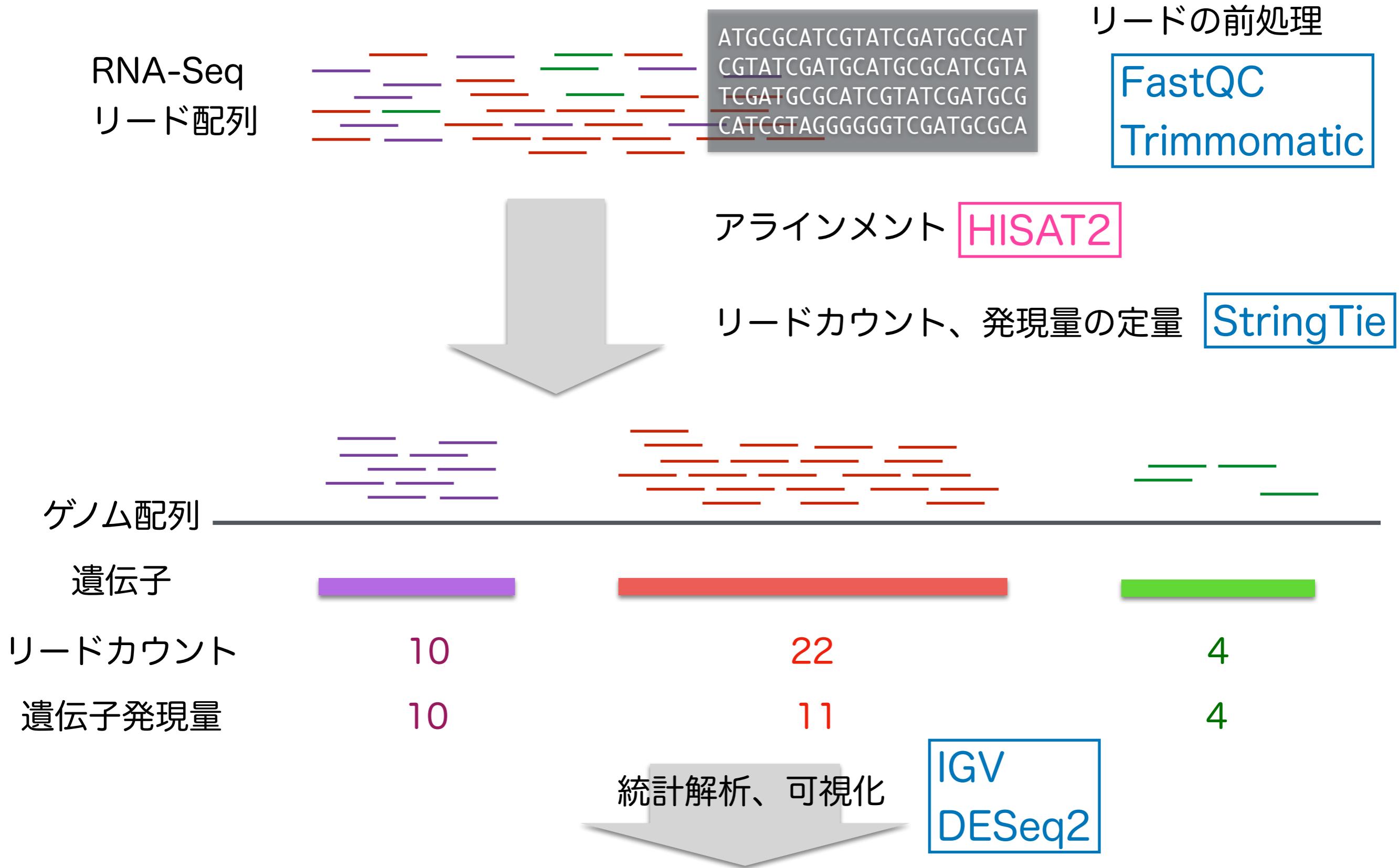
## 4. ゲノム、転写産物配列へのマップ率が低い

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。

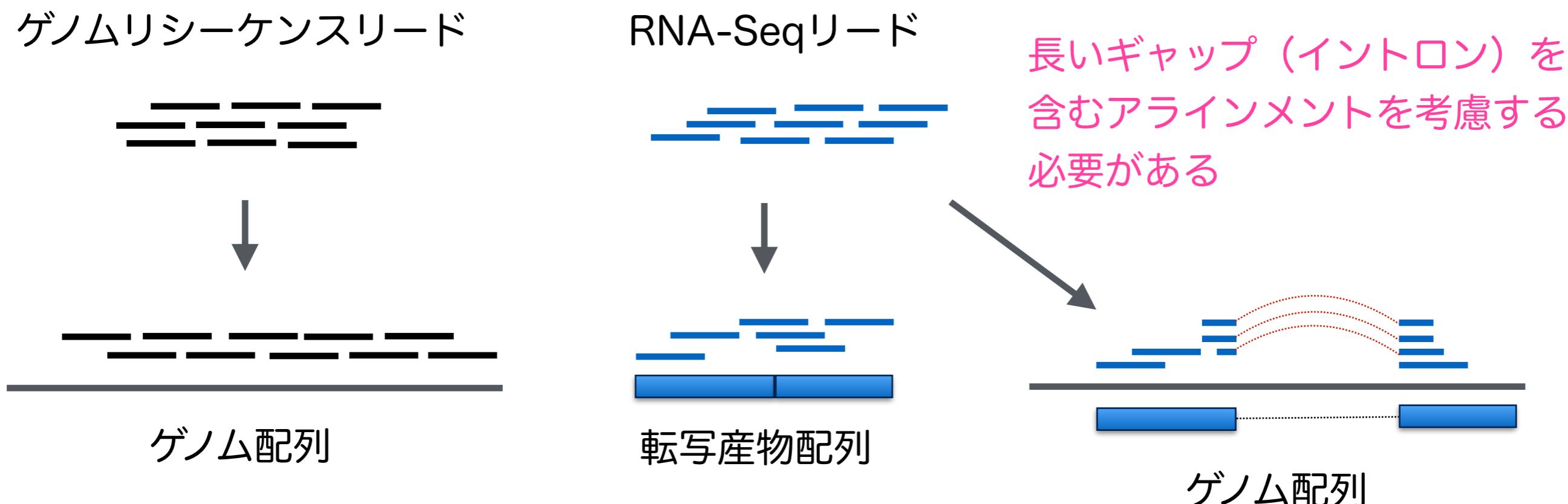
## 5. 有効なリード数が少ない

シーケ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）ペアも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



# RNA-Seqリードのゲノム配列へのアラインメントの難しさ



- ・ ゲノムリシーケンス解析でよく使われるBWAやBowtie2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、STAR2、GSNAPなど)が開発されている。

# 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 -

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data  
ToolDir=$WorkingDir/tool  
HISAT2_bin=$ToolDir/hisat2-2.2.1  
Samtools_bin=$ToolDir/samtools-1.18/bin
```

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 PATH=$HISAT2_bin:$Samtools_bin:$PATH .....
```

exportコマンドで各ツールの  
ディレクトリをPATHに追加。

```
### Step 2: 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
```

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

```
python $HISAT2_bin/hisat2_extract_exons.py \  
$DataDir/annotation.gtf > exon.tab
```

# Step 2. HISAT2のためのインデックスの作成

- step2\_HISAT2-build.sh のつづき -

```
# build index for HISAT2
hisat2-build -p $CPU \
--ss ss.tab \
--exon exon.tab \
$DataDir/genome.fa \
genome
```



HISAT2のための  
インデックスの作成

```
# build index for IGV etc.
samtools faidx $DataDir/genome.fa
```

..... IGVによる可視化に必要な  
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 2. 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12分)

```
$ bash ./step2_HISAT2-build.sh 2>&1 | tee step2_HISAT2-build.log
```

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ファイル
- genome.fa.faiは、IGVなどでの可視化に必要なインデックスファイル

インデックス作成は比較的メモリを多く使用するプロセスのため、ゲノムサイズが大きいデータの場合は特に注意が必要です。

イネゲノム（400Mb）のインデックス作成に20GBほどのメモリを使用します。

# 遺伝子アノテーションファイル (GTF/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, Documents, Publications, and Links. Below the navigation bar, there is a search bar with 'Keywords' and 'Search' buttons, and an 'Advanced' button. The main content area is titled 'Annotation data on Os-Nipponbare-Reference-IRGSP-1.0'. Under the 'Genome sequences' section, there are links for genome assemblies (12 chromosomes) and unanchored sequences, both with download options. A note says sequences are masked by Censor with MIPS and MSU repeat data. The 'Gene set (genes supported by FL-cDNAs, ESTs or proteins)' section is highlighted with a pink border and contains links for gene structure and function information in GFF format, gene structure (only exon) information in GTF format, gene annotation information in tab-delimited text format, and gene sequences in FASTA format.

## 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ウェブサイト

- Illumina社 iGenomes  
[https://support.illumina.com/sequencing/sequencing\\_software/igenome.html](https://support.illumina.com/sequencing/sequencing_software/igenome.html)
- EnsemblPlants  
<https://plants.ensembl.org/index.html>
- Phytozome  
<https://phytozome-next.jgi.doe.gov>

# 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 "0s02g0724000"; transcript_id "0s02t0724000-01"; gene_name "DTH2"; note
"CONSTANS-like protein, Heading promotion under long-day condition";
chr02    irgsp1_rep    exon        30094300      30094498      .      +
gene_id "0s02g0724000"; transcript_id "0s02t0724000-01";
chr02    irgsp1_rep    exon        30096198      30096893      .      +
gene_id "0s02g0724000"; transcript_id "0s02t0724000-01";
chr02    irgsp1_rep    exon        30097390      30097590      .      +
gene_id "0s02g0724000"; transcript_id "0s02t0724000-01";
chr02    irgsp1_rep    exon        30097861      30098165      .      +
gene_id "0s02g0724000"; transcript_id "0s02t0724000-01";
chr02    irgsp1_rep    exon        30098451      30099072      .      +
gene_id "0s02g0724000"; transcript_id "0s02t0724000-01";
...
.
```

- ・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 -

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data  
ToolDir=$WorkingDir/tool  
  
HISAT2_bin=$ToolDir/hisat2-2.2.1  
Samtools_bin=$ToolDir/samtools-1.18  
  
export PATH=$HISAT2_bin:$Samtools_bin:$PATH
```

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。

# Step 3. HISAT2によるRNA-Seqリードのアラインメント

- step3\_HISAT2.sh のつづき -

共通パラメータ

```
### Step 3: Align reads to the reference genome by HISAT2
```

```
HISAT2_COMMON_PARAM="--threads $CPU --min-intronlen 20 --max-intronlen 10000 --dta --rna-strandness RF -x genome --new-summary"
```

```
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 \  
        --summary-file ${DATASET}_summary.txt \  
        -1 ${DATASET}_r1.pe.fastq.gz \  
        -2 ${DATASET}_r2.pe.fastq.gz \  
        -S ${DATASET}.sam
```

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

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

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

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

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

```
done
```

# 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 2>&1 | tee step3_HISAT2.log
```

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やsummaryファイルに書き出される。

```
$ less rice_D_rep1_summary.txt
HISAT2 summary stats:
Total pairs: 13911912
    Aligned concordantly or discordantly 0 time: 732549 (5.27%)
    Aligned concordantly 1 time: 12733707 (91.53%)
    Aligned concordantly >1 times: 294213 (2.11%)
    Aligned discordantly 1 time: 151443 (1.09%)
Total unpaired reads: 1465098
    Aligned 0 time: 817284 (55.78%)
    Aligned 1 time: 622596 (42.50%)
    Aligned >1 times: 25218 (1.72%)
Overall alignment rate: 97.06%
```

\*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13,911,912リードのうち、97.06%がアラインメントされている。
- ・ アラインメント率はRNA-Seqリードやリファレンスゲノム配列のクオリティや、サンプルとリファレンス品種の進化的距離などによって変わる。
- ・ 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%台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リードのアラインメント結果の確認

samtoolsを使い、個々のリードのアラインメント情報を見ることができる。

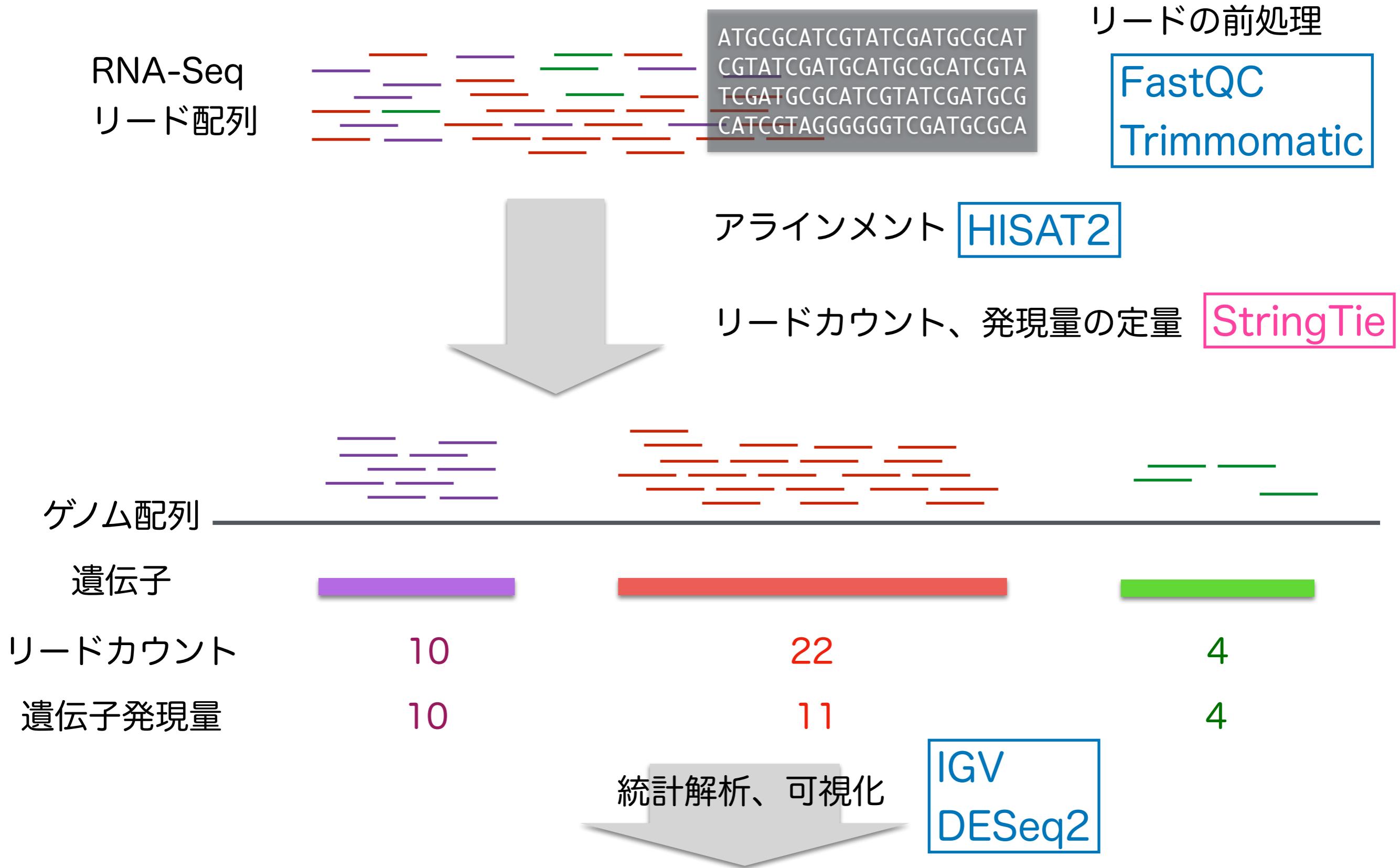
```
$ ./tool/samtools-1.18/bin/samtools view ./rice_D_rep1.bam | less
```

```
FCC3M5DACXX:1:2116:5852:43523# 147 chr01 6747903 0 100M= 6747847 -156
TGCTGAACCGCTACATCCCCTGCGCCCTCCGCAAGAAGTGGGACAACCTGGAAGAGCGCCTGGTTCTACACTCCCCGCGCCGACGAAGCGC
GCCTCCGACT
ccbba]Tccbcc`ccccccccccccccacccb ehhgfcXhgfeiggg
ec_Heec_ba AS:i:-18 ZS:i:-18 XN:
YS:i:0 YT:Z:CP XS:A:- NH:i:10 Z:44G12A17T24
FCC3M5DACXX:1:1115:13355:72883# 22873917 0
CCGATCTCCTGTACCCTCTCGATTATGCAAGAT CTGGAAATAATGGCG
CAGTTCCCTC
accckccc`b`Waca_c`^cccccbccccc`_a^Zcbeegdd`ddgbedgbagffffbdbeau eWdhged^[Jfiihghhdhihhgcg
ggeeeeec_` AS:i:-12 ZS:i:-12 XN:i:0 XM:i:0 X0:i:0 XG:i:0 M:i:0 MD:Z:94 YT:Z:UP
XS:A:- NH:i:10
FCC3M5DACXX:1:1104:5235:86490# 129 chr01 7719251 60 77M121N23M chr02 31248156 0
CTTGATGTACGAGTAGATGGAACACCAAGCTATGGCAATGCAAGTACCAATTCCAGTCTGTGGTAATTCTGTTGCCAAAAACAATGAT
GGAGAATCCA ___c^^c`begcefagg[hg_efiifhfhd़aeeffe[^^g`af^caafhbdeb]
[\bbc\ew\_M]bccgf`^b`dbdbdeZWZZ`bbc_babcb]`b AS:i:0 XN:i:0 XM:i:0 X0:i:0 XG:i:0
NM:i:0 MD:Z:100 YS:i:-2 YT:Z:DP XS:A:+ NH:i:1
```

CIGAR 77M 121N 23M  
リード \_\_\_\_\_  
ゲノム配列 \_\_\_\_\_

- ・ イントロンをまたぐリードアラインメントは「77M121N23M」のように、イントロン部分が”N”で表されている。121bpのイントロンを挟むアラインメントであるという意味。
- ・ SAMフォーマットについては<http://samtools.github.io/hts-specs/SAMv1.pdf> を参照。

# リファレンス情報を用いたRNA-Seq解析の流れ



# 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。

1遺伝子座に複数のスプラシングバリエントがある場合、バリエントごとに発現量の定量が可能。

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

- step4\_StringTie-abundance\_estimation.sh -

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data  
ToolDir=$WorkingDir/tool  
StringTie_bin=$ToolDir/stringtie-2.2.1.Linux_x86_64  
export PATH=$StringTie_bin:$PATH
```

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。

```
### Step 4: Estimate transcript abundances
```

```
STRINGTIE_COMMON_PARAM="-e -B -p $CPU -G $DataDir/annotation.gtf" ..... 共通パラメータ
```

```
OUTDIR=stringtie ..... 結果の出力ディレクトリ
```

```
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 \  
        -o ${OUTDIR}/${DATASET}/${DATASET}.gtf \  
        -A ${OUTDIR}/${DATASET}/${DATASET}.abundance.txt \  
        ${DATASET}.bam
```

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。

```
done
```

# 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 2>&1 | tee step4_StringTie-abundance_estimation.log
```

サンプルごとの発現量やCoverageデータの確認

```
$ ls stringtie/rice_D_rep1/
e2t.ctab      i2t.ctab      rice_D_rep1.abundance.txt  t_data.ctab
e_data.ctab   i_data.ctab   rice_D_rep1.gtf
```

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る。

- rice\_\*.gtf : 遺伝子構造と遺伝子発現量
- rice\_\*.abundance.txt : 遺伝子座ごとの発現量等の情報
- 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

遺伝子座ごとの発現量等の情報 (rice\_\*.abundance.txt) の確認

```
$ less stringtie/rice_D_rep1/rice_D_rep1.abundance.txt
```

Gene ID	Gene Name	Reference	Strand	Start	End	Coverage	FPKM	TPM
...								
Os02g0723200 13.934043	OsGT7 22.703176	chr02	+	30051750	30053275		0.271953	
Os02g0723300 760.321838	- 1238.816406	chr02	-	30053886	30058536		14.839326	
Os02g0722800 14.469565	OsWD40-53 775.483032	chr02	-	30015998	30025935			
Os02g0723400 319.038116	IAA8 519.818848	chr02	+	30064358	30066172		4.491709	
Os02g0723700 794.916687	- 1295.182861	chr02	+	30072166	30074833		15.514519	
Os02g0724000 314.994568	DTH2 513.230530	chr02	+	30094300	30099072		6.147800	
Os02g0724300 15.696391	- 25.574623	chr02	-	30102045	30102837		0.243380	
...								

- 1遺伝子座ごとに遺伝子ID、遺伝子名、座標情報、発現量 (FPKMとTPM) が記載されている。
- 点線-----は遺伝子座の区切り。

# 遺伝子発現量の指標 (RPKM/FPKM/TPM)

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／総リード数(M)／遺伝子の長さ(kb)

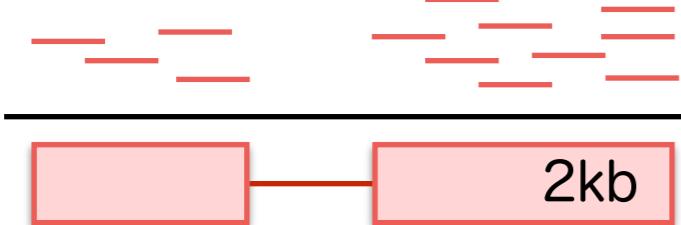
RPKM: reads per kilobase of exon model per million mapped reads

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



条件2

18リード



14リード



$$\text{RPKM: } 13/2/19=0.34$$

$$6/1/19=0.32$$

$$18/2/32=0.28$$

$$14/1/32=0.44$$

- 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。

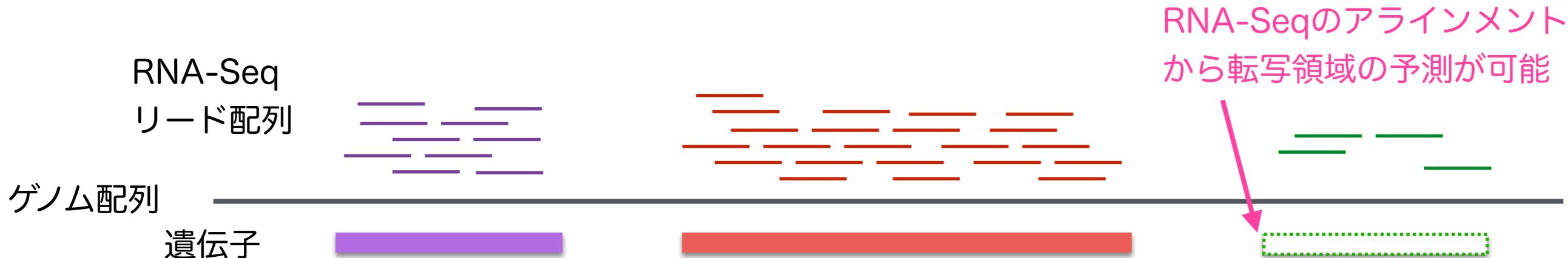
# 遺伝子発現量の指標 (RPKM/FPKM/TPM)

	RPKM/FPKM	TPM
Step.1	総リード数を補正 (100万リード換算)	遺伝子の長さを補正 (1kb換算)
Step.2	遺伝子の長さを補正 (1kb換算)	総リード数を補正 (100万リード換算)

- 全遺伝子のTPM(transcripts per million)の総和はサンプル間で等しいので、個々の遺伝子のTPM値はそのままでもサンプル間で比較がしやすい。
- 最近はTPMを使った比較が勧められている。

- Wagner et al. (2012) Theory in Biosciences
- <https://www.rna-seqblog.com/rpkf-fpkf-and-tpm-clearly-explained/>

# Step 5. StringTieによる遺伝子構造予測



StringTieによってRNA-Seqリードのアラインメント情報から転写領域を予測するスクリプト

```
$ less step5_StringTie_assemble_merge_gffcompare.sh
```

- step5\_StringTie\_assemble\_merge\_gffcompare.sh -

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data  
ToolDir=$WorkingDir/tool  
StringTie_bin=$ToolDir/stringtie-2.2.1.Linux_x86_64  
gffcompare_bin=$ToolDir/gffcompare-0.12.6.Linux_x86_64  
gffread_bin=$ToolDir/gffread-0.12.7.Linux_x86_64
```

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 PATH=$StringTie_bin:$gffcompare_bin:$gffread_bin:$PATH
```

exportコマンドで各ツールの  
ディレクトリをPATHに追加。

# Step 5. StringTieによる遺伝子構造予測

- step5\_StringTie\_assemble\_merge\_gffcompare.sh のつづき -

```
### Step 5-1: Assemble transcript structures by StringTie
STRINGTIE_COMMON_PARAM="-p $CPU" ..... 共通パラメータ

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 \
        -o ${DATASET}.gtf \
        -l ${DATASET} \
        ${DATASET}.bam
done

### Step 5-2: Merge assembled transcript structures by StringTie
# make the list of assembled transcripts files

ls ./rice_*.gtf > assemblies.txt ..... 各サンプルの遺伝子予測結果ファイルのリストを作成

# merge assembled transcripts
stringtie --merge \
    -G $DataDir/annotation.gtf \
    -o stringtie_merged.gtf \
    assemblies.txt
```

各サンプルごとに、アラインメントデータを指定して遺伝子構造を予測

全サンプルの遺伝子予測結果 (assemblies.txt) と既知遺伝子情報 (annotation.gtf) を合わせて、遺伝子構造を予測する。

# Step 5. StringTieによる遺伝子構造予測

- step5\_StringTie\_assemble\_merge\_gffcompare.sh のつづき -

```
### Step 5-3: Compare assembled transcripts with the reference annotation
```

```
gffcompare \  
-r $DataDir/annotation.gtf \  
-o gffcmp \  
stringtie_merged.gtf
```

既知遺伝子情報 (annotation.gtf) と遺伝子予測の結果 (stringtie\_merged.gtf) を比較し、様々な統計情報を出力する。

```
### Step 5-4: Make transcript sequence file
```

```
gffread \  
-g $DataDir/genome.fa \  
-w stringtie_merged_transcript.fa \  
stringtie_merged.gtf
```

全遺伝子（予測遺伝子も含む）の転写産物配列を  
FASTA形式で出力

- ・ 予測遺伝子情報 (stringtie\_merged.gtf) を指定して、Step 4をやり直せば全遺伝子（予測遺伝子も含む）についての発現データを得ることができる。
- ・ 得られた転写産物配列データを用いて、タンパク質コード領域を予測したり、機能アノテーションをつけたりすることができる（後ほど紹介します）。

# Step 5. StringTieによる遺伝子構造予測

シェルスクリプトの実行

(実行時間：約10秒)

```
$ bash ./step5_StringTie_assemble_merge_gffcompare.sh 2>&1 | tee  
step5_StringTie_assemble_merge_gffcompare.log
```

gffcompareの結果ファイル

出力ファイルの詳細は、<http://ccb.jhu.edu/software/stringtie/gffcompare.shtml> を参照

```
$ ls gffcmp*  
gffcmp.annotated.gtf  gffcmp.stringtie_merged.gtf.refmap  
gffcmp.loci           gffcmp.stringtie_merged.gtf.tmap  
gffcmp.stats          gffcmp.tracking
```

既知遺伝子と予測遺伝子の対応関係 (gffcmp.stringtie\_merged.gtf.tmap)

```
$ less gffcmp.stringtie_merged.gtf.tmap  
ref_gene_id  ref_id  class_code  qry_gene_id  qry_id  num_exons  FPKM  
TPM          cov     len         major_iso_id  ref_match_len  
0s01g0100100 0s01t0100100-01 j      MSTRG.1  MSTRG.1.1    12  0.000000  
0.000000    0.000000  3024       MSTRG.1.1    2935  
0s01g0100100 0s01t0100100-01 =      MSTRG.1  0s01t0100100-01 12  0.000000  
0.000000    0.000000  2935       MSTRG.1.1    2935  
0s01g0100400 0s01t0100400-01 k      MSTRG.2  MSTRG.2.1    12  0.000000  
0.000000    0.000000  5102       MSTRG.2.3    2161  
...
```

# Step 5. StringTieによる遺伝子構造予測

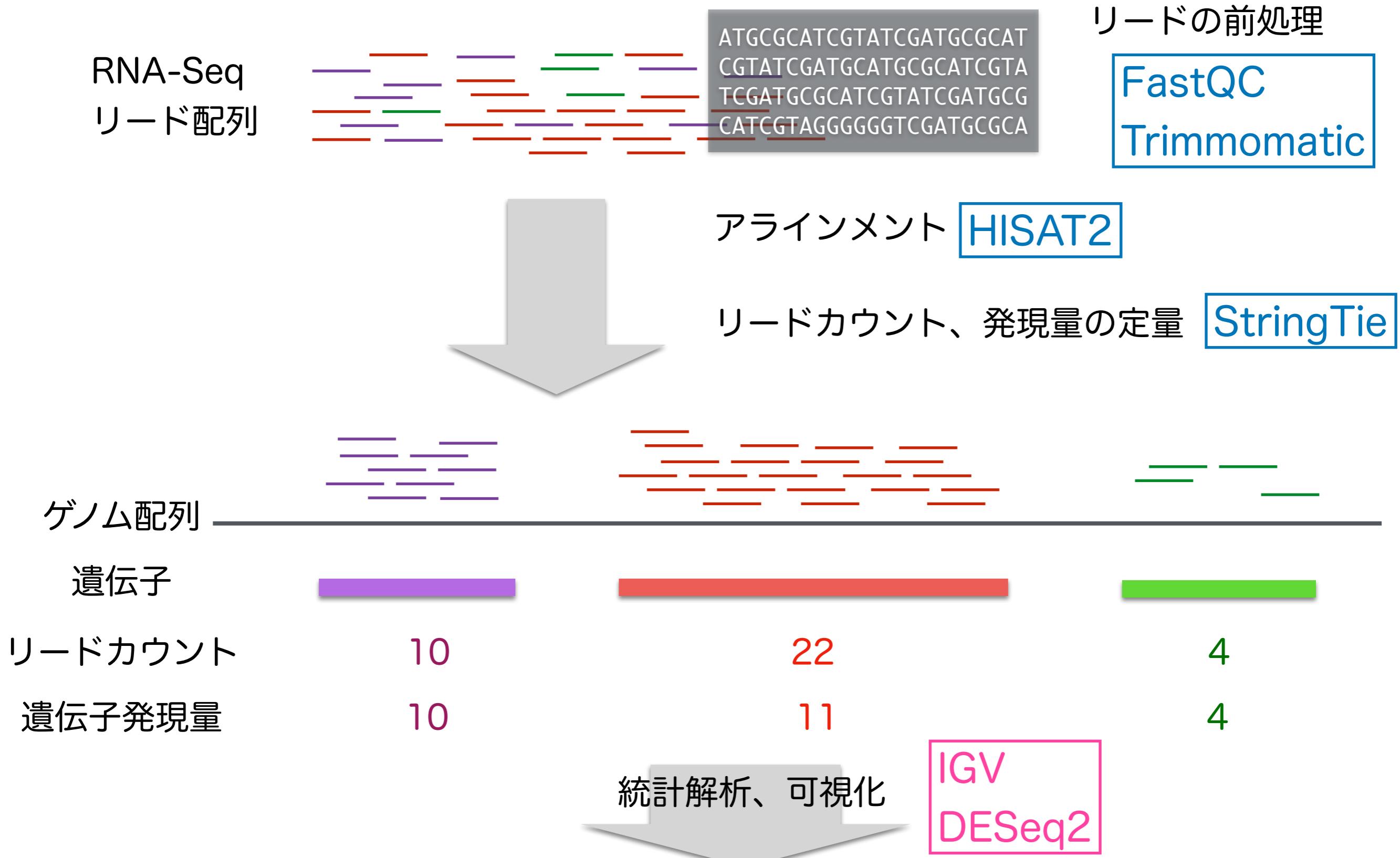
既知遺伝子と予測遺伝子の比較結果の統計情報 (gffcmp.stats)

```
$ less gffcmp.stats
...
#= Summary for dataset: stringtie_merged.gtf
#   Query mRNAs : 78245 in 46780 loci (62240 multi-exon transcripts)
#                   (13882 multi-transcript loci, ~1.7 transcripts per locus)
# Reference mRNAs : 43611 in 37866 loci (30947 multi-exon)
# Super-loci w/ reference transcripts: 37049
#-----| Sensitivity | Precision |
#       Base level: 100.0    | 83.3    |
#       Exon level: 96.8     | 71.5    |
#       Intron level: 100.0    | 76.3    |
# Intron chain level: 100.0    | 49.7    |
# Transcript level: 98.7     | 55.0    |
# Locus level: 99.1      | 78.6    |

# Matching intron chains: 30947
# Matching transcripts: 43062
# Matching loci: 37531

# Missed exons: 0/173381 ( 0.0%)
# Novel exons: 34665/241669 ( 14.3%)
# Missed introns: 23/126061 ( 0.0%)
# Novel introns: 26402/165286 ( 16.0%)
# Missed loci: 0/37866 ( 0.0%)
# Novel loci: 9731/46780 ( 20.8%)
```

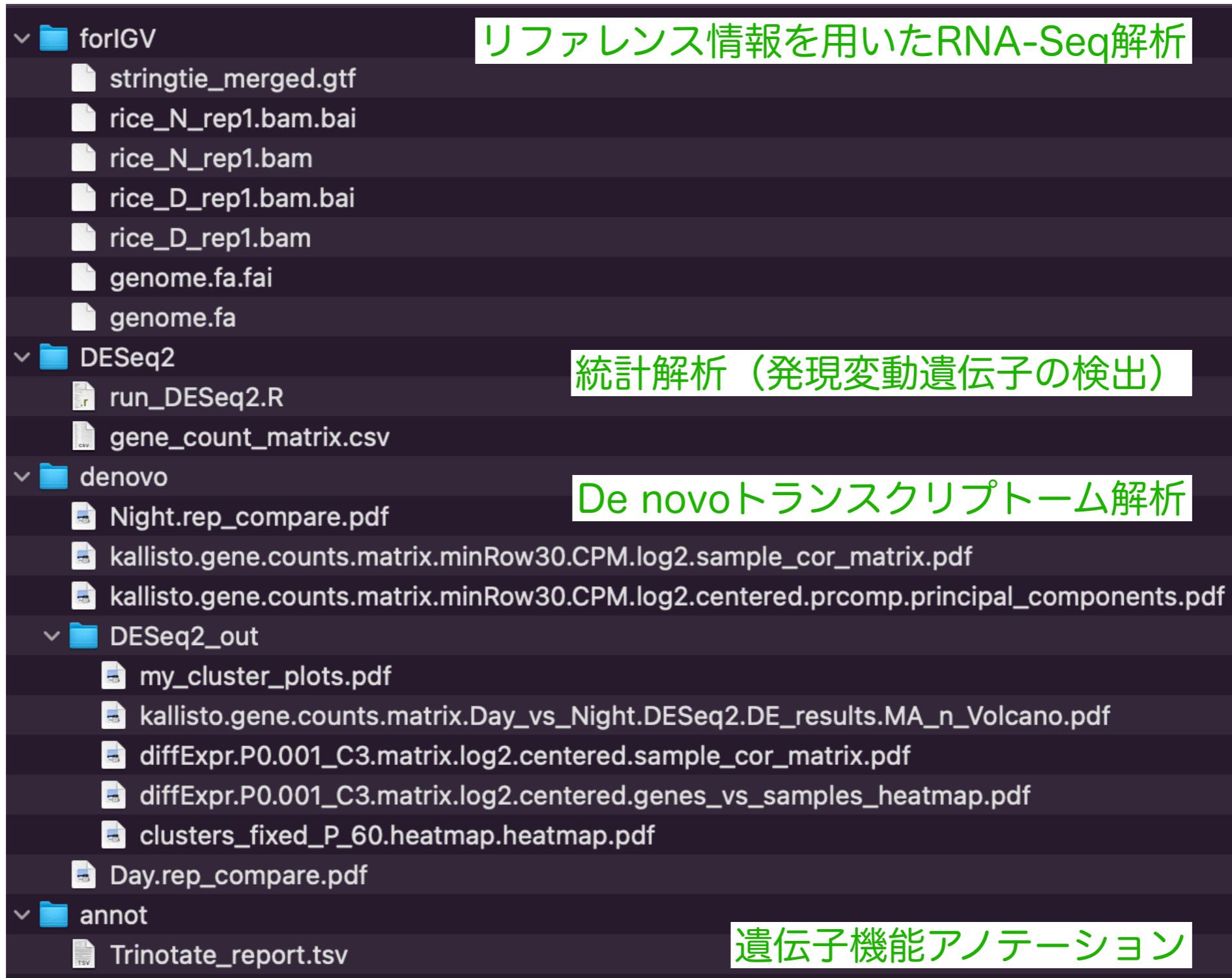
# リファレンス情報を用いたRNA-Seq解析の流れ



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 64

# 統計解析や可視化のためのサンプルデータ

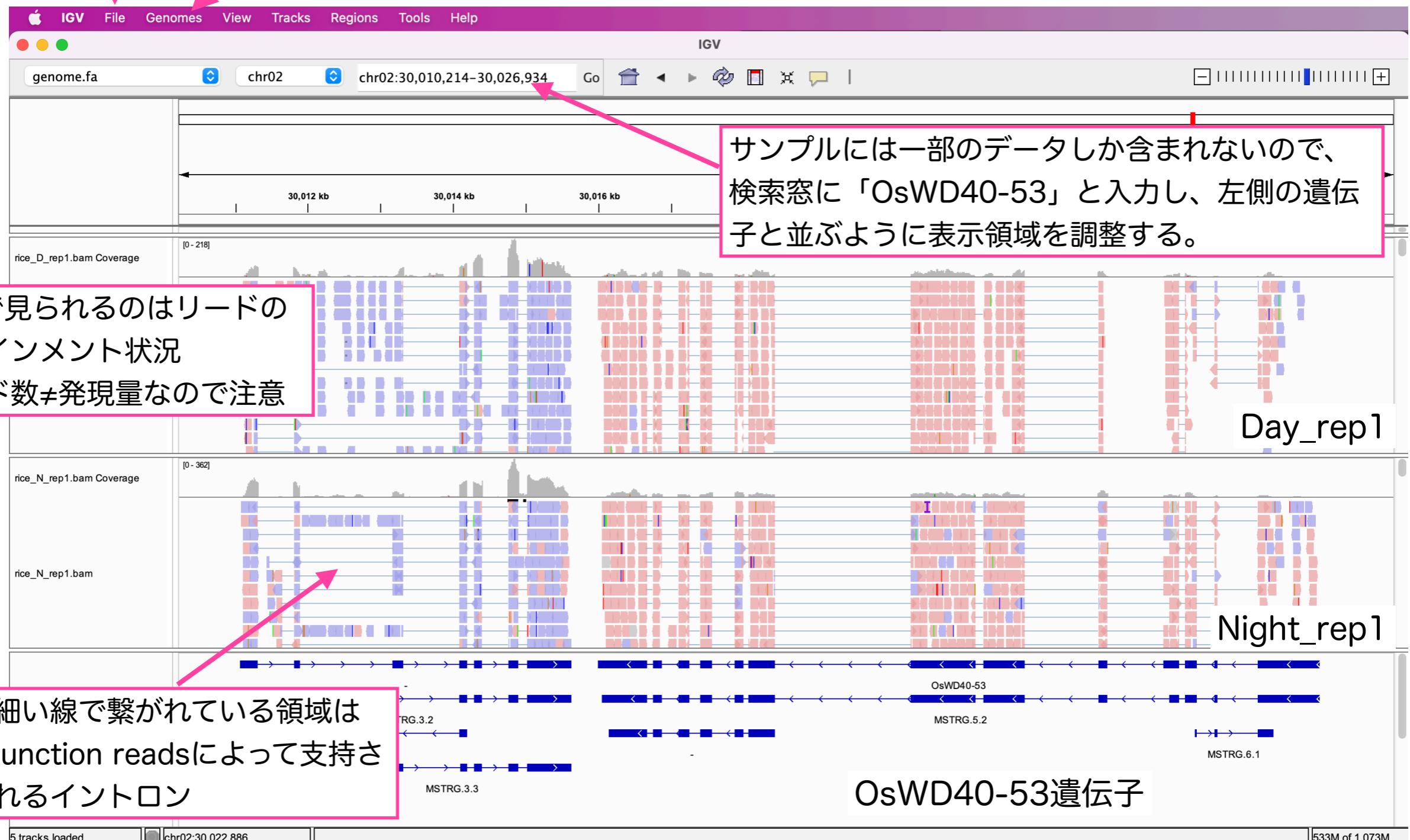
皆さんのデスクトップ上の「RNA-Seq」ディレクトリ中には、統計解析や可視化のためのサンプルデータや、可視化した結果の画像ファイルを置いてあります。



# IGVを用いたRNA-Seqデータの可視化

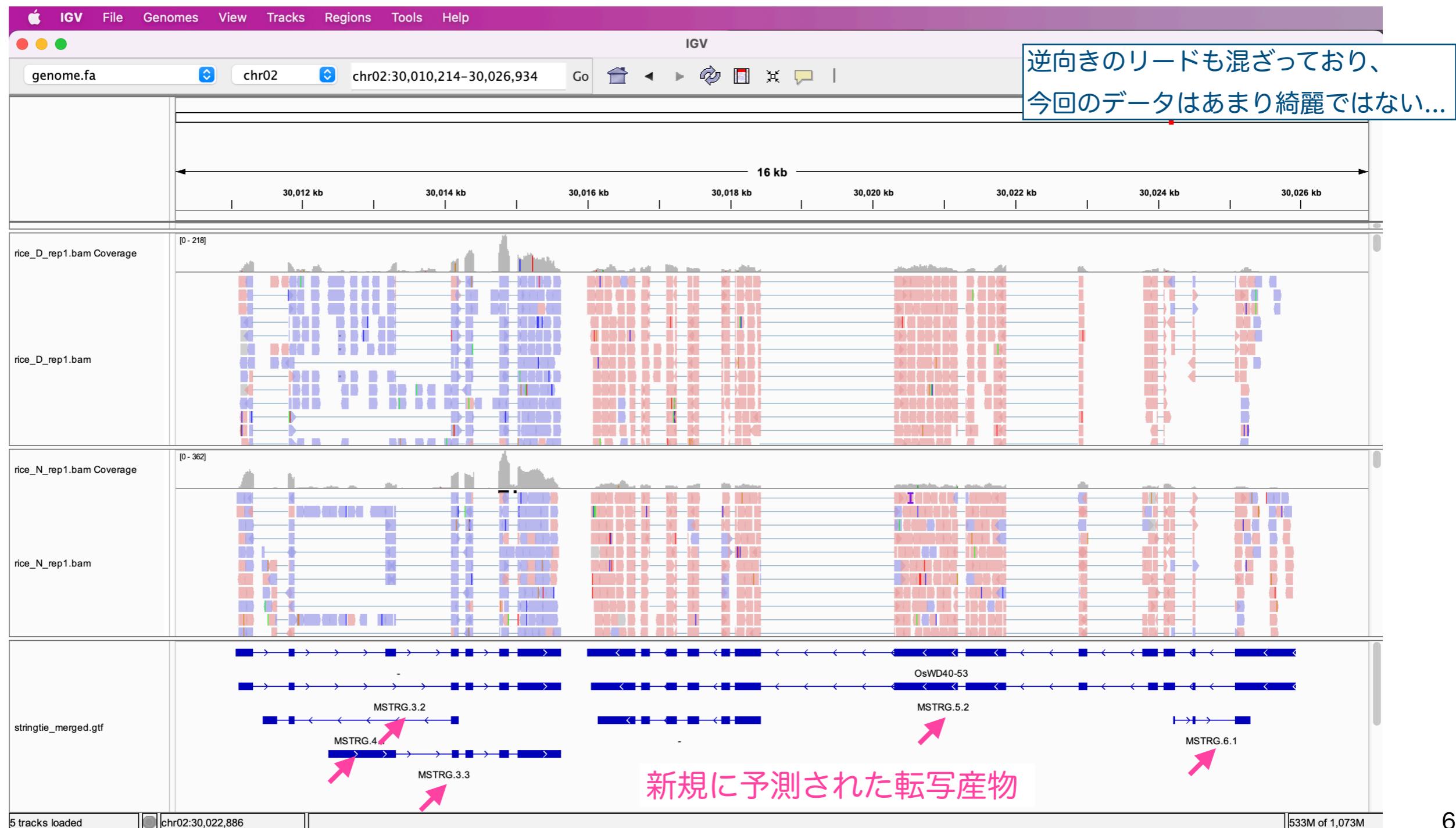
デスクトップ上の「RNA-Seq/forIGV/」ディレクトリ中のリファレンスゲノム (FASTA) と予測遺伝子を含む遺伝子アノテーション (GTF) 、RNA-Seqリードのアライメント (BAM) を読み込む。

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



# IGVによるRNA-Seqデータのアラインメント、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。アライン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、転写の向きが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。

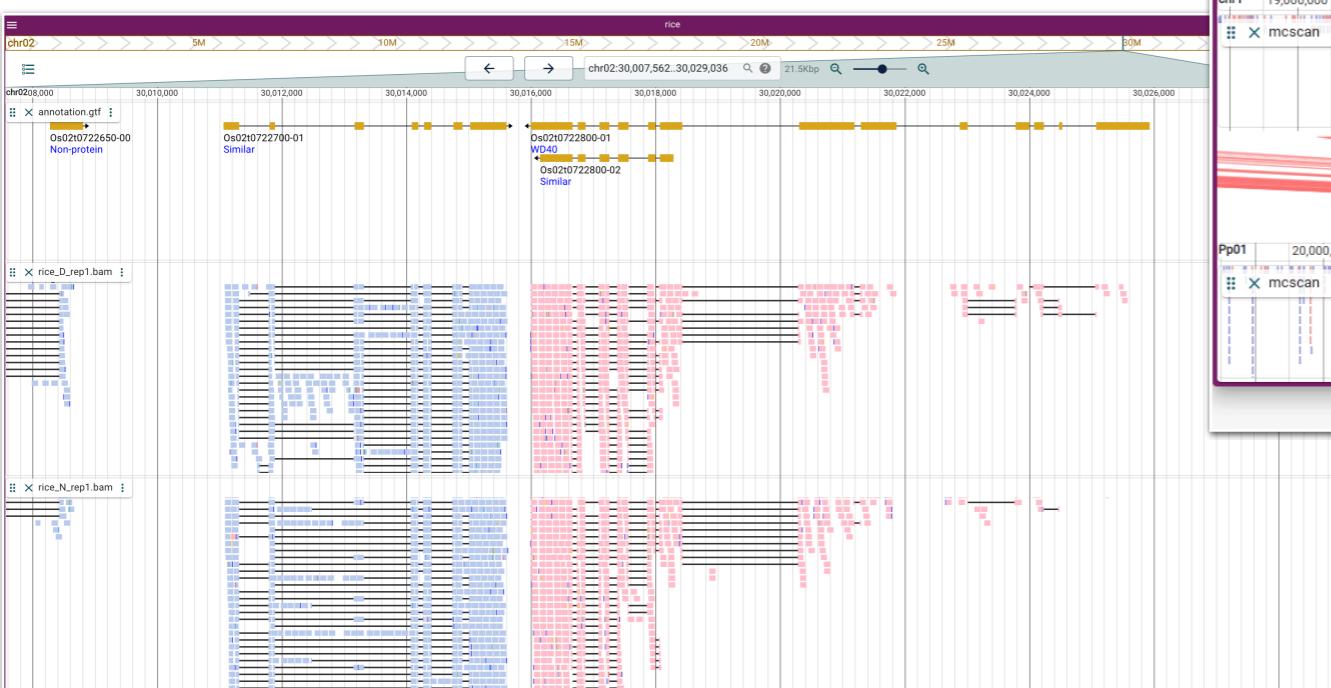


# Desktop版のJBrowse 2も様々なデータの可視化に便利

The screenshot shows the JBrowse website. At the top, there's a navigation bar with links for JBrowse, Docs, Blog, Download, Plugins, Features, Gallery, Demos, Contact, and a search bar. Below the navigation is a section titled "JBrowse" with the subtitle "The next-generation genome browser". It includes a brief description, download links for "DOWNLOAD" and "BROWSE DEMO", and links to latest release blogpost, embedded components, and command line tools. A large screenshot of the genome browser interface is displayed, showing a chromosome track for rice with various genomic features like genes and repeats.

## Features

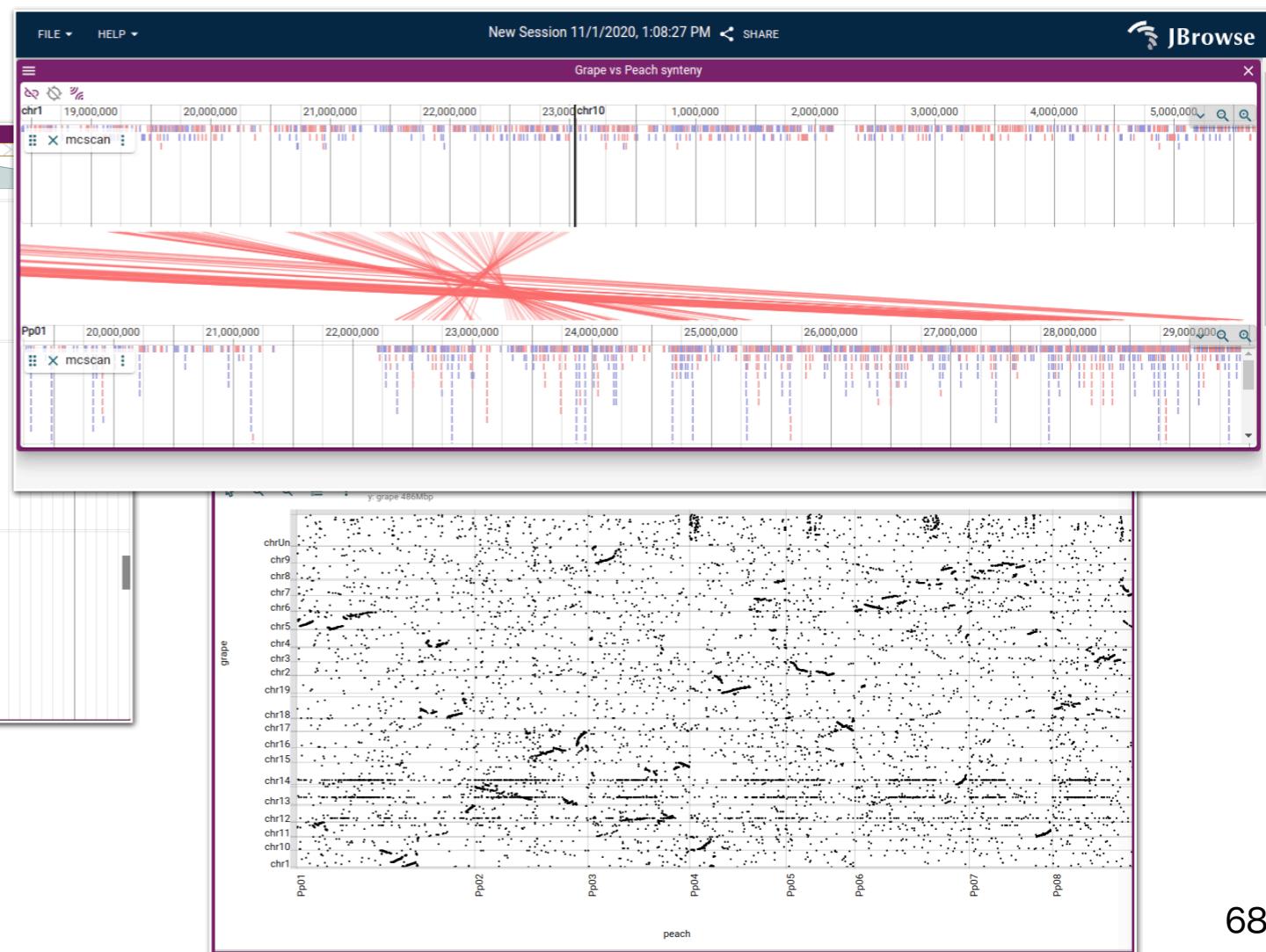
- Improved structural variant and comparative genomics visualization with linear, circular, dotplot, and synteny views
- Support for many common data types including BAM, CRAM, tabix indexed VCF, GFF, BED, BigBed, BigWig, and several specialized formats
- Endless extensibility with a plugin ecosystem which can add additional view types, track types, data adapters, and more!
- See a summary of new features and a comparison to JBrowse 1



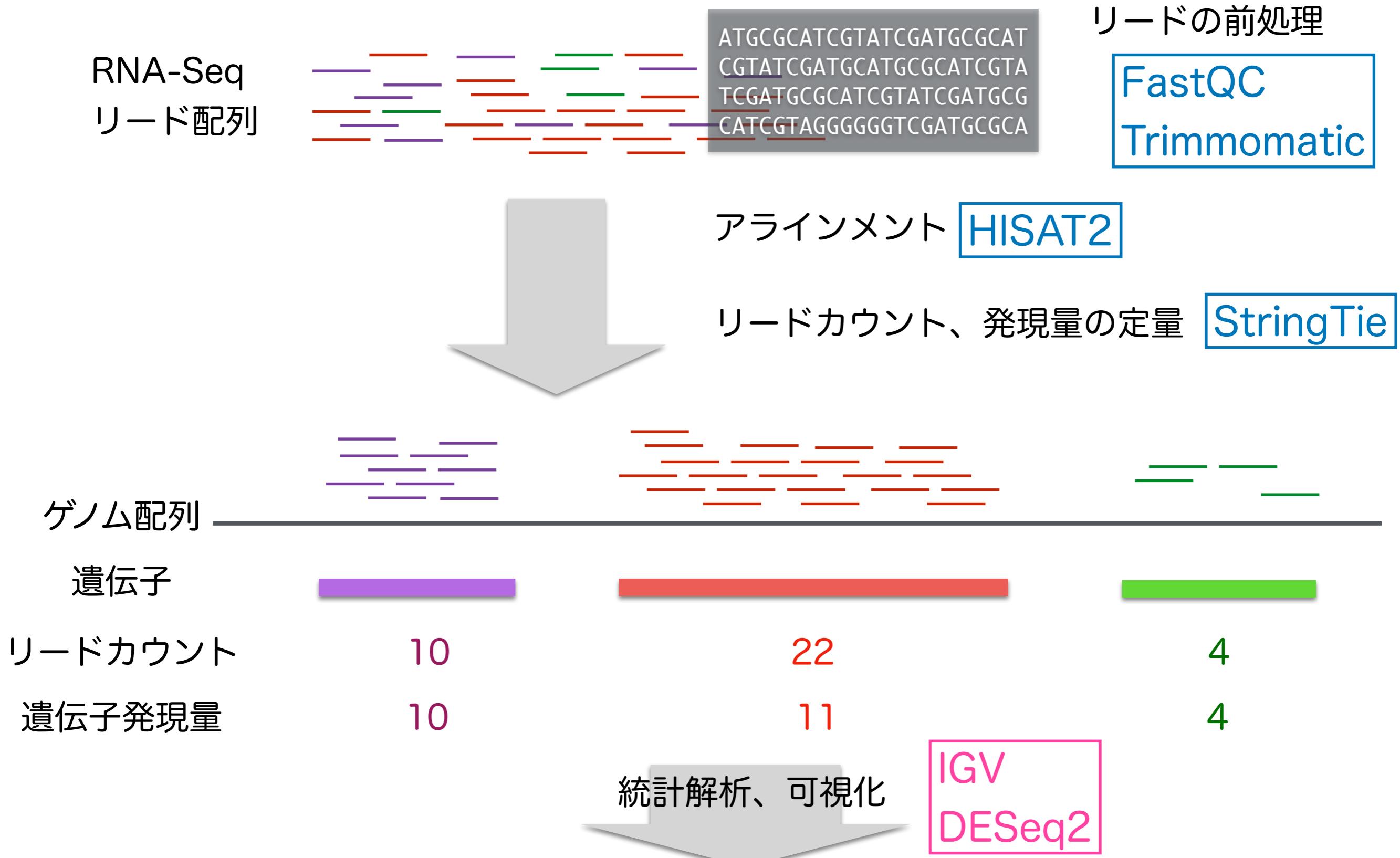
OSごとにプログラムが用意されており、ダウンロードしてすぐに使える。

<https://jbrowse.org/jb2/>

比較ゲノムデータを読み込ませて、ドットプロットを表示させたり、2つのゲノムの対応関係を可視化したり、様々な機能が実装されている。



# 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

# Step 6. リードカウントデータの作成 (for DESeq2)

StringTieに付属のPythonスクリプトを使い、  
遺伝子、転写讀仏ごとのリードカウントデータを作成する。

```
$ less ./step6_make_CountMatrix.sh
```

- step6\_make\_CountMatrix.sh -

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data  
ToolDir=$WorkingDir/tool  
StringTie_bin=$ToolDir/stringtie-2.2.1.Linux_x86_64
```

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 PATH=$StringTie_bin:$PATH
```

exportコマンドで各ツールの  
ディレクトリをPATHに追加。

```
### Step 6: Make count matrix
```

```
python $StringTie_bin/prepDE.py3 \  
-l 101 \  
-i stringtie/ \  
-p rice_
```

リード長やstringtieの結果が可能されたディレクトリ、  
サンプル名のPrefixを指定し、カウントデータを出力

# Step 6. リードカウントデータの作成 (for DESeq2)

シェルスクリプトの実行

(実行時間：約5秒)

```
$ bash ./step6_make_CountMatrix.sh 2>&1 | tee step6_make_CountMatrix.log
```

遺伝子、転写産物ごとのリードカウントデータが出力されている。

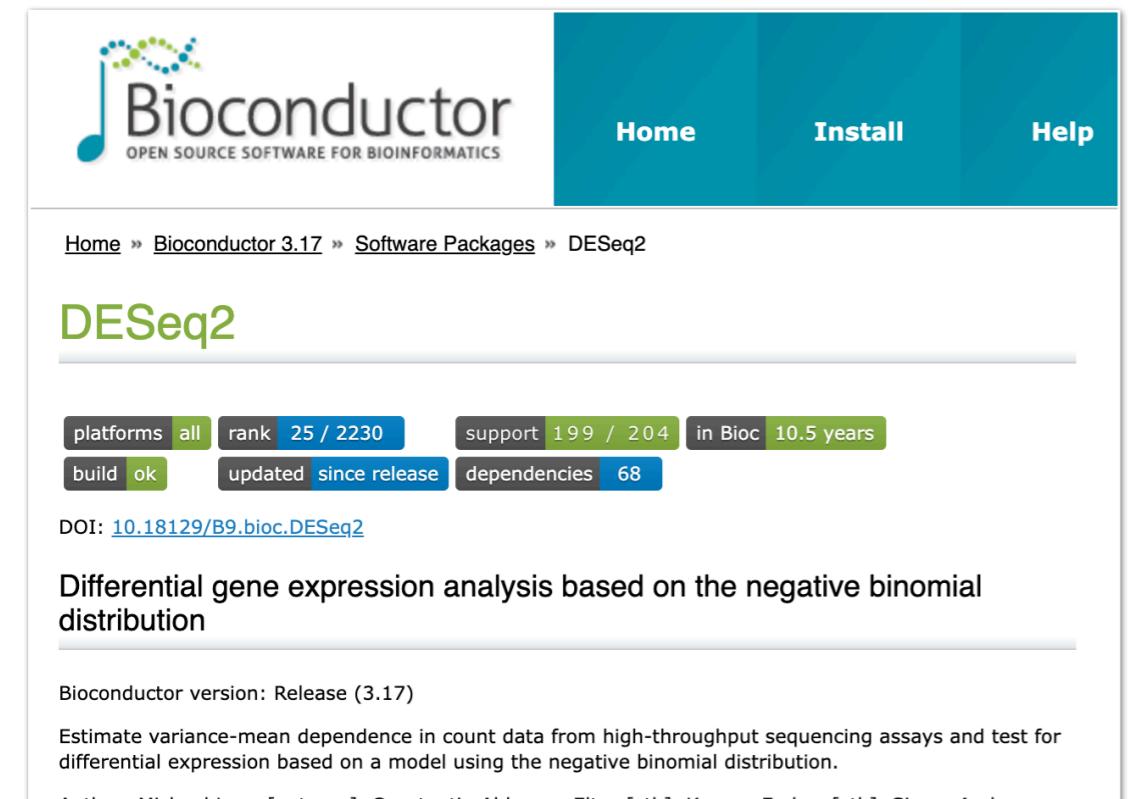
```
$ ls *.csv  
gene_count_matrix.csv transcript_count_matrix.csv
```

遺伝子、サンプルごとにリードカウントがカンマ区切りで保存されている。

```
$ less gene_count_matrix.csv  
gene_id,rice_D_rep1,rice_D_rep2,rice_D_rep3,rice_N_rep1,rice_N_rep2,rice_N_rep3  
...  
0s02g07225001-,147,141,144,130,132,140  
0s02g07227001-,692,678,666,1135,1142,1135  
0s02g072320010sGT7,5,6,7,10,13,16  
0s02g07233001-,131,131,139,370,347,390  
0s02g072280010sWD40-53,631,608,635,936,905,919  
0s02g07228001-,13,10,6,40,21,30  
0s02g07234001IAA8,1,0,1,2,1,1  
0s02g07234001-,62,43,67,29,30,22  
0s02g07237001-,170,163,163,329,319,375  
0s02g07240001DTH2,124,112,101,3982,3974,4012  
...
```

「gene\_count\_matrix.csv」をDESeq2による発現解析に用います。

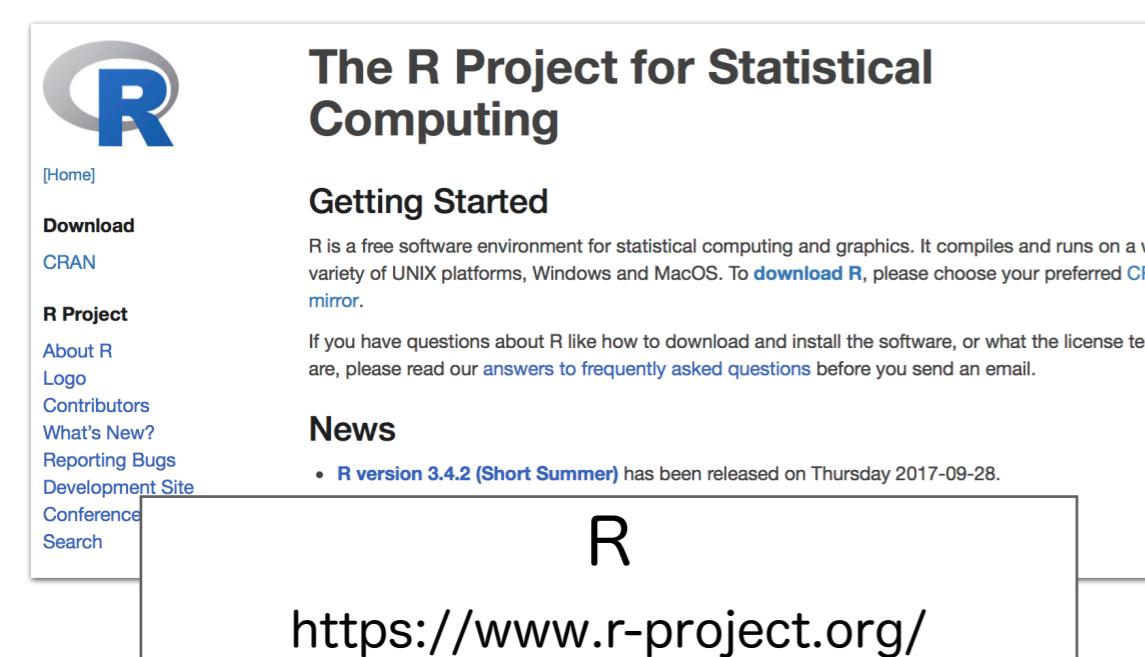
# R/Bioconductorによる統計解析と可視化



The screenshot shows the Bioconductor website for the DESeq2 package. At the top, there's a navigation bar with links for Home, Install, and Help. Below the navigation, a breadcrumb trail shows the path: Home > Bioconductor 3.17 > Software Packages > DESeq2. The main title is "DESeq2". Below the title, there are several status indicators: platforms all (rank 25 / 2230), support 199 / 204 (in Bioc 10.5 years), build ok (updated since release), and dependencies 68. A DOI link ([10.18129/B9.bioc.DESeq2](https://doi.org/10.18129/B9.bioc.DESeq2)) is also present. The description states: "Differential gene expression analysis based on the negative binomial distribution". It also mentions the Bioconductor version (Release 3.17) and provides a brief overview of its function. The author's name is listed as Michael Love [aut, cre], Constantin Ahlmann-Eltze [ctb], Kwame Forbes [ctb], Simon Anders [ctb].

**DESeq2 (Bioconductor)**

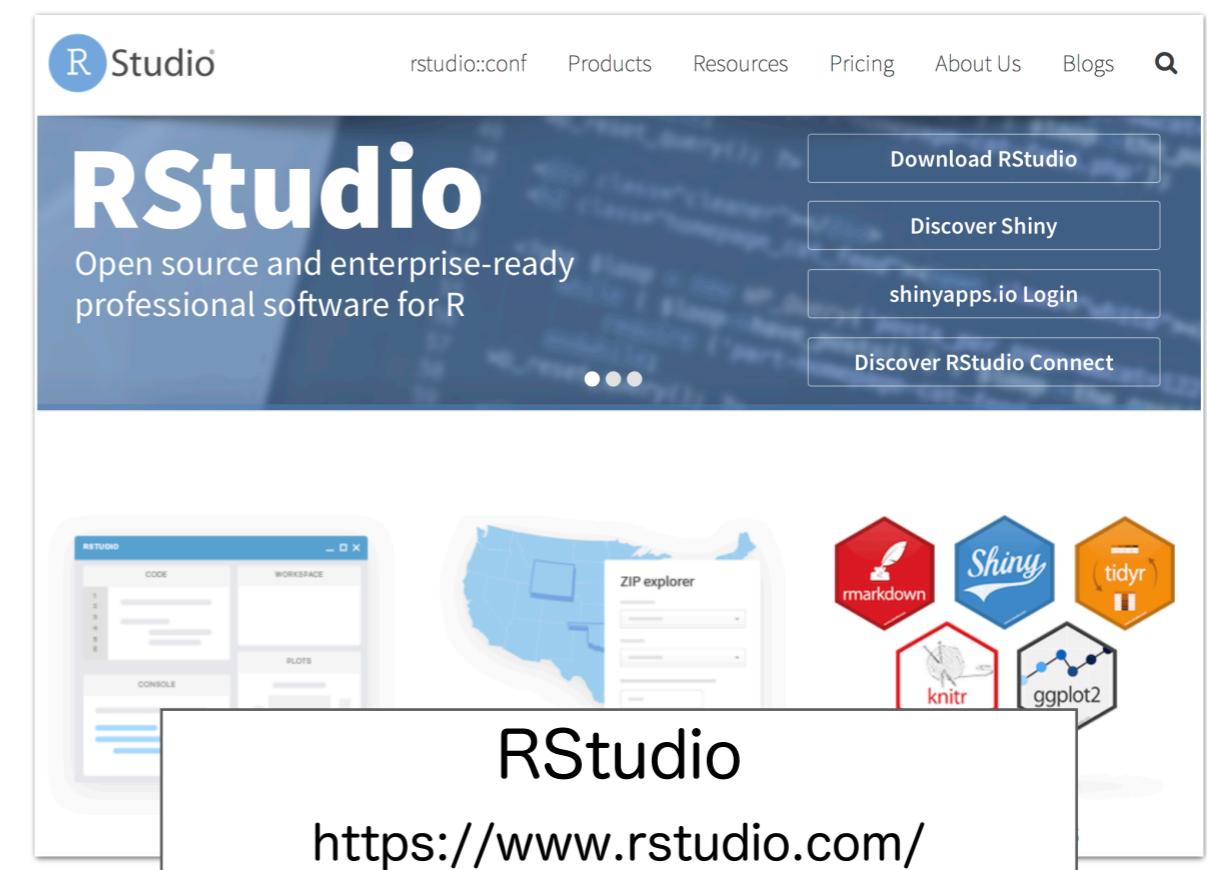
<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>



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 Site, Conference, and Search. The main content area is titled "The R Project for Statistical Computing" and includes sections for "Getting Started" and "News". The "Getting Started" section provides information about how to download and install R, mentioning CRAN mirrors and license terms. The "News" section notes the release of R version 3.4.2 (Short Summer) on Thursday 2017-09-28. At the bottom, there's a large "R" logo and the URL <https://www.r-project.org/>.

DESeq2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. The header includes the R Studio logo and navigation links for rstudio::conf, Products, Resources, Pricing, About Us, and Blogs, along with a search icon. The main title is "RStudio" with the subtitle "Open source and enterprise-ready professional software for R". Below the title, there are four buttons: "Download RStudio", "Discover Shiny", "shinyapps.io Login", and "Discover RStudio Connect". The page features images of the RStudio interface, including the code editor, workspace, plots, and console. To the right, there are icons for various R packages: rmarkdown (red hexagon), Shiny (blue hexagon), tidyverse (orange hexagon), knitr (red hexagon), ggplot2 (grey hexagon), and dplyr (grey hexagon). At the bottom, there's a large "RStudio" logo and the URL <https://www.rstudio.com/>.

# Rstudioの起動



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:

- Source (Top Left):** Displays the R script code for running DESeq2. A box highlights the text "Rスクリプトの編集".
- Environment (Top Right):** Shows the Global Environment with objects like count\_matrix and sample\_info. A box highlights the text "変数一覧".
- Files (Bottom Right):** Displays the file structure under the DESeq2 project. A box highlights the text "作業履歴".
- Console (Bottom Left):** Shows the R command-line interface output. A box highlights the text "Rコンソール".

**Source Pane Content (run\_DESeq2.R):**

```
3 install.packages("pheatmap")
4 install.packages("RColorBrewer")
5 install.packages("tidyverse")
6
7 library("DESeq2")
8 library("tidyverse")
9 library("pheatmap")
10 library("RColorBrewer")
11
12
13 ### Load data & const
14 count_matrix <- as.matrix(read.csv("gene_count_matrix.csv", sep = "\t", row.names="gene_id"))
15 sample_info <- data.frame(condition=c("Day", "Day", "Day", "Day", "Night", "Night", "Night", "Night"))
16 rownames(sample_info) <- c("rice_D_rep1", "rice_D_rep2", "rice_D_rep3", "rice_D_rep4",
17 "rice_N_rep1", "rice_N_rep2", "rice_N_rep3", "rice_N_rep4")
18 sample_info$CONDITION <- factor(sample_info$CONDITION, levels=c("Night", "Day"))
19
20 rownames(sample_info) %in% colnames(count_matrix)
21 all(rownames(sample_info) == colnames(count_matrix))
22
23 dds <- DESeqDataSetFromMatrix(countData = count_matrix,
24 colData = sample_info,
25 design = ~ CONDITION)
26
27 dds
28
29
30
31 ### Pre-filtering
22:1 (Top Level) ▾
```

**Console Pane Output:**

```
The downloaded binary packages are in
  /var/folders/7s/z73njqz94zn434qc9p4ghbp000gn/T//Rtmp4Bj1m9/downloaded_packages
> library("pheatmap")
> library("RColorBrewer")
> count_matrix <- as.matrix(read.csv("gene_count_matrix.csv", sep = "\t", row.names="gene_id"))
Error in h(simpleError(msg, call))
  error in evaluating the argument
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'gene_count_matrix.csv': No such file or directory
> setwd("~/Desktop/DESeq2")
> count_matrix <- as.matrix(read.csv("gene_count_matrix.csv", sep = "\t", row.names="gene_id"))
> sample_info <- data.frame(condition=c("Day", "Day", "Day", "Day", "Night", "Night", "Night", "Night"))
> rownames(sample_info) <- 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")
> sample_info$CONDITION <- factor(sample_info$CONDITION, levels=c("Night", "Day"))
>
```

**Bottom Right Boxes:**

- 変数一覧
- 作業履歴
- ディレクトリ構造
- プロット
- ヘルプ

# 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

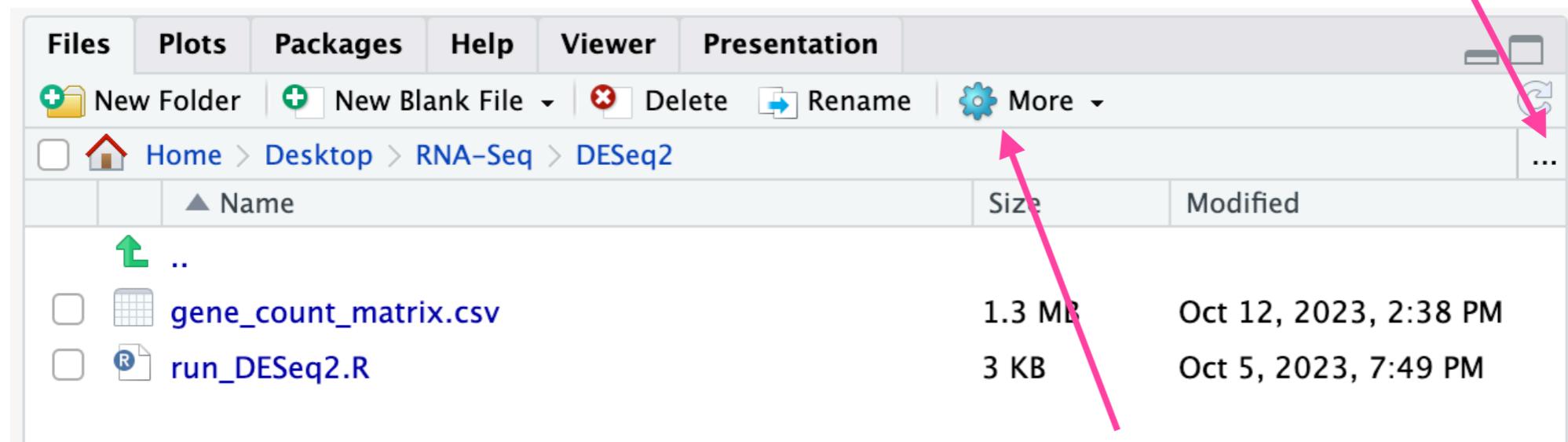
```
> setwd("c:/Users/user/Desktop/RNA-Seq/DESeq2")
```

\*各自の環境にあったパスに置き換えて下さい

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

1. 「...」をクリックし、「DESeq2」ディレクトリを選択



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

# 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「RNA-Seq/DESeq2/」ディレクトリには

- RNA-Seq解析によって得られたリードカウントデータ (gene\_count\_matrix.csv)
- DESeq2解析で実行するコマンドを記載したRのスクリプト (run\_DESeq2.R)

が置いてあります。

The screenshot shows the RStudio interface. On the left, the code editor displays the content of run\_DESeq2.R. On the right, the file browser shows the directory structure: Home > Desktop > RNA-Seq > DESeq2. Inside this folder are two files: gene\_count\_matrix.csv and run\_DESeq2.R. A blue arrow points from the file browser to the run\_DESeq2.R file in the code editor. A pink arrow points from the number 1 in the pink text below to the line 'count\_matrix <- as.matrix(read.csv("gene\_count\_matrix.csv", sep=",", row.names="gene\_id"))' in the code editor. Another pink arrow points from the number 2 in the pink text below to the 'Run' button in the toolbar.

クリックすると左上のペインで  
編集や実行が可能に

2. Runボタンをクリックすると対象行  
のスクリプトが実行される

1. 実行したい行（複数行も可）を選択

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。

```
run_DESeq2.R x gene_count_matrix.csv x
1 gene_id, rice_D_rep1, rice_D_rep2, rice_D_rep3, rice_N_rep1, rice_N_rep2, rice_N_rep3
2 Os01g01002001-, 12, 4, 6, 0, 0, 0
3 Os01g01003001-, 6, 2, 0, 0, 0, 0
4 Os01g01004001-, 109, 114, 117, 159, 171, 159
5 Os01g01004661-, 3, 2, 0, 5, 7, 9
6 Os01g01001001-, 913, 859, 854, 1146, 1155, 1105
7 Os01g01005001-, 429, 473, 477, 746, 739, 680
8 Os01g01006001-, 329, 329, 330, 830, 788, 756
9 Os01g01006501-, 7.8, 6.33, 31.38

run_DESeq2.R x gene_count_matrix.csv x
1 install.packages("BiocManager")
2 BiocManager::install("DESeq2")
3 install.packages("pheatmap")
4 install.packages("RColorBrewer")
5 install.packages("tidyverse")
6
7 library("DESeq2")
8 library("tidyverse")
9 library("pheatmap")
10 library("RColorBrewer")
11
12
13 ### Load data & construct a DESeqDataSet(dds)
14
15 count_matrix <- as.matrix(read.csv("gene_count_matrix.csv", sep=",", row.names="gene_id"))
16
17 sample_info <- data.frame(CONDITION=c("Day", "Day", "Day", "Day", "Night", "Night", "Night", "Night"))
18 rownames(sample_info) <- c("rice_D_rep1", "rice_D_rep2", "rice_D_rep3", "rice_D_rep4",
19 "rice_N_rep1", "rice_N_rep2", "rice_N_rep3", "rice_N_rep4")
20 sample_info$CONDITION <- factor(sample_info$CONDITION, levels=c("Night", "Day"))
21
22 rownames(sample_info) %in% colnames(count_matrix)
23 all(rownames(sample_info) == colnames(count_matrix))
24
25 dds <- DESeqDataSetFromMatrix(countData = count_matrix,
26 colData = sample_info,
27 design = ~ CONDITION)
28
29 dds
```

# パッケージのインストールと読み込み

DESeq2を用いた統計解析に必要なパッケージのインストールと読み込み

```
install.packages("BiocManager")
BiocManager::install("DESeq2")
BiocManager::install("apeglm")
install.packages("tidyverse")
install.packages("pheatmap")
install.packages("RColorBrewer")
```

- DESeq2、apeglm は R/Bioconductor のパッケージとして提供されており、BiocManager の install 関数でインストールする。
- pheatmap、RColorBrewer、tidyverse はデータの可視化に利用する R のパッケージ。R の install.packages 関数を用いてインストールする。
- 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がある。

```
library("DESeq2")
library("apeglm")
library("tidyverse")
library("pheatmap")
library("RColorBrewer")
```

インストールされたパッケージは library 関数で読み込ませると利用できるようになる。

今回は事前にパッケージのインストールが完了しているので、  
library 関数によるパッケージの読み込みから行えば OK です。

# カウントデータの読み込みとサンプル情報の準備

```
count_matrix <- as.matrix(read.csv("gene_count_matrix.csv", sep=",",  
row.names="gene_id"))
```

カウントデータの読み込み

```
sample_info <- data.frame(CONDITION=c("Day", "Day", "Day", "Night", "Night", "Night"))  
rownames(sample_info) <- c("rice_D_rep1", "rice_D_rep2", "rice_D_rep3",  
                           "rice_N_rep1", "rice_N_rep2", "rice_N_rep3")  
sample_info$CONDITION <- factor(sample_info$CONDITION, levels=c("Night", "Day"))
```

サンプル情報を作成

```
> head(count_matrix)  
          rice_D_rep1 rice_D_rep2 rice_D_rep3 rice_N_rep1 rice_N_rep2 rice_N_rep3  
0s01g0100100|-     823      971     1525     1013     1585     1308  
0s01g0100200|-      0        0        4        8       11        0  
0s01g0100466|-      0        0        0        0       0        0  
0s01g0100300|-      5        0        0        2       0        0  
0s01g0100400|-    118      106      97     136      72     146
```

```
> sample_info  
  CONDITION  
rice_D_rep1      Day  
rice_D_rep2      Day  
rice_D_rep3      Day  
rice_N_rep1     Night  
rice_N_rep2     Night  
rice_N_rep3     Night
```

# DESeqオブジェクトにデータを格納する

"sample\_info"と"count\_matrix"中のサンプル情報が一致しているかを確認。  
一致していれば全て"TRUE"が返ってくる。

```
rownames(sample_info) %in% colnames(count_matrix)
all(rownames(sample_info) == colnames(count_matrix))
```

カウントデータとサンプルデータをDESeqオブジェクトに格納

```
dds <- DESeqDataSetFromMatrix(countData = count_matrix,
                               colData = sample_info,
                               design = ~ CONDITION)
```

DESeqオブジェクトの中身を確認

```
> dds
class: DESeqDataSet
dim: 39471 6
metadata(1): version
assays(1): counts
rownames(39471): 0s01g01001001- 0s01g01002001- ... 0s12g06402001- 0s12g06409501-
rowData names(0):
colnames(6): rice_D_rep1 rice_D_rep2 ... rice_N_rep2 rice_N_rep3
colData names(1): CONDITION
```

# 低発現遺伝子のフィルタリング

低発現遺伝子（リードカウントの少ない遺伝子）を除く

ここでは6サンプルのうちの3サンプル以上でリードカウントが10以上ある遺伝子のみを残している

```
smallestGroupSize <- 3  
keep <- rowSums(counts(dds) >= 10) >= smallestGroupSize  
dds.filtered <- dds[keep,]
```

フィルタリング後のdds.filteredオブジェクトの中身を確認

遺伝子数が 39,471個から22,959個に減っていることが分かる

```
> dds.filtered  
class: DESeqDataSet  
dim: 22959 6  
metadata(1): version  
assays(4): counts mu H cooks  
rownames(22959): 0s01g01001001- 0s01g01004001- ... 0s12g02223001- 0s12g04955251-  
rowData names(22): baseMean baseVar ... deviance maxCooks  
colnames(6): rice_D_rep1 rice_D_rep2 ... rice_N_rep2 rice_N_rep3  
colData names(2): CONDITION sizeFactor
```

フィルタリングは必ずしも必要ではないが、計算時間や計算リソースの節約になる。

# 発現変動遺伝子の検出

発現変動遺伝子の検出は「DESeq」関数で行う。

その後、「results」関数を使い、指定した条件での比較結果を取り出す。

```
dds.filtered <- DESeq(dds.filtered)
res <- results(dds.filtered, contrast=c("CONDITION", "Night", "Day"))
```

「夜 vs 昼」の発現変動解析の結果

```
> head(res)
log2 fold change (MLE): CONDITION Night vs Day
Wald test p-value: CONDITION Night vs Day
DataFrame with 6 rows and 6 columns
  baseMean log2FoldChange    lfcSE      stat     pvalue     padj
  <numeric>       <numeric> <numeric>   <numeric>   <numeric>   <numeric>
0s01g01001001-  1168.324    0.0554208  0.186909  0.2965120 7.66839e-01 9.04033e-01
0s01g01004001-  116.592     -0.0520142   0.533964 -0.0974113 9.22400e-01 9.71054e-01
0s01g01005001-  861.826     -0.1283867   0.248570 -0.5165009 6.05505e-01 8.17442e-01
0s01g01008001-  612.356     -0.0980066   0.212735 -0.4606983 6.45015e-01 8.41387e-01
0s01g01006001-  875.783     0.1612476   0.247954  0.6503121 5.15491e-01 7.55980e-01
0s01g01007001-  1645.227    -1.6529284   0.302303 -5.4677953 4.55668e-08 1.57195e-06
```

各項目の詳細を表示

```
> mcols(res)$description
[1] "mean of normalized counts for all samples"      "log2 fold change (MLE): CONDITION Night vs Day"
[3] "standard error: CONDITION Night vs Day"        "Wald statistic: CONDITION Night vs Day"
[5] "Wald test p-value: CONDITION Night vs Day"      "BH adjusted p-values"
```

# 発現変動遺伝子の検出

解析結果をP値の低い順にソート

```
resOrdered <- res[order(res$pvalue),]
```

```
> head(resOrdered)
```

...

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
0s03g01109001-	3899.54	-5.80983	0.249837	-23.2545	1.28142e-119	2.93086e-115
0s12g06018001BZIP88	1835.02	-5.45429	0.294559	-18.5168	1.51162e-76	1.72868e-72
0s05g03950001-	2237.44	-4.49354	0.254506	-17.6559	9.16391e-70	6.98657e-66
0s04g05839001LHY	7115.01	9.42496	0.540021	17.4530	3.26770e-68	1.86847e-64
0s02g068520010sMyb1R	4412.86	8.86984	0.532802	16.6475	3.15276e-62	1.44220e-58
0s07g05296001-	3736.21	-10.70180	0.675737	-15.8372	1.72250e-56	6.56616e-53

...

「summary」関数を使うと、発現変動遺伝子数を表示してくれる（デフォルトは p-value < 0.1）

```
> summary(resOrdered)
```

out of 22959 with nonzero total read count

adjusted p-value < 0.1

LFC > 0 (up) : 2452, 11%

LFC < 0 (down) : 2902, 13%

outliers [1] : 87, 0.38%

low counts [2] : 0, 0%

...

# 発現変動遺伝子の検出とファイル出力

「results」関数を実行時に、p-valueの閾値を指定できる（ここでは p-value < 0.01）。

```
res001 <- results(dds.filtered, contrast=c("CONDITION", "Night", "Day"), alpha=0.01)
```

```
> summary(res001)
out of 22959 with nonzero total read count
adjusted p-value < 0.01
LFC > 0 (up)      : 1156, 5%
LFC < 0 (down)    : 1599, 7%
outliers [1]       : 87, 0.38%
low counts [2]     : 0, 0%
...
...
```

「results」関数の出力に対してsubset関数を使うことでも、padj が閾値以下の遺伝子を抽出できる。また、抽出した発現変動遺伝子についてのデータを、タブ区切りのテキストファイルで出力できる。

```
resSig <- subset(res0rdered, padj < 0.01)

write.table(as.data.frame(resSig),
            file="DEG_genes.tsv", quote=F, sep="\t", row.names=T, col.names=NA)
```

# 発現変動遺伝子の検出とファイル出力

出力したタブ区切りのテキストファイルはExcelで開くことができる。

	A	B	C	D	E	F	G
1		baseMean	log2FoldChange	IfcSE	stat	pvalue	padj
2	Os03g0110900 -	3899.541581	-5.809832586	0.249837082	-23.25448468	1.28E-119	2.93E-115
3	Os12g0601800 BZIP88	1835.017856	-5.454292414	0.294559141	-18.51679901	1.51E-76	1.73E-72
4	Os05g0395000 -	2237.436504	-4.493539208	0.254506188	-17.65591337	9.16E-70	6.99E-66
5	Os04g0583900 LHY	7115.012902	9.424963705	0.540020511	17.45297356	3.27E-68	1.87E-64
6	Os02g0685200 OsMyb1R	4412.859148	8.869839564	0.532801678	16.64754434	3.15E-62	1.44E-58
7	Os07g0529600 -	3736.211494	-10.70179867	0.675736877	-15.83722752	1.72E-56	6.57E-53
8	Os09g0286600 -	763.0360114	-5.07560565	0.332230154	-15.277378	1.08E-52	3.54E-49
9	Os02g0724000 DTH2	2359.935296	5.229617738	0.354713431	14.74321884	3.40E-49	9.73E-46
10	Os01g0810100 -	1867.714483	-3.51411798	0.240878255	-14.58877218	3.31E-48	8.41E-45
11	Os01g0825500 -	1966.834352	-4.563437247	0.314962874	-14.48881003	1.43E-47	3.26E-44
12	Os06g0270900 -	13161.46746	-2.430997538	0.170460162	-14.26138228	3.81E-46	7.92E-43
13	Os03g0339900 CIPK10	2339.900331	-5.251956798	0.377074356	-13.92817283	4.27E-44	8.14E-41
14	Os10g0330400 -	3539.156078	6.645839382	0.492543448	13.49289978	1.72E-41	3.03E-38
15	Os03g0279600 -	671.1552561	-2.731663738	0.204286059	-13.37175796	8.84E-41	1.44E-37
16	Os02g0744000 -	8052.401106	3.568624632	0.26877169	13.27753172	3.13E-40	4.77E-37
17	Os10g0565200 -	608.9188508	7.829732121	0.589987107	13.27102241	3.41E-40	4.87E-37
18	Os04g0678700 OsPorA	1480.469239	-7.579595748	0.575800019	-13.16359065	1.42E-39	1.91E-36



log2(Day/Night)

正の値 = 昼の発現量の方が高い

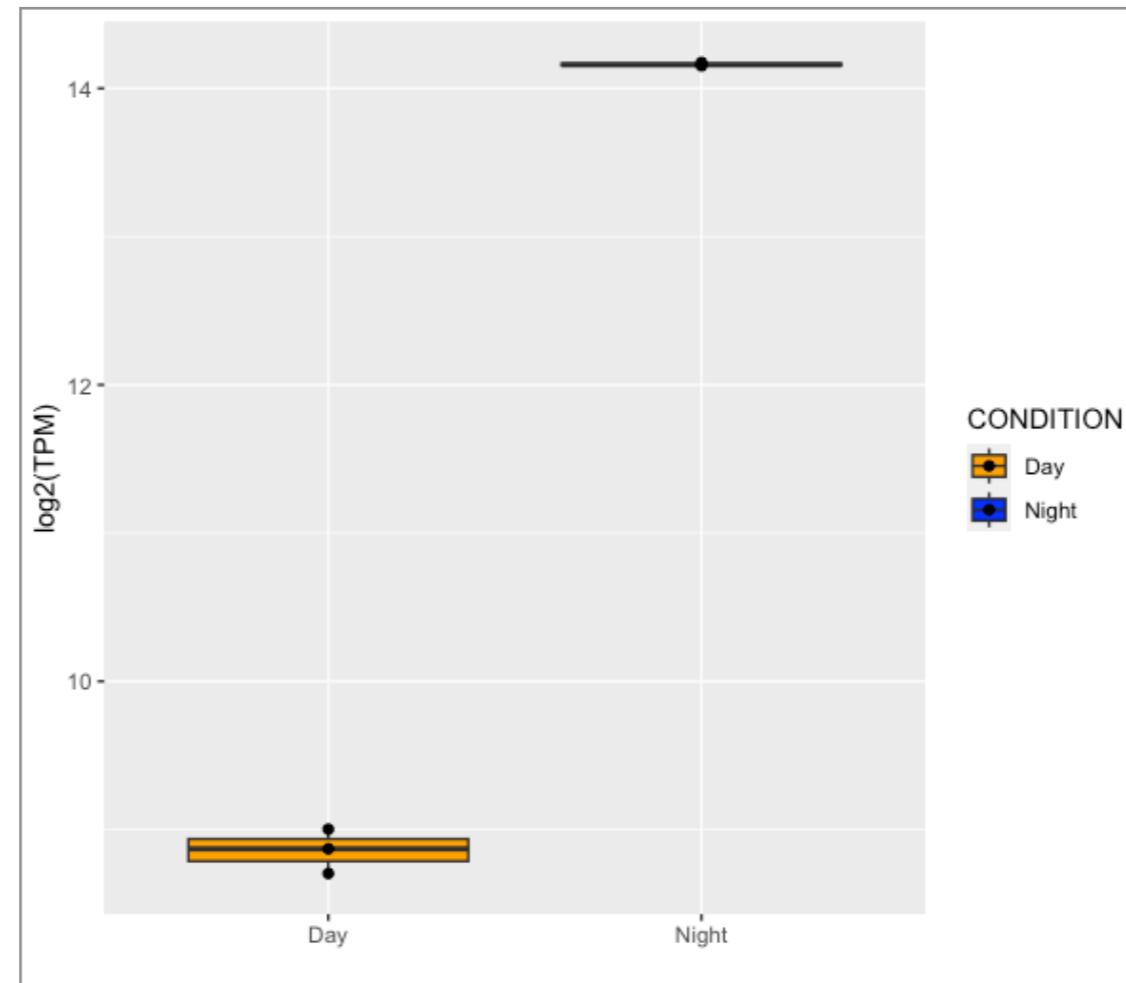
負の値 = 夜の発現量の方が高い

# Boxplotによる個々の遺伝子の発現量の差の可視化

StringTieの出力結果にある「DTH2」の各サンプルのTPM値を調べて、以下のようなスクリプトを用意すれば、論文でよく見るような発現比較のBoxplotを描くことができる。

```
DTH2 TPM <- data.frame(CONDITION=c("Day", "Day", "Day", "Night", "Night", "Night"),
                         TPM=c(513, 468, 417, 18292, 18494, 18206))

ggplot(DTH2 TPM, aes(x=CONDITION, y=log2(TPM), fill=CONDITION)) +
  geom_boxplot() +
  geom_point() +
  scale_fill_manual(values=c("orange", "blue")) +
  xlab("") +
  ylab("log2(TPM)")
```



# 発現データの変換と可視化

発現変動遺伝子を検出する際は、離散的なリードカウントデータを用いたが、クラスタリングや可視化のような解析には数値を変換したものを用いる。Log2対数変換が有名だが、ここではDESeq2で用意されている vst (variance stabilizing transformation) を用いている。

この変換では単に対数変換するだけではなく、ライブラリサイズ等を考慮した正規化も行われている。

```
vsd <- vst(dds, blind=FALSE)
```

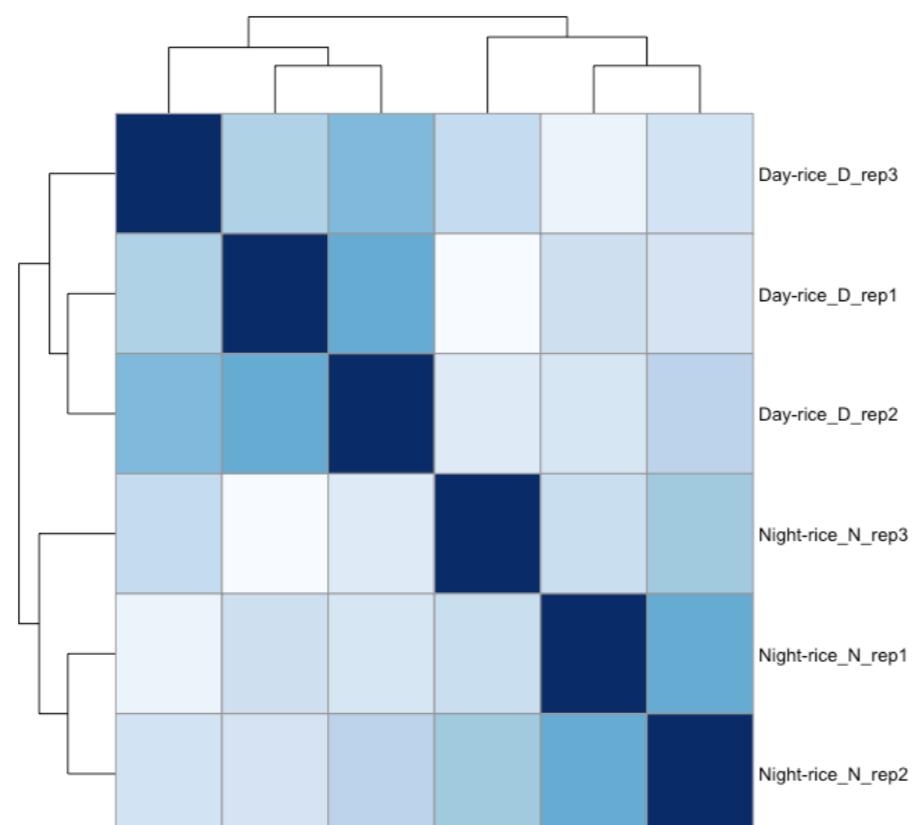
```
> head(assay(vsd))
```

	rice_D_rep1	rice_D_rep2	rice_D_rep3	rice_N_rep1	rice_N_rep2	rice_N_rep3
0s01g01001001-	10.240086	10.216480	10.411150	10.240052	10.373907	10.411757
0s01g01002001-	5.771442	5.771442	6.124666	6.343701	6.335826	5.771442
0s01g01004661-	5.771442	5.771442	5.771442	5.771442	5.771442	5.771442
0s01g01003001-	6.274122	5.771442	5.771442	6.058975	5.771442	5.771442
0s01g01004001-	7.998133	7.738110	7.423467	7.937242	7.169270	7.887204
0s01g01005001-	10.000105	9.828206	10.087429	9.713056	9.720833	10.114253

# クラスタリングによるサンプル間の遺伝子発現比較

以下のスクリプトでは vst 変換した発現データを用いて、サンプル間の発現プロファイルの相関を計算し、クラスタリング結果とヒートマップを描画している。

```
sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(vsd$CONDITION, colnames(vsd), sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



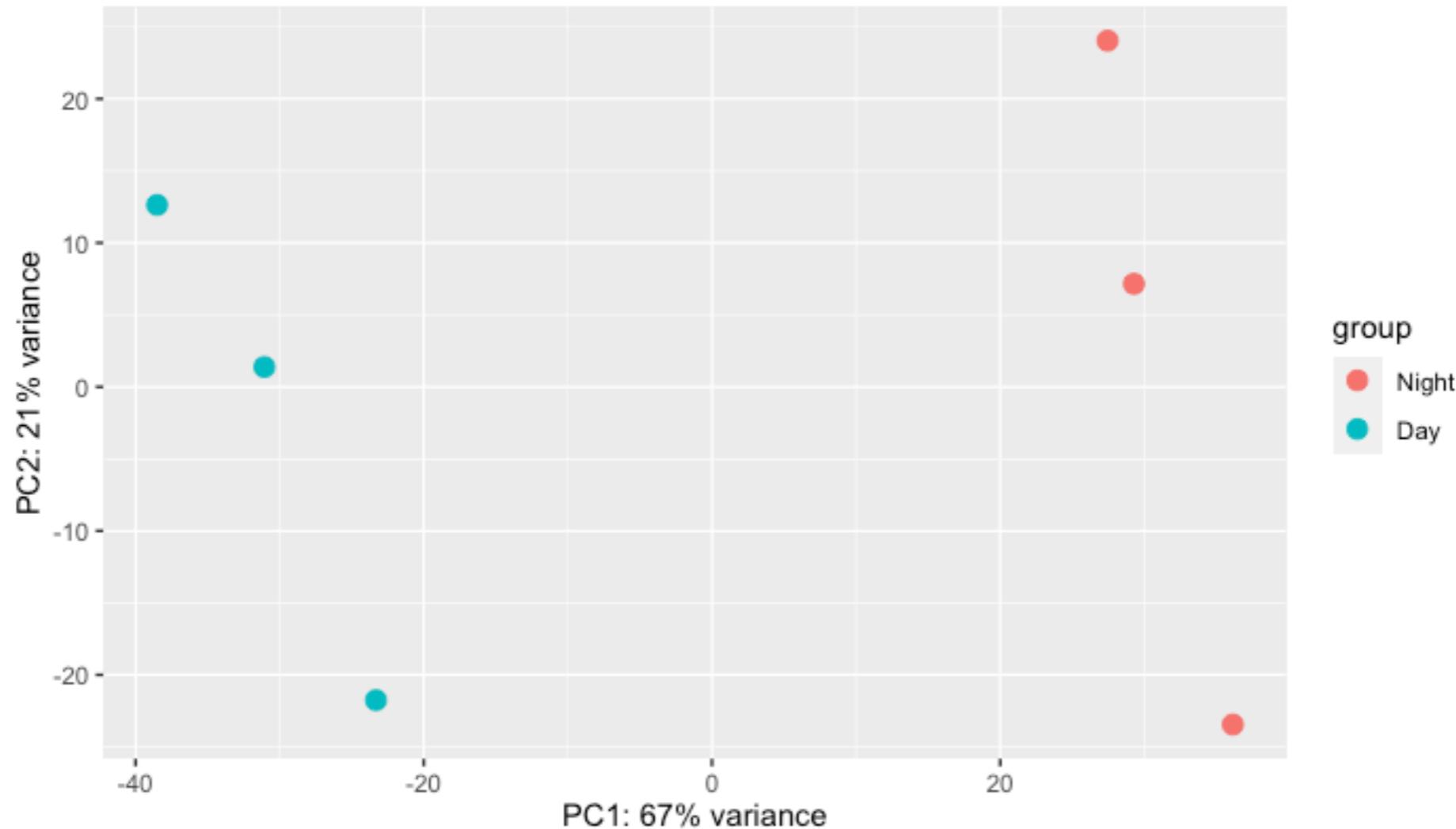
あまりに相関の低い反復サンプルは除いた方が良い。  
このようなプロットを描くことで、サンプルの特徴が  
分かり、取り違い等にも気づける。

今回のデータでは夜の反復サンプル間の相関がやや低  
いことが分かる。

# 主成分分析によるサンプル間の遺伝子発現比較

主成分分析によって、サンプル間の遺伝子発現比較を行う。

```
plotPCA(vsd, intgroup=c("CONDITION"))
```



# MA-plotによる発現変動の可視化

MA-plotを描くことによって、2条件間での発現変動の様子を俯瞰することができる。

MA-plotを描くに当たり、Log2FoldChange (LFC) の値の縮小を行っています。

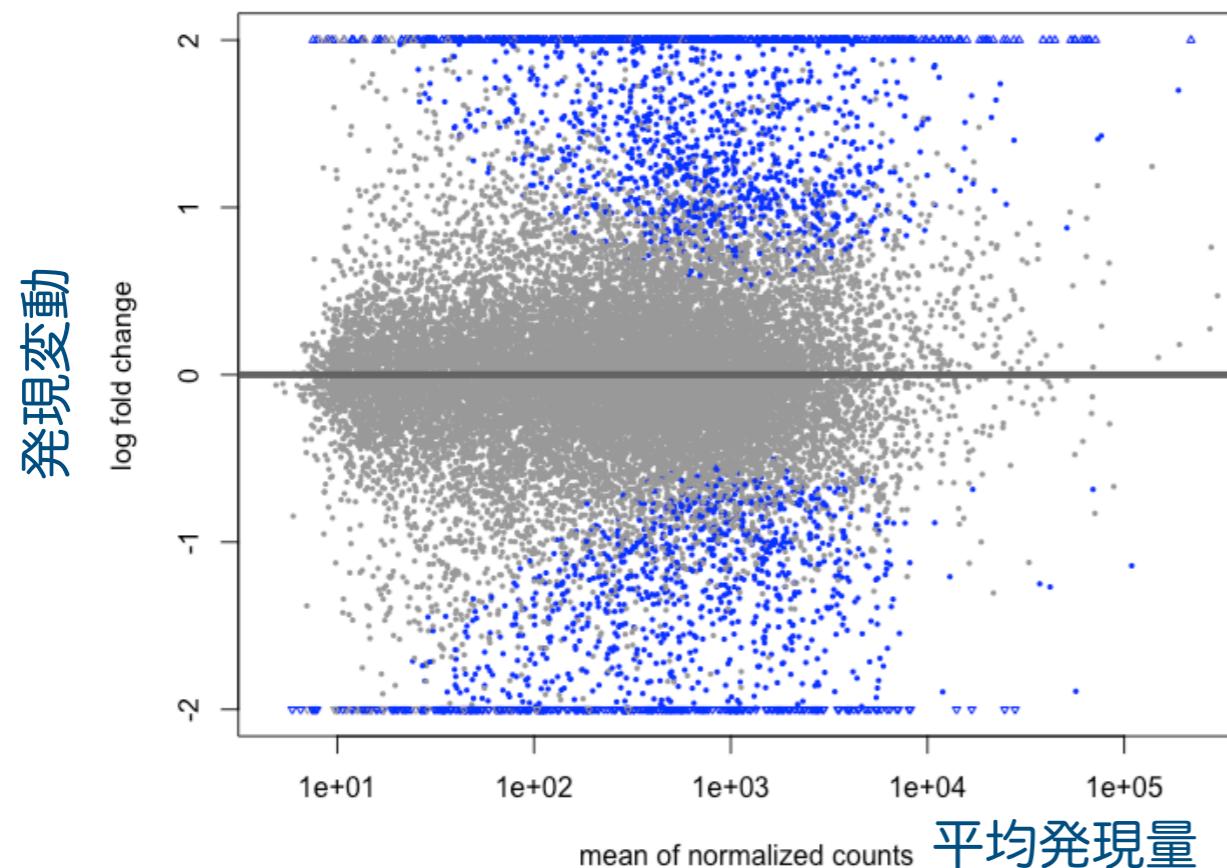
LFCを縮小することにより、より効果的な可視化が行える。

以下で lfcShrink 関数で指定する係数の名前を調べる。

```
> resultsNames(dds.filtered)
[1] "Intercept"           "CONDITION_Day_vs_Night"
```

lfcShrink 関数を用いて apegIm 法によって縮小したLFCを取得し、MA-plotを描画する。

```
resLFC <- lfcShrink(dds.filtered, coef="CONDITION_Day_vs_Night", type="apeglm")
plotMA(resLFC, ylim=c(-2,2), alpha=0.01)
```



MA-plot は横軸が平均発現量、縦軸が発現変動 (LFC) を表している。  
青いポイントは、有意 ( $p\text{-value} < 0.01$ ) に発現が変動している遺伝子を示す。

# DESeq2の使い方

開発者が非常に丁寧なマニュアルを公開してくれています。必読！

<https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html>

## Analyzing RNA-seq data with DESeq2

Michael I. Love, Simon Anders, and Wolfgang Huber

06/23/2023

### Abstract

A basic task in the analysis of count data from RNA-seq is the detection of differentially expressed genes. This vignette shows how to use the `DESeq2` package to perform this task. The package provides a function `DESeq` which takes a `CountDataFrame` object as input and returns a `DESeqResult` object, presented as a table which reports, for each sample, the number of sequence fragments that have been mapped. Analogous data also arise for other assay types, including comparative ChIP-Seq, HiC, shRNA screens, and RNA-seq. An important analysis question is the quantification and statistical inference of systematic changes between samples compared to within-condition variability. The package `DESeq2` provides methods to test for differential expression using negative binomial generalized linear models; the estimates of dispersion and logarithmic fold change are based on maximum likelihood estimation. This vignette explains the use of the package and demonstrates typical workflows. A more detailed description of the analysis of RNA-seq data can be found in the Bioconductor website ([www.bioconductor.org](http://www.bioconductor.org)). The Bioconductor website covers similar material to this vignette but at a slower pace, including the generation of differential expression tables and plots. The `DESeq2` package version: 1.40.2

### Standard workflow

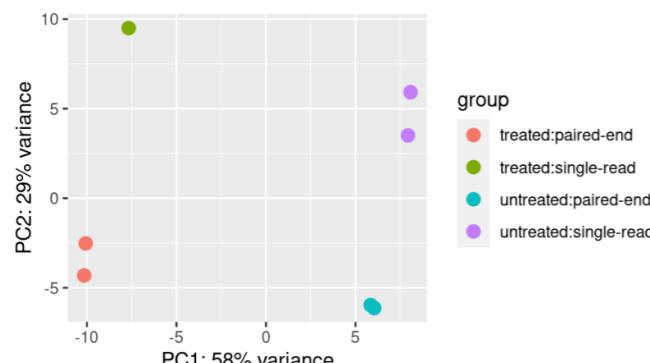
- Quick start
- How to get help for `DESeq2`
- Acknowledgments
- Funding
- Input data

#### Why un-normalized counts?

### Principal component plot of the samples

Related to the distance matrix is the PCA plot, which shows the samples in the 2D plane spanned by their first two principal components. This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects.

```
plotPCA(vsd, intgroup=c("condition", "type"))
```



### Differential expression analysis

The standard differential expression analysis steps are wrapped into a single function, `DESeq`. The estimation steps performed by this function are described [below](#), in the manual page for `?DESeq` and in the Methods section of the `DESeq2` publication (Love, Huber, and Anders 2014).

Results tables are generated using the function `results`, which extracts a results table with log2 fold changes, *p* values and adjusted *p* values. With no additional arguments to `results`, the log2 fold change and Wald test *p* value will be for the **last variable** in the design formula, and if this is a factor, the comparison will be the **last level** of this variable over the **reference level** (see previous [note on factor levels](#)). However, the order of the variables of the design do not matter so long as the user specifies the comparison to build a results table for, using the `name` or `contrast` arguments of `results`.

Details about the comparison are printed to the console, directly above the results table. The text, `condition treated vs untreated`, tells you that the estimates are of the logarithmic fold change  $\log_2(\text{treated}/\text{untreated})$ .

```
dds <- DESeq(dds)
res <- results(dds)
res
```

```
## log2 fold change (MLE): condition treated vs untreated
## Wald test p-value: condition treated vs untreated
## DataFrame with 9921 rows and 6 columns
```

### Heatmap of the sample-to-sample distances

Another use of the transformed data is sample clustering. Here, we apply the `dist` function to the transpose of the transformed count matrix to get sample-to-sample distances.

```
sampleDists <- dist(t(assay(vsd)))
```

A heatmap of this distance matrix gives us an overview over similarities and dissimilarities between samples. We have to provide a hierarchical clustering `hc` to the heatmap function based on the sample distances, or else the heatmap function would calculate a clustering based on the distances between the rows/columns of the distance matrix.

```
library("RColorBrewer")
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(vsd$condition, vsd$type, sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
heatmap(sampleDistMatrix,
        clustering_distance_rows=sampleDists,
        clustering_distance_cols=sampleDists,
        col=colors)
```



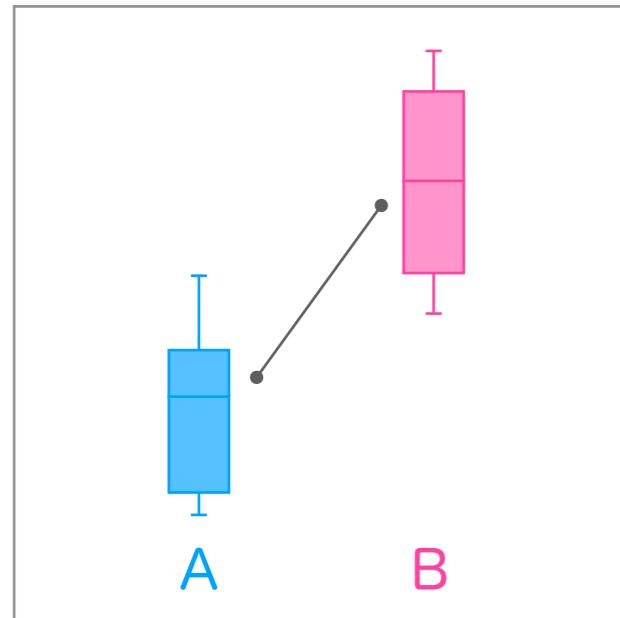
	pvalue	padj
<numeric>	<numeric>	
0.9918817	0.997211	
0.8172987	NA	
0.0575591	0.288002	
0.4808558	0.826834	
0.7597879	0.943501	
...	...	
0.020307	0.144240	
0.216203	0.607848	
0.910615	0.982657	
0.936291	0.988179	
0.860522	0.967928	

# 3群以上の発現比較はどうする？

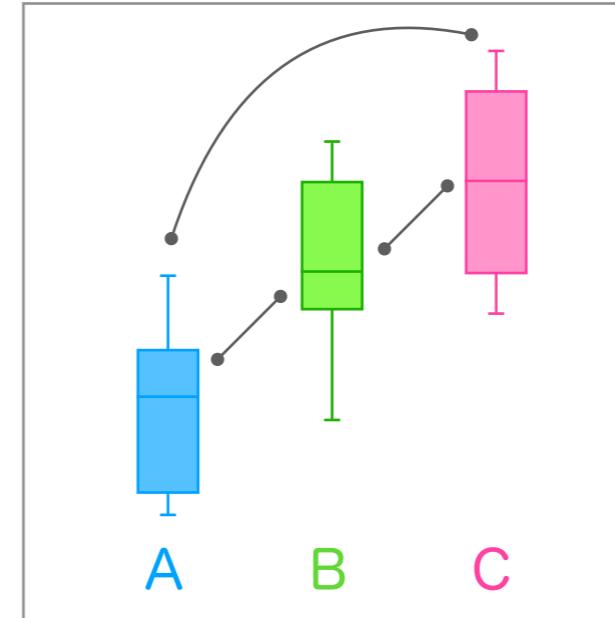
3群以上、時系列データの発現比較には尤度比検定が便利。

その後、DEGをクラスタリング等によって発現プロファイルごとに分類すると良い。

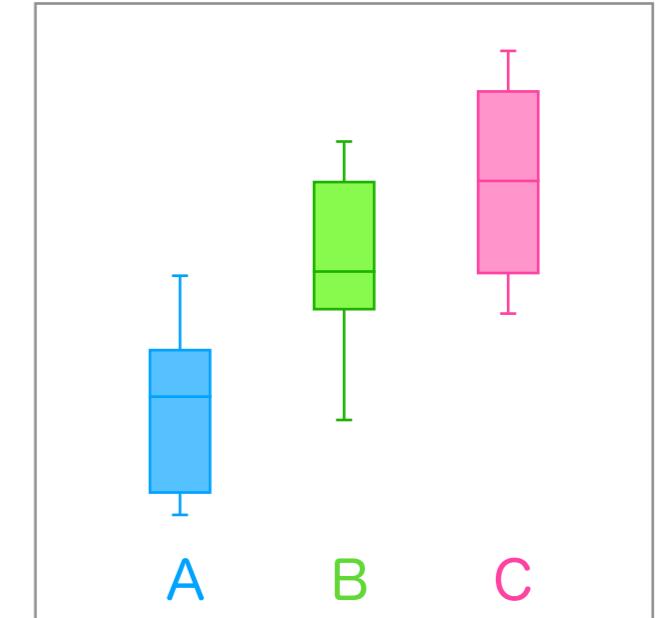
2群間比較  
(Wald 検定)



3群間比較  
(Wald 検定)



3群間比較  
(尤度比検定、LRT)



Full model  $A \neq B \neq C$  尤度を  
Reduced model  $A = B = C$  比較

詳しくは以下のウェブページ等を参照

Analyzing RNA-seq data with DESeq2

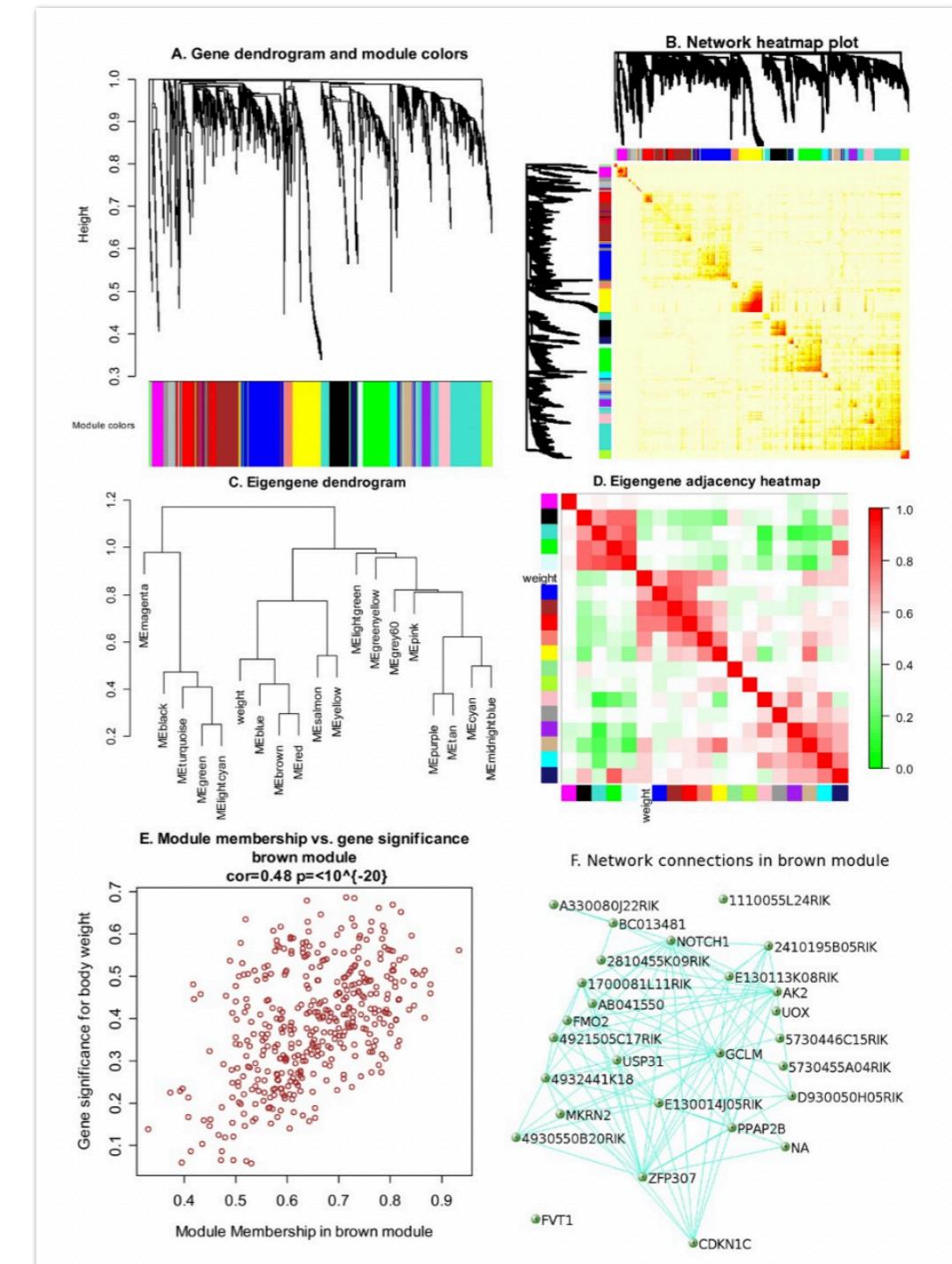
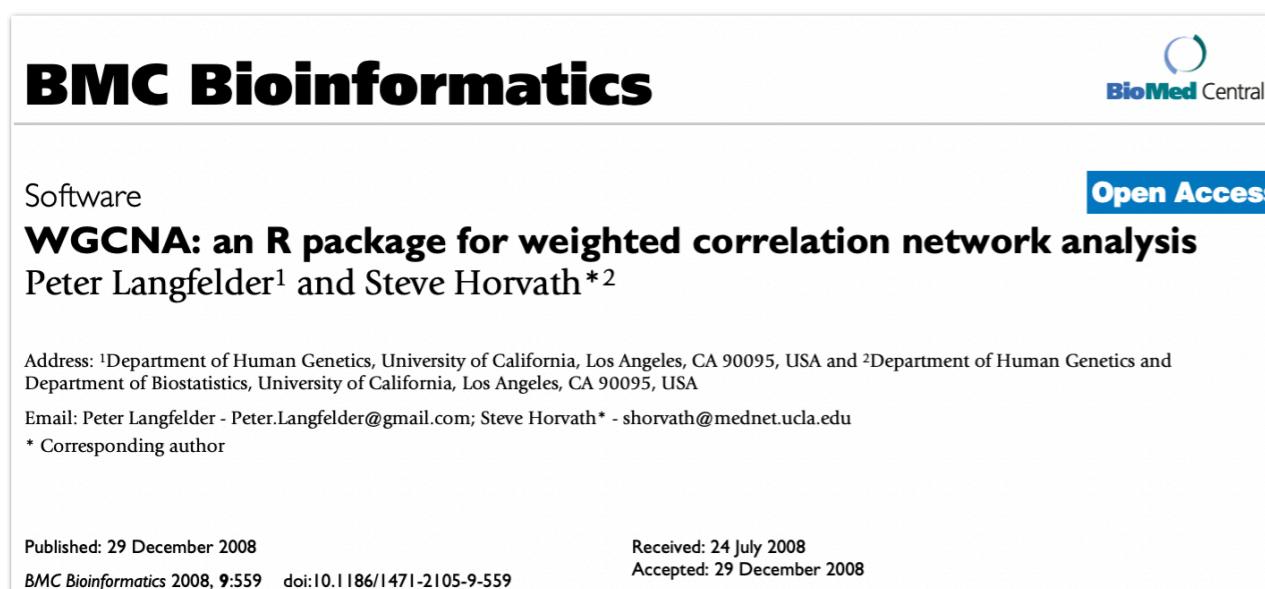
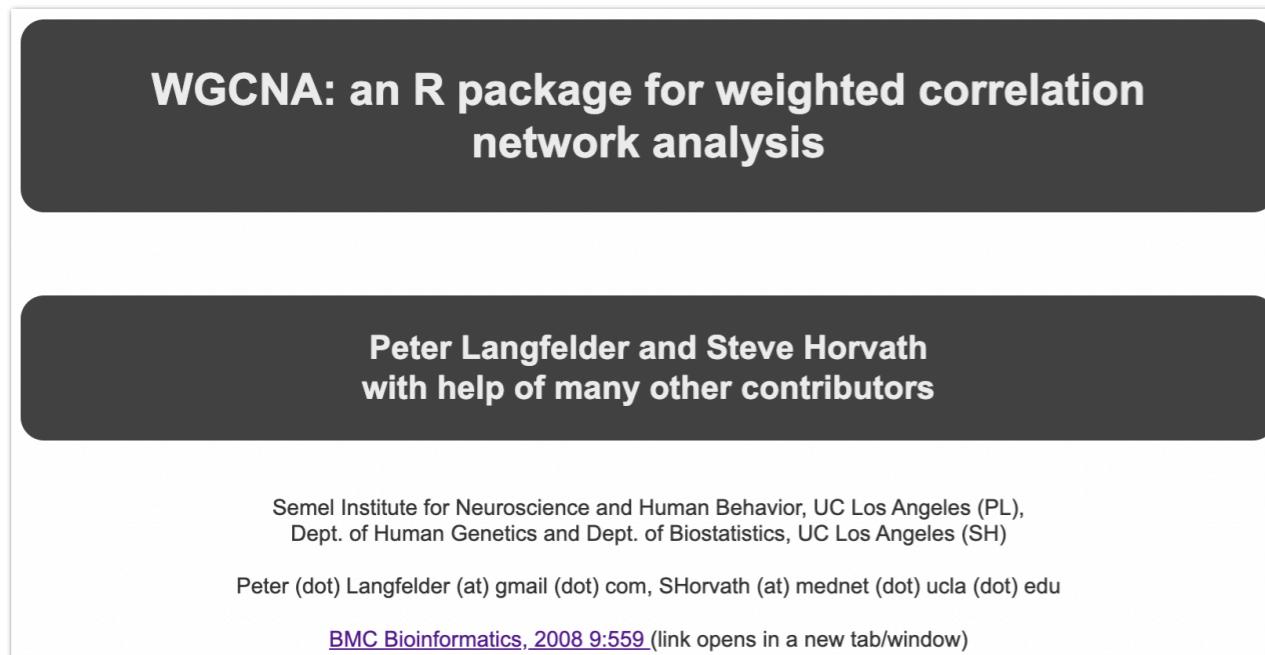
<https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html>

RNA-seq workflow

<https://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>

# 共発現解析：発現プロファイルが似た遺伝子をクラスタリング

Weighted Gene Coexpression Network Analysis (WGCNA) を使うと、遺伝子発現量の相関を元に、互いによく似た発現プロファイルをもつ遺伝子をクラスタリングすることができる。



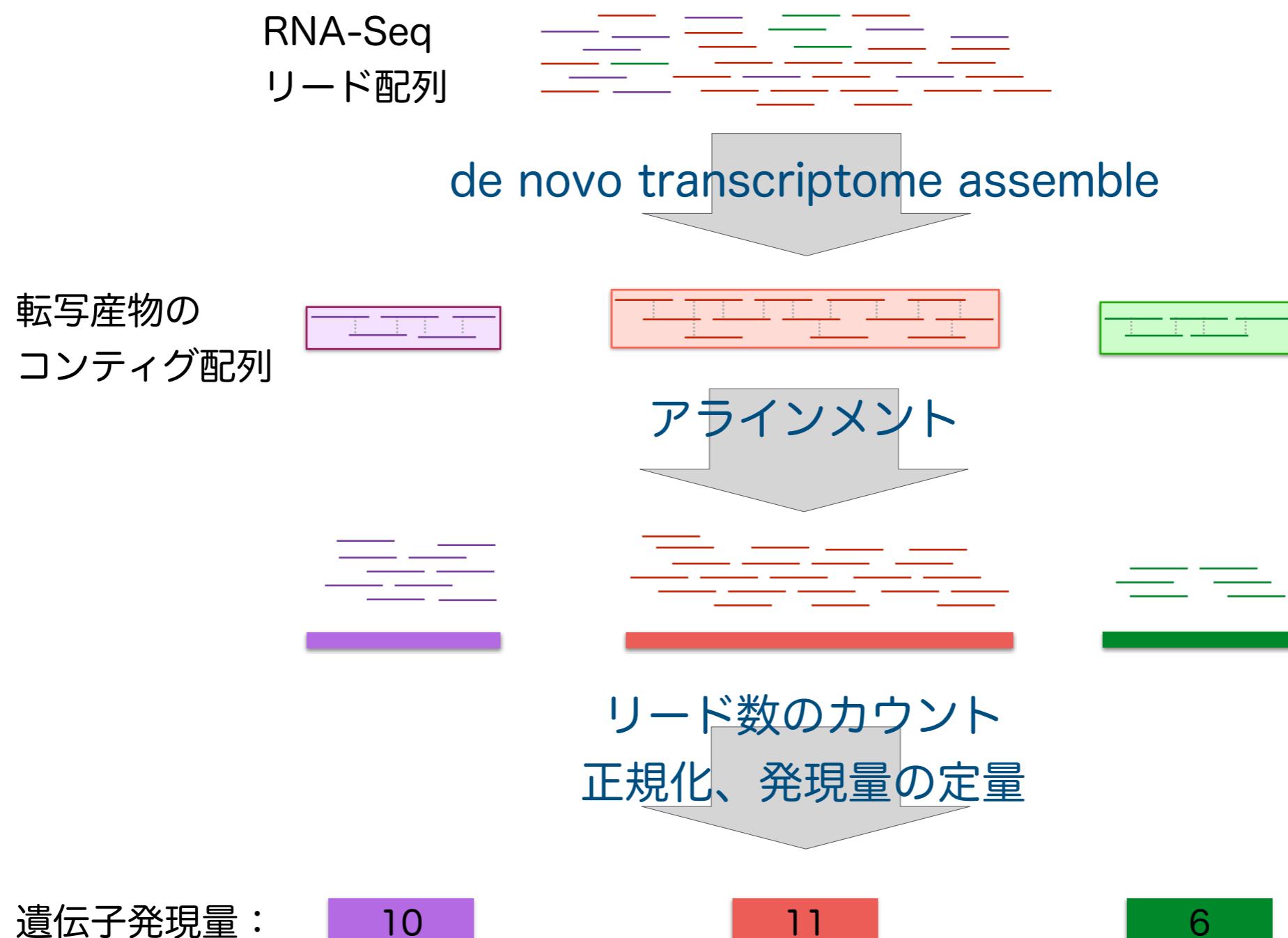
詳しくは以下のウェブページ等を参照

<https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/>

- ▶ 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
- ▶ 遺伝子機能アノテーション

# de novo transcriptome assemble法による遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物配列を再構築する。  
その後、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して遺伝子発現を定量する。



# De novo transcriptome assemble 解析の特徴

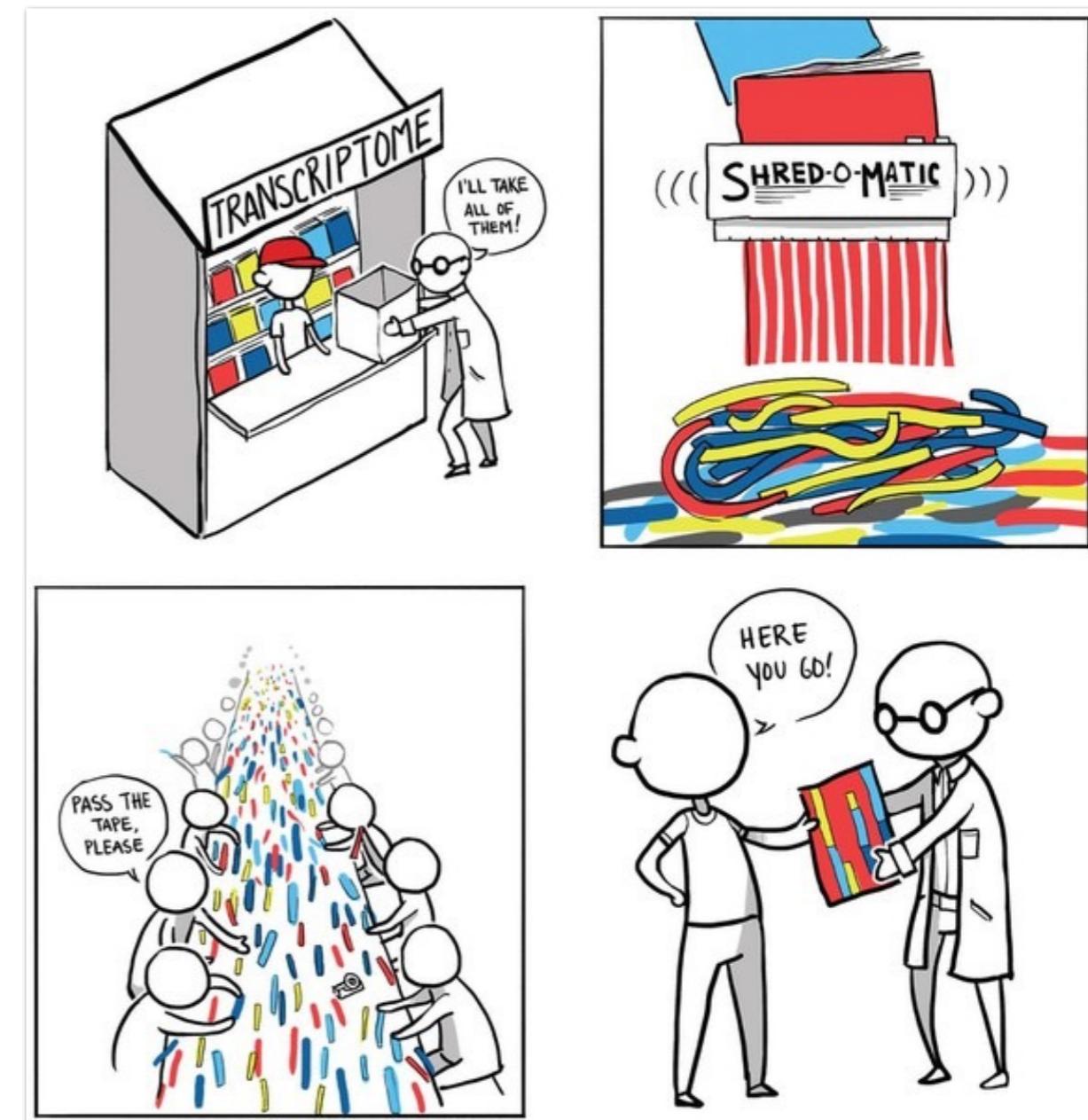
## 良い点

- ・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、コストパフォーマンスの良い解析手法としてよく行われている。



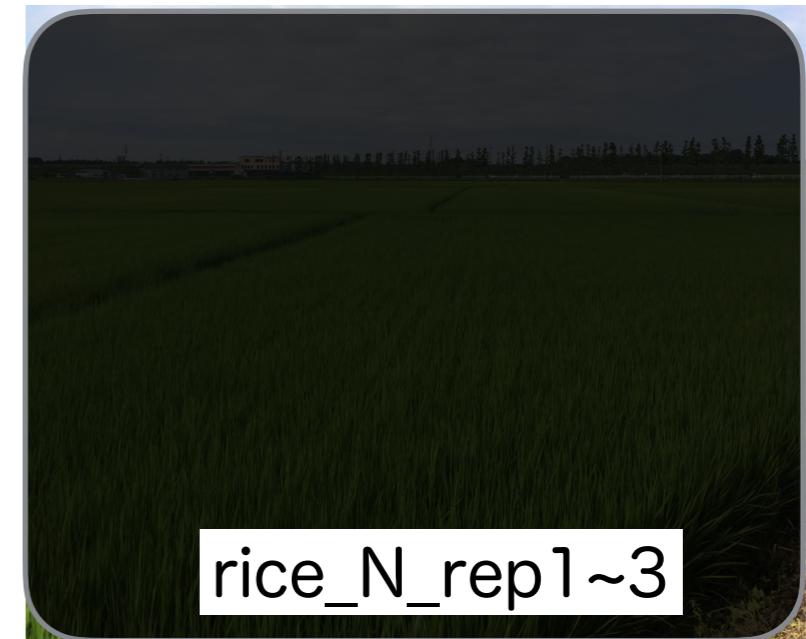
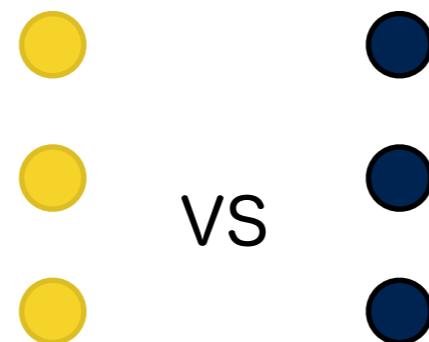
Nature Methods 10, 1165–1166 (2013)

# 演習の目的と使用するデータ

目的 : De novo transcriptome assemble 解析によって  
昼と夜でのイネの葉の遺伝子発現の違いを調べる



2:00 PM



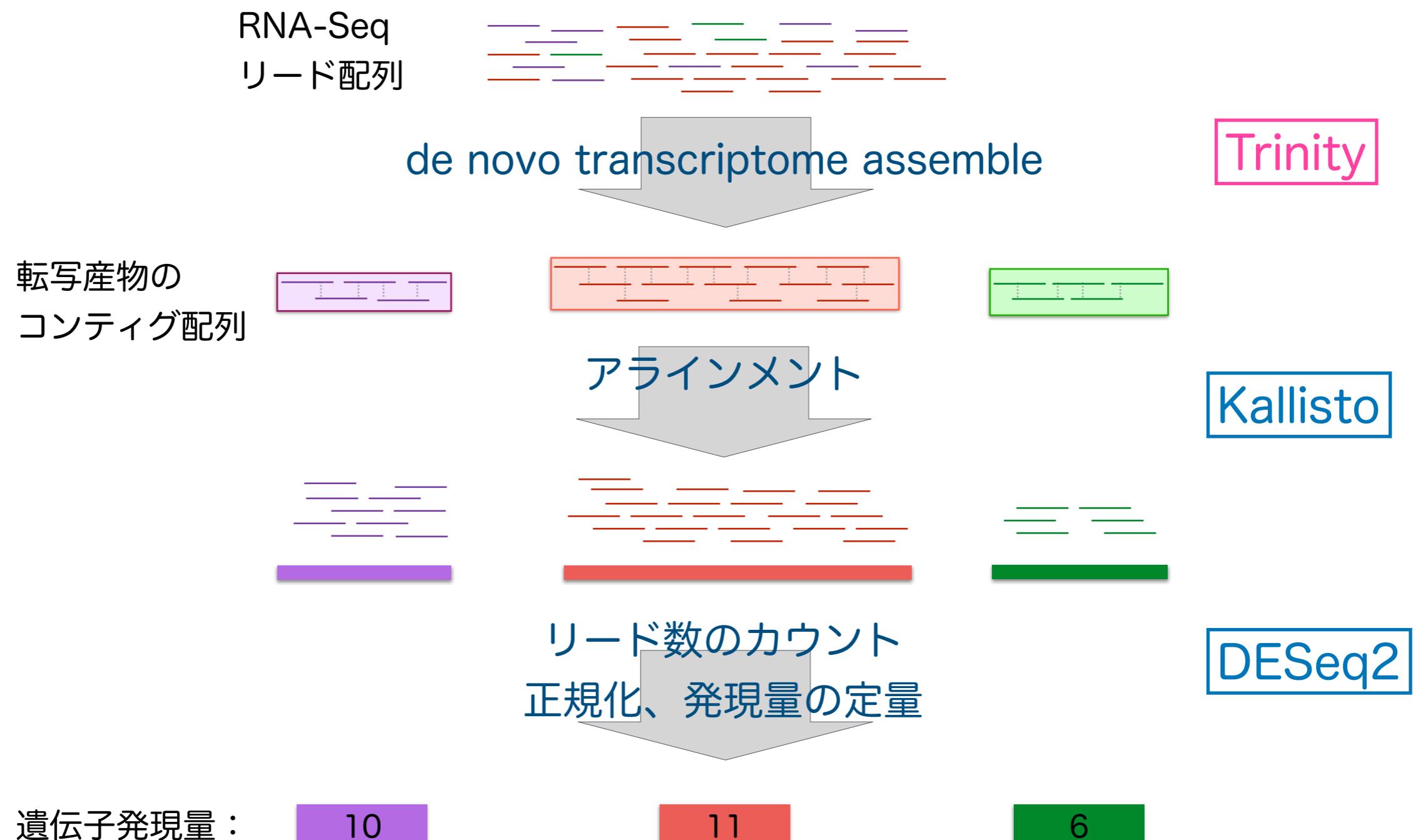
2:00 AM

- 人工気象器で栽培しているイネ（日本晴）の苗の葉身のサンプル。
- 昼（14 時）と夜（2 時）にサンプリング（2 条件）。
- それぞれ3 個体からサンプリング（3 反復）
- Illumina社のStranded mRNA-Seq法でライブラリ調製。
- Illumina社のHiSeq2000による、100bpのPaired-endシーケンシング。

\*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法による遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物配列を再構築する。  
その後、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して遺伝子発現を定量する。



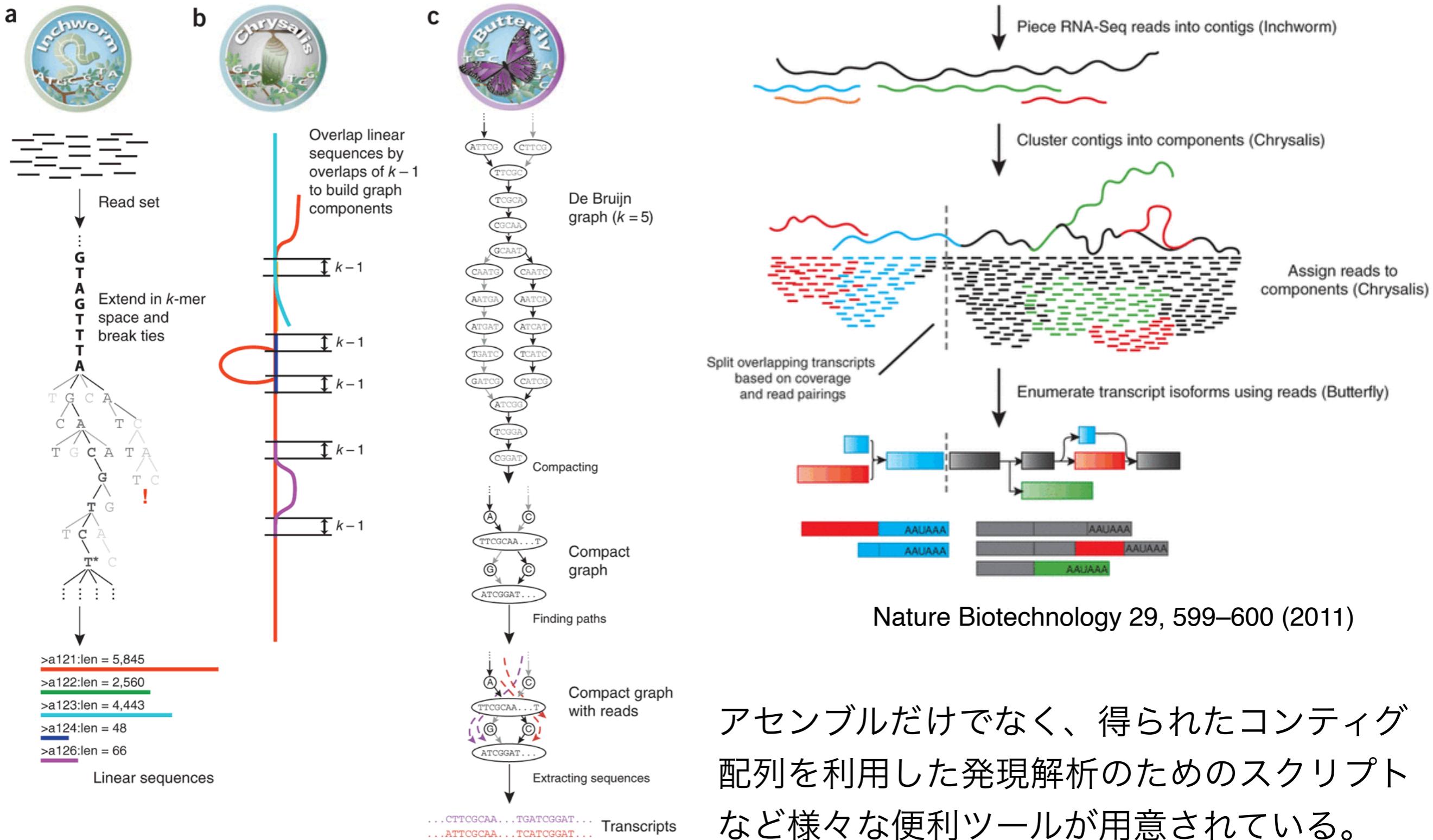
遺伝子発現量：

10

11

6

# Trinity : もっともポピュラーなアセンブルツール



アセンブルだけでなく、得られたコンティグ配列を利用した発現解析のためのスクリプトなど様々な便利ツールが用意されている。

# Step1. リード配列の準備

De novoトランスクリプ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_QC_samples.sh    step6_DE_analysis.sh
step1_prep_fastq.sh  step3_kallisto.sh   step5_EvalAssembly.sh
```

シェルスクリプトが6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_kallisto.sh
$ bash ./step4_QC_samples.sh
$ bash ./step5_EvalAssembly.sh
$ bash ./step6_DE_analysis.sh
```

# 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"になっている必要がある。

```
$ less step1_prep_fastq.sh
```

```
- step1_prep_fastq.sh -
```

```
WorkingDir=$HOME/RNA-Seq  
DataDir=$WorkingDir/data
```

```
echo START: `date`  
  
### Step 1: Prepare input reads  
# 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
```

今回のデータは元々末尾に数字が付いていたので、スクリプトの前半部分はコメントアウトしてある。後半の元データのシンボリックリンクを作成する処理のみが実行される。

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

元データのシンボリックリンク  
を作成する処理

- step1\_prep\_fastq.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
    ln -s ${DataDir}/${DATASET}_r1.org.fastq.gz ./${DATASET}_r1.trinity.fastq.gz
    ln -s ${DataDir}/${DATASET}_r2.org.fastq.gz ./${DATASET}_r2.trinity.fastq.gz
done
```

シェルスクリプトの実行

(実行時間：約1秒)

```
$ bash ./step1_prep_fastq.sh 2>&1 | tee step1_prep_fastq.log
```

RNA-Seqリードファイルのシンボリックリンクが作成されている

```
$ ls
rice_D_rep1_r1.trinity.fastq.gz  rice_N_rep1_r2.trinity.fastq.gz  step1_prep_fastq.sh
rice_D_rep1_r2.trinity.fastq.gz  rice_N_rep2_r1.trinity.fastq.gz  step2_Trinity.sh
rice_D_rep2_r1.trinity.fastq.gz  rice_N_rep2_r2.trinity.fastq.gz  step3_kallisto.sh
rice_D_rep2_r2.trinity.fastq.gz  rice_N_rep3_r1.trinity.fastq.gz  step4_QC_samples.sh
rice_D_rep3_r1.trinity.fastq.gz  rice_N_rep3_r2.trinity.fastq.gz  step5_EvalAssembly.sh
rice_D_rep3_r2.trinity.fastq.gz  sample_info.txt
                                step6_DE_analysis.sh
rice_N_rep1_r1.trinity.fastq.gz  step1_prep_fastq.log
```

# 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 -

```
WorkingDir=$HOME/RNA-Seq
ToolDir=$WorkingDir/tool
TrinityImage=$ToolDir/trinityrnaseq.v2.15.1.simg
CPU=2
MEM=16G

### Step 2: De novo transcriptome assemble by Trinity

singularity exec -e $TrinityImage \
    Trinity \
    --seqType fq \
    --max_memory $MEM \
    --CPU $CPU \
    --samples_file `pwd`/sample_info.txt \
    --SS_lib_type RF \
    --output trinity_out
```

TrinityのSingularityイメージを用いて  
解析を行っている

- 入力ファイル等はsample\_info.txt内  
に記載することで指定している。
- CPUやメモリの使用数に加え、  
Stranded RNA-Seqライブラリのオ  
プションを指定している。

## Step 2. Trinityによるde novo transcriptome assemble

- step2\_Trinity.sh のつづき -

```
singularity exec -e $TrinityImage \
/usr/local/bin/util/TrinityStats.pl \
trinity_out.Trinity.fasta \
> trinity_out.Trinity.fasta.stats
```

Trinityのアセンブル結果（コンティグ配列）  
の統計情報を計算

- sample\_info.txt -

```
$ cat sample_info.txt
```

Day	Day_rep1	rice_D_rep1_r1.trinity.fasta.gz rice_D_rep1_r2.trinity.fasta.gz
Day	Day_rep2	rice_D_rep2_r1.trinity.fasta.gz rice_D_rep2_r2.trinity.fasta.gz
Day	Day_rep3	rice_D_rep3_r1.trinity.fasta.gz rice_D_rep3_r2.trinity.fasta.gz
Night	Night_rep1	rice_N_rep1_r1.trinity.fasta.gz rice_N_rep1_r2.trinity.fasta.gz
Night	Night_rep2	rice_N_rep2_r1.trinity.fasta.gz rice_N_rep2_r2.trinity.fasta.gz
Night	Night_rep3	rice_N_rep3_r1.trinity.fasta.gz rice_N_rep3_r2.trinity.fasta.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。
- ・アセンブル後の発現量の定量や発現変動解析でもこのファイルを使用する。解析全体で1つのsample infoファイルを使い回すことで、サンプルの指定ミスの防止にもなる。

## 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15分)

```
$ bash ./step2_Trinity.sh 2>&1 | tee step2_Trinity.log
```

出力ファイルの確認

```
$ ls | grep trinity_out
trinity_out
trinity_out.Trinity.fasta
trinity_out.Trinity.fasta.gene_trans_map
trinity_out.Trinity.fasta.stats
```

## Step 2. Trinityによるde novo transcriptome assemble

アセンブルされた転写産物（コンティグ）配列の統計情報

```
$ less trinity_out.Trinity.fasta.stats
#####
## Counts of transcripts, etc.
#####
Total trinity 'genes': 688
Total trinity transcripts:      1141
Percent GC: 48.68

#####
Stats based on ALL transcript contigs:
#####

Contig N10: 4571
Contig N20: 3449
Contig N30: 2727
Contig N40: 2295
Contig N50: 1916

Median contig length: 619
Average contig: 1068.61
Total assembled bases: 1219285

#####
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#####
...
```

## Step 2. Trinityによるde novo transcriptome assemble

アセンブルされた転写産物（コンティグ）配列の確認

```
$ less trinity_out.Trinity.fasta
>TRINITY_DN356_c0_g1_i1 len=371 path=[2:0-148 4:149-230 6:231-370]
CGTATGTGGGTTTCAAATTAACTGCAATTGTCTACGGCTAGAGCTCCTCATCCGTG
...
TGCCTTTCTTACCCCTTCTTGTGGTTCTTGAGTGTCTCATGTGTCTGCCTGTTGTGT
GTTTCTTCTG
>TRINITY_DN356_c0_g1_i2 len=371 path=[0:0-179 2:180-328 3:329-370]
CCTCCCCATAACTCATGTCCGAGAAGTGCCTTAGTATTGGTACATTAAACCAAGACGGAAA
...
AGTTTCTTTCTTCTCTCCTTGCCTTCTCTCTCTCTCTCTCTCTCTCTCTCTCTCTCTCT
AACATCTTCCC
```

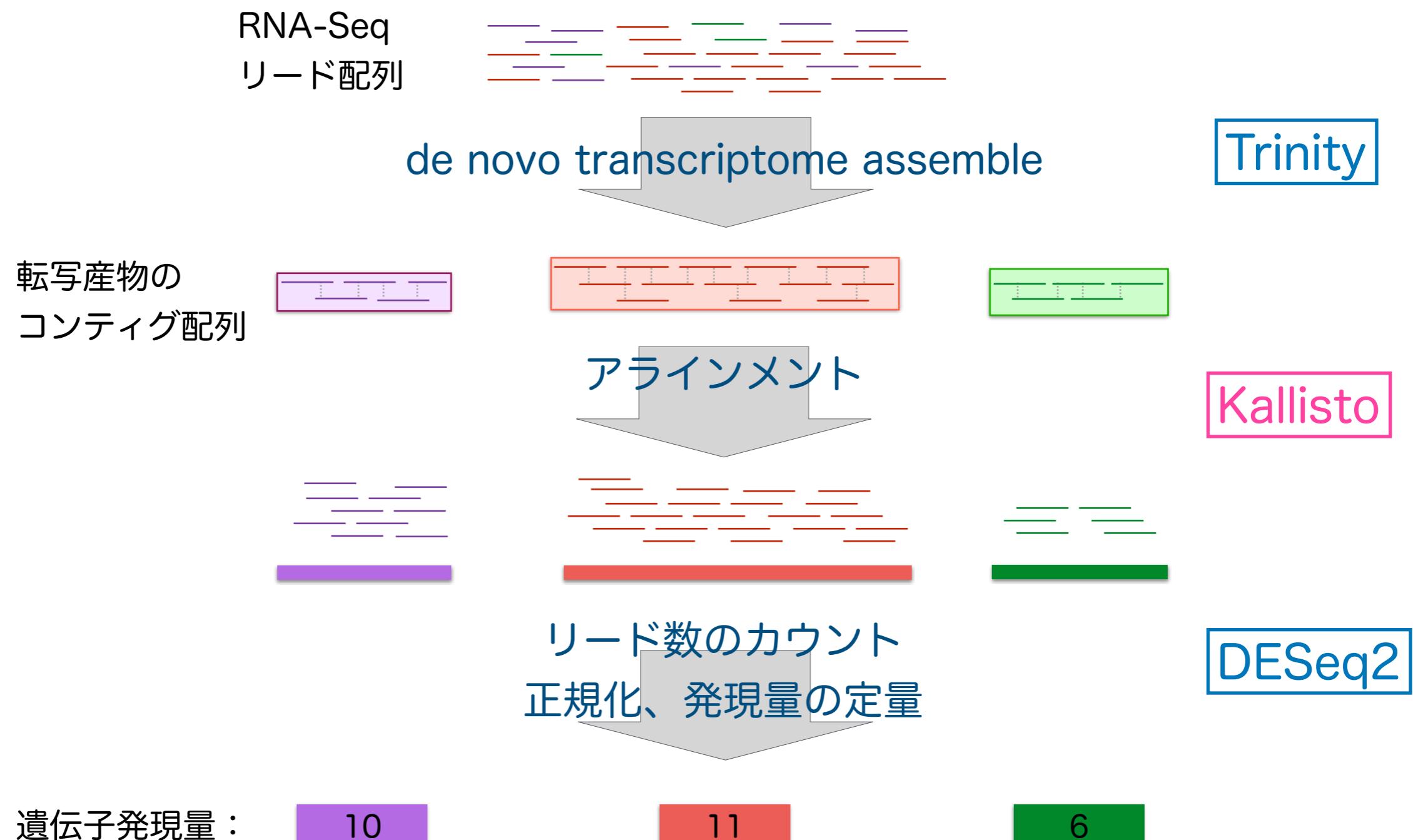
転写産物ID「TRINITY\_DN356\_c0\_g1\_i1」は  
「TRINITY\_DN356\_c0\_g1」遺伝子座のスプライシングバリエント1という意味

アセンブルされた転写産物（コンティグ）配列の遺伝子座とスプライシングバリエントの対応表

```
$ less trinity_out.Trinity.fasta.gene_trans_map
TRINITY_DN356_c0_g1      TRINITY_DN356_c0_g1_i1
TRINITY_DN356_c0_g1      TRINITY_DN356_c0_g1_i2
TRINITY_DN356_c0_g1      TRINITY_DN356_c0_g1_i3
TRINITY_DN356_c0_g1      TRINITY_DN356_c0_g1_i5
TRINITY_DN398_c0_g1      TRINITY_DN398_c0_g1_i1
...
```

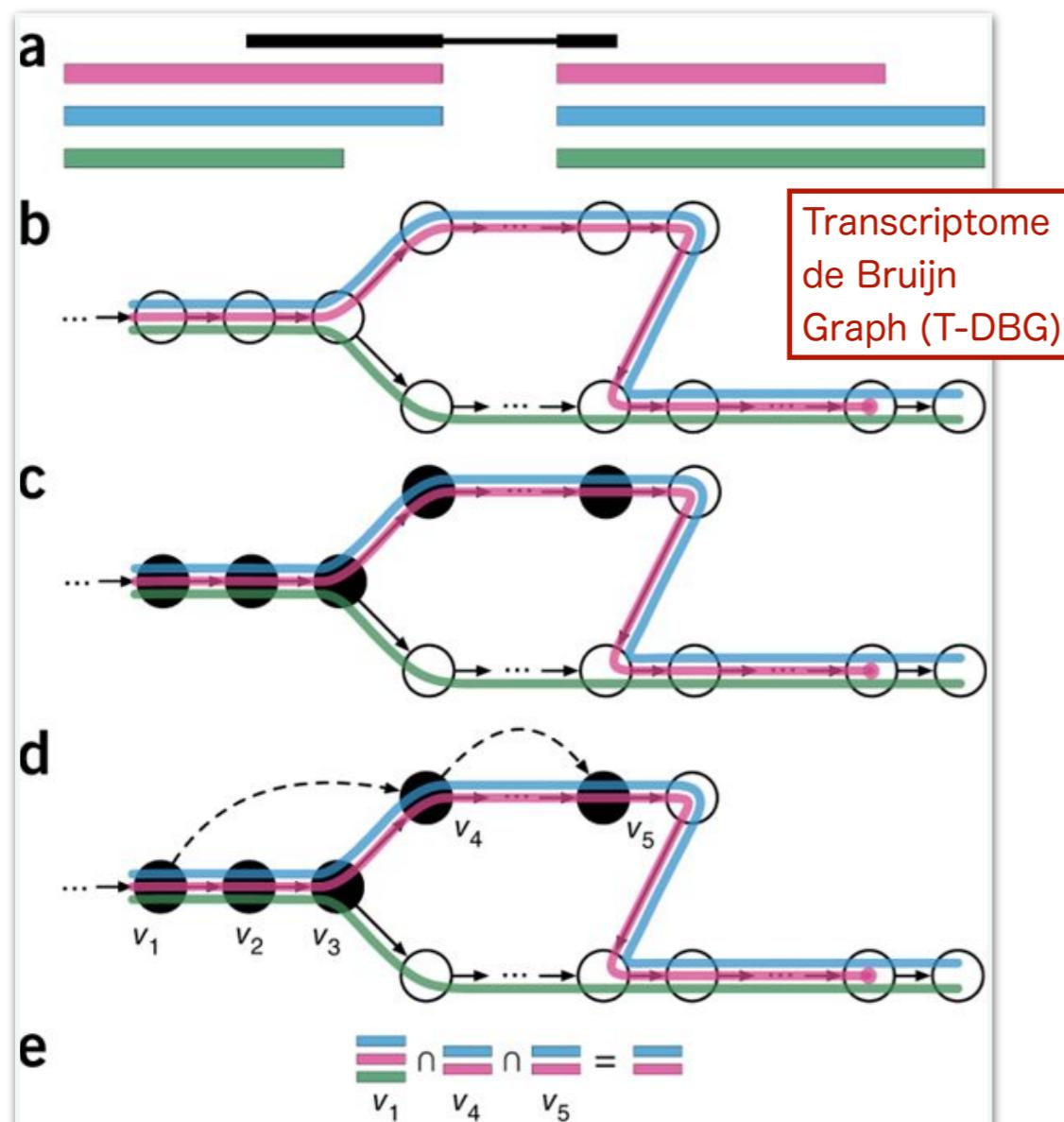
# de novo transcriptome assemble法による遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物配列を再構築する。  
その後、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して遺伝子発現を定量する。



# Step 3. Kallistoによるアラインメントフリーの発現解析

「全転写産物配列」と「RNA-Seqリードデータ」のみを用いて、アラインメントせず！に発現量を定量する方法。計算速度がとにかく早い。



a-b: 全転写産物配列をkmerに分解し、T-DBGを構築。

c-e: RNA-Seqリードも同様にkmerに分解し、T-DBG上のどのパスに由来するか照合、転写産物に対応付け、発現量を推定する。

Figure 1: Overview of kallisto

From Bray NL et al. 2016 Nature Biotech.

# Step 3. Kallistoによるアラインメントフリーの発現解析

Kallistoによって各Trinityコンティグの遺伝子発現量を定量するスクリプト

```
$ less step3_kallisto.sh
```

- step3\_kallisto.sh -

```
WorkingDir=$HOME/RNA-Seq_all
ToolDir=$WorkingDir/tool
TrinityImage=$ToolDir/trinityrnaseq.v2.15.1.simg

### Step 3: Estimating transcript abundance

singularity exec -e $TrinityImage \
/usr/local/bin/util/align_and_estimate_abundance.pl \
--transcripts trinity_out.Trinity.fasta \
--est_method kallisto \
--seqType fq \
--SS_lib_type RF \
--samples_file sample_info.txt \
--thread_count $CPU \
--gene_trans_map trinity_out.Trinity.fasta.gene_trans_map \
--prep_reference
```

Trinityのコンティグ配列と元の  
RNA-Seqリードを指定して、  
Kallistoによる発現量の定量を実行

# Step 3. Kallistoによるアライメントフリーの発現解析

- step3\_kallisto.sh のつづき -

```
# build transcript and gene expression matrices
```

```
ls */abundance.tsv > kallisto_abundance_tsv.list
```

Kallisto の全 abundance ファイルの  
リストを作成

```
singularity exec -e $TrinityImage \  
 /usr/local/bin/util/abundance_estimates_to_matrix.pl \  
 --est_method kallisto \  
 --gene_trans_map trinity_out.Trinity.fasta.gene_trans_map \  
 --name_sample_by_basedir \  
 --quant_files kallisto_abundance_tsv.list
```

Kallistoの結果から様々な発現  
データのマトリックスを作成

```
# counting numbers of expressed transcripts or genes
```

TPMとコンティグ数の統計情報の作成

```
singularity exec -e $TrinityImage \  
 /usr/local/bin/util/misc/count_matrix_features_given_MIN TPM_threshold.pl \  
 kallisto.isoform.TPM.not_cross_norm \  
 > kallisto.isoform.TPM.not_cross_norm.counts_by_min TPM
```

```
singularity exec -e $TrinityImage \  
 /usr/local/bin/util/misc/count_matrix_features_given_MIN TPM_threshold.pl \  
 kallisto.gene.TPM.not_cross_norm \  
 > kallisto.gene.TPM.not_cross_norm.counts_by_min TPM
```

# Step 3. Kallistoによるアラインメントフリーの発現解析

シェルスクリプトの実行

(実行時間：約10秒)

```
$ bash ./step3_kallisto.sh 2>&1 | tee step3_kallisto.log
```

出力ファイルの確認

```
$ ls -tr
```

..  
trinity\_out.Trinity.fasta.kallisto\_idx ..... Kallisto の index ファイル  
Day\_rep1  
Day\_rep2  
Day\_rep3  
Night\_rep1  
Night\_rep2  
Night\_rep3  
kallisto\_abundance\_tsv.list ..... Kallisto の全 abundance ファイルのパス  
kallisto.isoform.TPM.not\_cross\_norm  
kallisto.isoform.counts.matrix  
kallisto.gene.TPM.not\_cross\_norm  
kallisto.gene.counts.matrix  
kallisto.isoform.TPM.not\_cross\_norm.runTMM.R  
kallisto.isoform.TPM.not\_cross\_norm.TMM\_info.txt  
kallisto.isoform.TMM.EXPR.matrix  
kallisto.gene.TPM.not\_cross\_norm.runTMM.R  
kallisto.gene.TPM.not\_cross\_norm.TMM\_info.txt  
kallisto.gene.TMM.EXPR.matrix  
kallisto.isoform.TPM.not\_cross\_norm.counts\_by\_min TPM  
kallisto.gene.TPM.not\_cross\_norm.counts\_by\_min TPM

各サンプルごとのKallistoの結果が置かれたディレクトリ

kallisto.\*.counts.matrix : リードカウントデータ  
kallisto.\*.TPM.not\_cross\_norm : TPMデータ（非正規化）  
kallisto.\*.TMM.EXPR.matrix : TMM正規化後の発現データ

TPMとcontig数の分布

# Step 3. Kallistoによるアラインメントフリーの発現解析

出力された遺伝子発現量データの確認

遺伝子座ごとのリードカウントデータ

```
$ less kallisto.gene.counts.matrix
```

	Day_rep1	Day_rep2	Day_rep3	Night_rep1	Night_rep2	Night_rep3
TRINITY_DN0_c0_g1	473.55	492.47	496.98	640.14	606.68	673.16
TRINITY_DN0_c0_g2	469.10	437.34	437.53	468.46	512.63	479.02
TRINITY_DN102_c0_g1	0.00	0.00	0.00	89.02	25.73	88.15
TRINITY_DN104_c0_g1	2834.00	2900.27	2880.78	163.89	189.14	198.89
...						

最小TPM値と遺伝子座数の集計結果

```
$ less kallisto.gene.TPM.not_cross_norm.counts_by_min TPM
```

```
neg_min_tpm      num_features
```

```
-165648 1
```

```
-118651 2
```

発現量高い

```
-115546 3
```

```
-80002 4
```

```
...
```

```
-69 683
```

発現量低い

```
-66 684
```

```
0 688
```

# Step 3. Kallistoによるアラインメントフリーの発現解析

遺伝子座ごとのTMM正規化後の発現データ

```
$ less kallisto.gene.TMM.EXPR.matrix
```

	Day_rep1	Day_rep2	Day_rep3	Night_rep1	Night_rep2	Night_rep3
TRINITY_DN0_c0_g1	4479.010	4799.390	4582.656	4303.437	4200.103	4529.423
TRINITY_DN0_c0_g2	4436.959	4262.116	4034.426	3149.308	3548.976	3223.094
TRINITY_DN102_c0_g1	0.000	0.000	0.000	598.467	178.149	593.102
TRINITY_DN104_c0_g1	26805.137	28264.439	26563.418	1101.813	1309.456	1338.269
TRINITY_DN106_c0_g1	88.296	113.462	150.321	219.189	106.317	248.198
TRINITY_DN107_c0_g1	1278.562	1356.603	1107.836	1168.285	1333.927	1513.971
TRINITY_DN107_c0_g2	0.000	0.000	3.109	1365.883	908.338	746.355
TRINITY_DN107_c0_g3	530.028	0.000	221.905	432.090	554.122	583.781
TRINITY_DN108_c0_g1	632.217	345.419	380.414	276.474	381.618	201.320
...						

# Step 4. サンプル間の遺伝子発現比較解析

Trinity付属のスクリプトでサンプル間の遺伝子発現を比較し、サンプルに問題が無いかを確認する。

```
$ less step4_QC_samples.sh
```

- step4\_QC\_samples.sh -

```
WorkingDir=$HOME/RNA-Seq  
ToolDir=$WorkingDir/tool  
TrinityImage=$ToolDir/trinityrnaseq.v2.15.1.simg
```

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

```
### Step 4: Differential expression analysis  
# compare replicates  
singularity exec -e $TrinityImage \  
/usr/local/bin/Analysis/DifferentialExpression/PtR \  
--matrix kallisto.gene.counts.matrix \  
--samples sample_info.txt \  
--log2 \  
--CPM \  
--min_rowSums 30 \  
--compare_replicates
```

「Day」と「Night」それぞれの  
3反復間の比較解析

パラメータの詳細等は以下のページを参照

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/QC-Samples-and-Biological-Replicates>

# Step 4. サンプル間の遺伝子発現比較解析

- step4\_QC\_samples.sh のつづき -

```
# Compare replicates across all samples
singularity exec -e $TrinityImage \
/usr/local/bin/Analysis/DifferentialExpression/PtR \
--matrix kallisto.gene.counts.matrix \
--samples sample_info.txt \
--log2 \
--CPM \
--min_rowSums 30 \
--sample_cor_matrix
```

サンプル間の発現の相関を計算し、クラスタリングを行う

```
# PCA
singularity exec -e $TrinityImage \
/usr/local/bin/Analysis/DifferentialExpression/PtR \
--matrix kallisto.gene.counts.matrix \
--samples sample_info.txt \
--log2 \
--CPM \
--center_rows \
--min_rowSums 30 \
--prin_comp 3
```

全サンプルの発現データを元に主成分分析を行う

# Step 4. サンプル間の遺伝子発現比較解析

シェルスクリプトの実行

(実行時間：約10秒)

```
$ bash ./step4_QC_samples.sh 2>&1 | tee step4_QC_samples.log
```

出力ファイルの確認

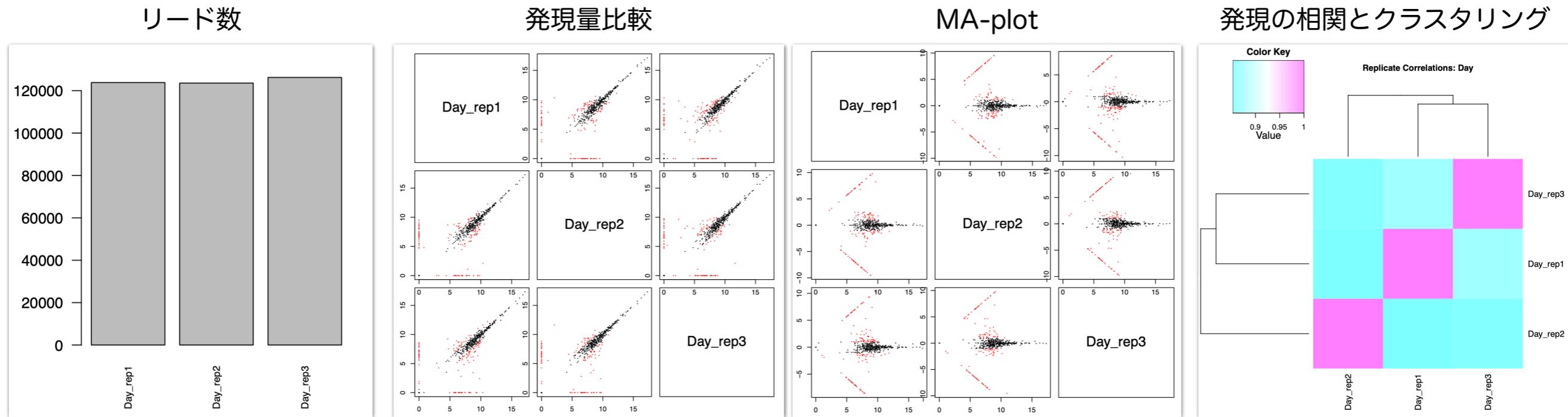
```
$ ls
```

Day.rep\_compare.pdf  
Night.rep\_compare.pdf  
kallisto.gene.counts.matrix.minRow30.CPM.log2.dat  
kallisto.gene.counts.matrix.minRow30.CPM.log2.sample\_cor.dat  
kallisto.gene.counts.matrix.minRow30.CPM.log2.sample\_cor\_matrix.pdf  
kallisto.gene.counts.matrix.R  
kallisto.gene.counts.matrix.minRow30.CPM.log2.centered.dat  
kallisto.gene.counts.matrix.minRow30.CPM.log2.centered.PCA.prcomp.scores  
kallisto.gene.counts.matrix.minRow30.CPM.log2.centered.PCA.prcomp.loadings  
kallisto.gene.counts.matrix.minRow30.CPM.log2.centered.prcomp.principal\_components.pdf

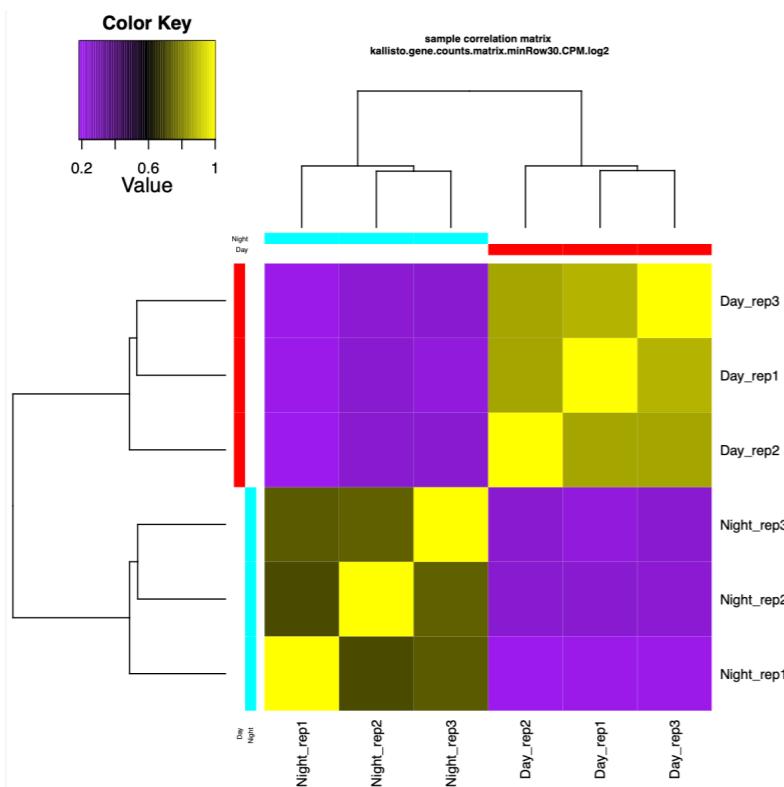
「Day」と「Night」それぞれの3反復間の比較解析結果  
全サンプルの発現データを元にしたクラスタリング結果  
全サンプルの発現データを元にした主成分分析結果

# Step 4. サンプル間の遺伝子発現比較解析

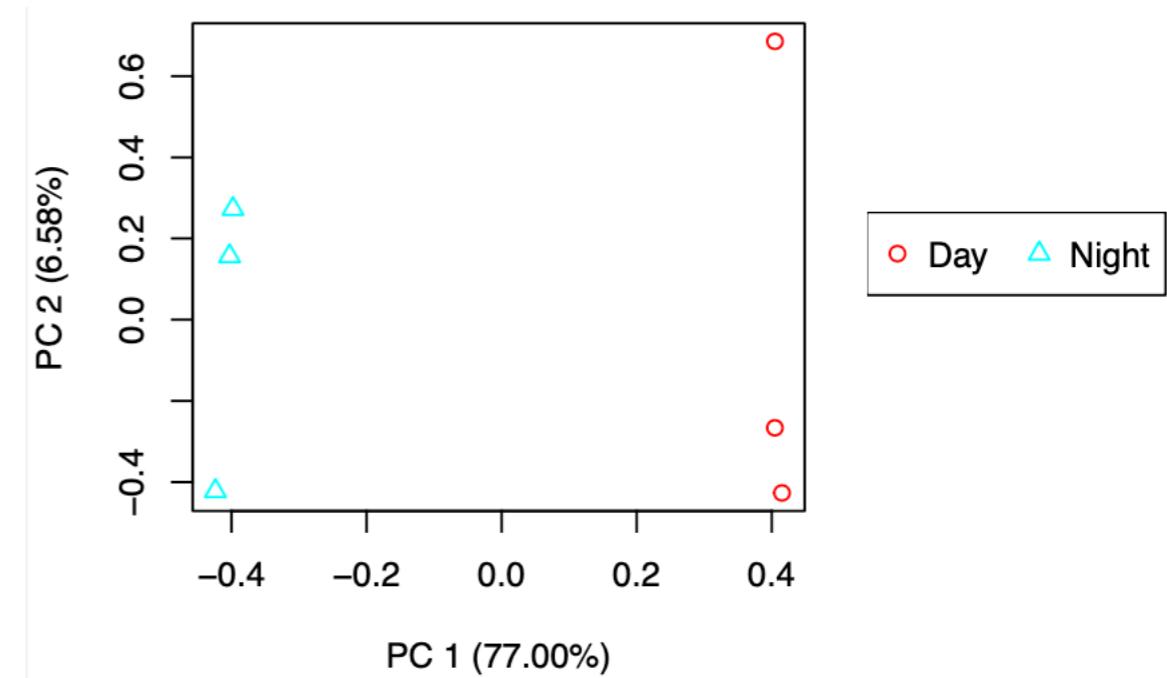
「Day」と「Night」それぞれの3反復間の比較解析結果



全サンプルの発現データの相関とクラスタリング



全サンプルの発現データの主成分分析



# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

アセンブルされた転写産物がどれぐらい正確なものかを検証する。

- 元のリードをコンティグにマッピングし、どれぐらいのリードがアセンブルに貢献しているかを調べる。
- BLAST検索により既知のタンパク質配列がどれぐらい正確にアセンブルされているかを調べる。

```
$ less step5_EvalAssembly.sh
```

- step5\_EvalAssembly.sh -

```
WorkingDir=$HOME/RNA-Seq  
ToolDir=$WorkingDir/tool  
Bowtie2_bin=$ToolDir/bowtie2-2.5.1-linux-x86_64  
Samtools_bin=$ToolDir/samtools-1.18/bin  
BLAST_bin=$ToolDir/ncbi-blast-2.14.1+/bin  
TrinityImage=$ToolDir/trinityrnaseq.v2.15.1.simg
```

使用するツールのパスを変数に格納

```
export PATH=$Bowtie2_bin:$Samtools_bin:$BLAST_bin:$PATH ..... パスの設定
```

```
### Step 5: Evaluation of de novo transcriptome assemblies by Trinity  
# check read representation
```

```
bowtie2-build \  
trinity_out.Trinity.fasta trinity_out.Trinity.fasta .....
```

Trinityでアセンブルされた  
コンティグ配列のBowtie用  
のindexを作成

コンティグの評価方法についての詳細は以下のページを参照

<https://github.com/trinityrnaseq/trinityrnaseq/wiki/Transcriptome-Assembly-Quality-Assessment>

# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

- step5\_EvalAssembly.sh のつづき -

```
bowtie2 \
-p $CPU \
-q \
--no-unal \
-k 20 \
-x trinity_out.Trinity.fasta \
-1
rice_D_rep1_r1.trinity.fastq.gz, rice_D_rep2_r1.trinity.fastq.gz, r
ice_D_rep3_r1.trinity.fastq.gz, rice_N_rep1_r1.trinity.fastq.gz, ri
ce_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, r
ice_D_rep3_r2.trinity.fastq.gz, rice_N_rep1_r2.trinity.fastq.gz, ri
ce_N_rep2_r2.trinity.fastq.gz, rice_N_rep3_r2.trinity.fastq.gz \
2> align_stats.txt \
| samtools sort \
-@ $CPU \
-o bowtie2_read_to_contig.sort.bam

samtools index bowtie2_read_to_contig.sort.bam
samtools faidx trinity_out.Trinity.fasta
```

Bowtie2によって、元のRNA-Seqリードをアセンブルされたコンティグ配列にマッピング

統計情報のファイル出力とBAM形式への変換

IGV等で閲覧する際に必要なアラインメントとコンティグ配列のindexを作成

# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

- step5\_EvalAssembly.sh のつづき -

```
# homology search against rice protein sequences
ln -s ../data/rice_protein.fa

makeblastdb \
-dbtype prot -in rice_protein.fa

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

singularity exec -e $TrinityImage \
/usr/local/bin/util/misc/blast_outfmt6_group_segments.pl \
blastx_trinity_to_rice_protein.outfmt6 \
trinity_out.Trinity.fasta \
rice_protein.fa \
> blastx.outfmt6.grouped

singularity exec -e $TrinityImage \
/usr/local/bin/util/misc/
blast_outfmt6_group_segments.tophit_coverage.pl \
blastx.outfmt6.grouped \
> blastx_TopHitCoverage.grouped.txt
```

既知のタンパク質配列を用意し、BLAST用のDBを作成

コンティグ配列（塩基配列）をクエリとした、既知のタンパク質配列に対するBLASTXによる相同性検索

複数のHSPに分かれてしまっているアライメントをグルーピング

既知タンパク質配列に対するコンティグ配列のカバー率を計算し、集計

# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

シェルスクリプトの実行

(実行時間：約5分)

```
$ bash ./step5_EvalAssembly.sh 2>&1 | tee step5_EvalAssembly.sh
```

出力ファイルの確認

```
$ ls -tr
...
trinity_out.Trinity.fasta.*.bt2
align_stats.txt ..... Bowtie2によるRNA-Seqリードのマッピング結果
bowtie2_read_to_contig.sort.bam
bowtie2_read_to_contig.sort.bam.bai
trinity_out.Trinity.fasta.fai ..... RNA-Seqリードのアラインメントデータ
rice_protein.fa*
blastx_trinity_to_rice_protein.outfmt6
blastx.outfmt6.grouped
blastx_TopHitCoverage.grouped.txt ..... コンティグ配列のindex
...
... ..... BLASTXの結果
... ..... 既知タンパク質配列に対するコンティグ配列
... ..... のカバー率の集計結果
```

## Step 5. アセンブルされた転写産物（コンティグ）配列の評価

元のRNA-Seqリードをコンティグにマッピングしたところ、98%以上のリードがマップされ、リードのほぼ全てがコンティグ配列に貢献していることが分かる。

```
$ less align_stats.txt
756268 reads; of these:
  756268 (100.00%) were paired; of these:
    37558 (4.97%) aligned concordantly 0 times
    41400 (5.47%) aligned concordantly exactly 1 time
    677310 (89.56%) aligned concordantly >1 times
    ----
    37558 pairs aligned concordantly 0 times; of these:
      1381 (3.68%) aligned discordantly 1 time
    ----
    36177 pairs aligned 0 times concordantly or discordantly; of these:
      72354 mates make up the pairs; of these:
        21177 (29.27%) aligned 0 times
        2446 (3.38%) aligned exactly 1 time
        48731 (67.35%) aligned >1 times
98.60% overall alignment rate
```

# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

各コンティグ配列をクエリとして、既知のタンパク質配列に対して相同性検索（BLASTX）をした結果。複数のHSPに分かれているアラインメントをグルーピングしたもの。

```
$ less blastx.outfmt6.grouped
```

#query_acc	target_acc	avg_per_id	min_Evalue	query_match_range	target_match_range	pct_query_len	pct_target_len	max_pct_len
TRINITY_DN356_c0_g1_i1	0s02t0724600-01	100.00	4.69e-58	65-349	600-694	76.82	13.69	76.82
TRINITY_DN356_c0_g1_i3	0s02t0724600-01	98.98	1.79e-44	337-622	600-694	44.41	13.69	44.41
TRINITY_DN356_c0_g1_i5	0s02t0724600-01	98.98	1.37e-44	245-530	600-694	51.81	13.69	51.81
TRINITY_DN323_c0_g1_i1	0s02t0758100-01	100.00	3.76e-56	3-257	236-320	99.22	13.75	99.22
TRINITY_DN363_c1_g1_i1	0s02t0730800-01	100.00	1.26e-116	118-597	241-400	80.40	40.00	80.40
...								

アラインメントのカバー率ごとにコンティグ数を集計した結果。全1,141コンティグのうち、既知のタンパク質配列とアラインメントが得られたものが230コンティグ（約20%）あった。

```
$ less blastx_TopHitCoverage.grouped.txt
```

#hit_pct_cov_bin	count_in_bin	>bin_below
100	118	118
90	21	139
80	10	149
70	11	160
60	6	166
50	15	181
40	12	193
30	19	212
20	15	227
10	3	230

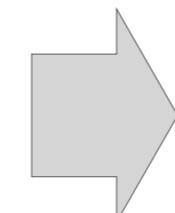
# Step 5. アセンブルされた転写産物（コンティグ）配列の評価

Trinityによるコンティグ配列をゲノム配列にマッピングし、既存の遺伝子アノテーションと比較することでも、その正確性を調べることができる。

既知のタンパク質に対するBLASTXの結果から、  
DTH2遺伝子に対応するコンティグ配列が  
「TRINITY\_DN17\_c0\_g1\_i10」であることが分かる。

```
>TRINITY_DN17_c0_g1_i10 len=2182 path=[0:0-253 2:254-254 6:255-951 8:952-2181]
CCCCCTCCCCCACTCGCTCGTCTTCGTTCTCGCTCGCGCATCGTCTTGTGC
TTCTCCACCTGTTGTTCCATTTCCTCGCTTATCTCTTCCCTCGCTCGCTCCCTTC
GCGCACCAAGAGTCGTTTTTAATTTCGAGTTTCCGTGGAAGATTCTGATACCTGT
CCCTCGTCGTCGTCCTCCGCCGGCCGACGACCGCAGCAGGGGTGGATCCGG
GGCGGCCGCGATTGATCCTGGTAGGATCCCAGTGAATTCAACTGTTGAACAAAAAGCC
...
TGAGATGTCGAATCACTCTACCATCTGTGCATTAATCAAATACAATACATGGGCCT
ACCGCACGACGACGAACGACCTGCTCTGTTGTGATGTTGTGCGGCTGTCAGGC
TGGATCTCTAGGTAACATGATTAATTATACTACTGATGGAATTGACCGCTGTAATATT
ATTTACGAATCACAATCTGCCGTGGATTGTTCTCTGCGTGTATGTATGTATATAG
ATGGATGGCTGTTAAGAAAATTGTGACGATTAAAGTTACCTTGAAATGATAGT
AAAATGCTGATTGATCTGTAG
```

既存の遺伝子アノテーションと完全に一致する  
転写産物構造が得られた



RAP-DBのBLAT機能で  
ゲノム配列にマッピング

## BLAT

### Query sequence

Enter query sequence in FASTA format

```
>TRINITY_DN17_c0_g1_i10 len=2182 path=[0:0-253 2:254-254
6:255-951 8:952-2181]
CCCCCTCCCCCACTCGCTCGTCTTCGTTCTCGCTCGCGCATCGTCTTGTGC
TTCTCCACCTGTTGTTCCATTTCCTCGCTTATCTCTTCCCTCGCTCGCTCCCTTC
GCGCACCAAGAGTCGTTTTTAATTTCGAGTTTCCGTGGAAGATTCTGATACCTGT
CCCTCGTCGTCGTCCTCCGCCGGCCGACGACCGCAGCAGGGGTGGATCCGG
GGCGGCCGCGATTGATCCTGGTAGGATCCCAGTGAATTCAACTGTTGAACAAAAAGCC
...
TGAGATGTCGAATCACTCTACCATCTGTGCATTAATCAAATACAATACATGGGCCT
ACCGCACGACGACGAACGACCTGCTCTGTTGTGATGTTGTGCGGCTGTCAGGC
TGGATCTCTAGGTAACATGATTAATTATACTACTGATGGAATTGACCGCTGTAATATT
ATTTACGAATCACAATCTGCCGTGGATTGTTCTCTGCGTGTATGTATGTATATAG
ATGGATGGCTGTTAAGAAAATTGTGACGATTAAAGTTACCTTGAAATGATAGT
AAAATGCTGATTGATCTGTAG
```

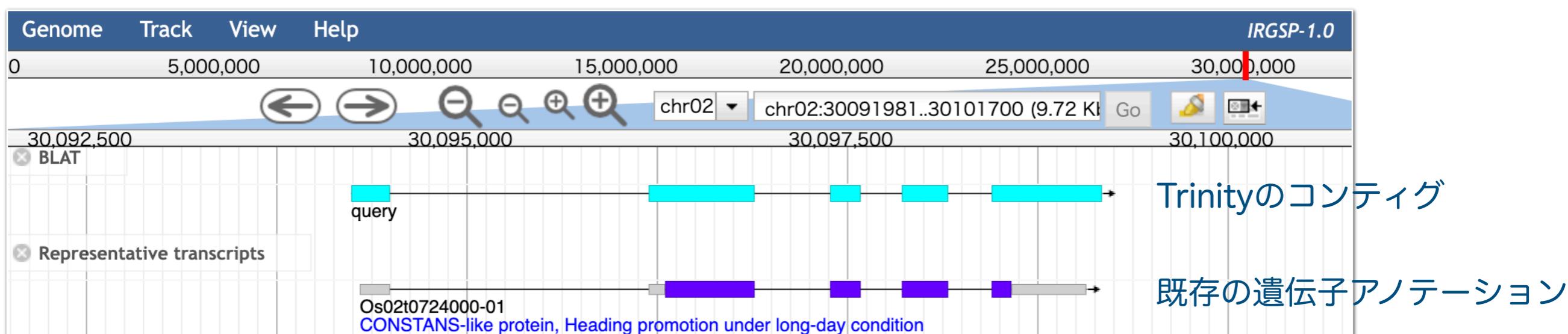
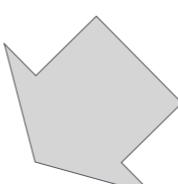
Upload FASTA file ファイルを選択 選択されていません

### Target sequence

Rice genome sequence (Os-Nipponbare-Reference-IRGSP-1.0)

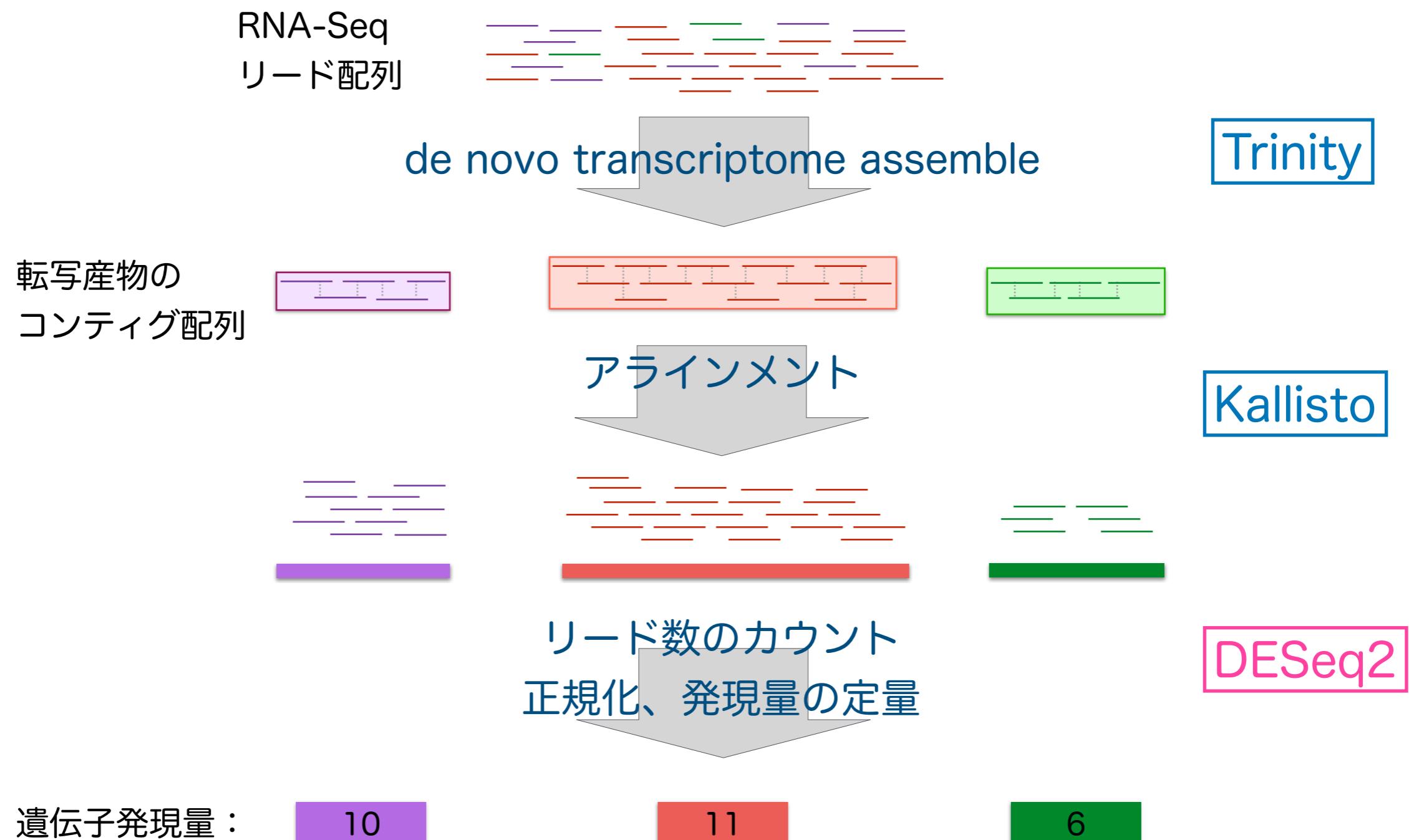
Submit

Reset



# de novo transcriptome assemble法による遺伝子発現解析

まず始めに、RNA-Seqリード配列から転写産物配列を再構築する。  
その後、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して遺伝子発現を定量する。



遺伝子発現量：

10

11

6

# Step 6. 発現変動するコンティグ配列の検出

KallistoのカウントデータとDESeq2を用いて、昼と夜で発現変動する遺伝子（コンティグ）を検出する。

```
$ less step6_DE_analysis.sh
```

- step6\_DE\_analysis.sh -

```
WorkingDir=$HOME/RNA-Seq  
ToolDir=$WorkingDir/tool  
TrinityImage=$ToolDir/trinityrnaseq.v2.15.1.simg  
  
### Step 6: Running differential expression analysis  
singularity exec -e $TrinityImage \  
/usr/local/bin/Analysis/DifferentialExpression/run_DE_analysis.pl \  
--matrix kallisto.gene.counts.matrix \  
--method DESeq2 \  
--samples_file sample_info.txt \  
--output DESeq2_out
```

■ 使用するツールのパスを変数に格納

■ カウントデータを指定して、  
DESeq2によるDEG解析を実行

# Step 6. 発現変動するコンティグ配列の検出

- step6\_DE\_analysis.sh のつづき -

```
# Extracting and clustering differentially expressed genes
cd DESeq2_out ..... DESeq2解析の出力ディレクトリに移動

singularity exec -e $TrinityImage \
/usr/local/bin/Analysis/DifferentialExpression/analyze_diff_expr.pl \
--matrix ../kallisto.gene.TMM.EXPR.matrix \
-P 0.001 ..... p < 0.001かつ23 = 8倍よりも
-C 3 ..... 大きく発現変動する遺伝子を抽出
--samples ../sample_info.txt

# Automatically partitioning genes into expression clusters
singularity exec -e $TrinityImage \
/usr/local/bin/Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl \
-R diffExpr.P0.001_C3.matrix.RData \
--Ptree 60 ..... DEGを対象にクラスタリングを実行
```

シェルスクリプトの実行

(実行時間：約15秒)

```
$ bash step6_DE_analysis.sh 2>&1 | tee step6_DE_analysis.log
```

# Step 6. 発現変動するコンティグ配列の検出

## 出力ファイルの確認

```
$ ls -tr DESeq2_out
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.Rscript           Volcano, MA-plot
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.count_matrix
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.MA_n_Volcano.pdf
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.samples
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.P0.001_C3.Night-UP.subset
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.P0.001_C3.DE.subset
kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.P0.001_C3.Day-UP.subset
...
diffExpr.P0.001_C3.matrix.log2.centered.sample_cor_matrix.pdf ..... サンプルのクラスタリング結果
diffExpr.P0.001_C3.matrix.log2.centered.genes_vs_samples_heatmap.pdf ... 遺伝子のクラスタリング結果
...
clusters_fixed_P_60.heatmap.heatmap.pdf ..... 遺伝子のクラスタリング結果
diffExpr.P0.001_C3.matrix.RData.clusters_fixed_P_60/               (枝長によるグルーピング結果付き)
```

発現変動  
遺伝子  
リスト

```
$ ls -tr DESeq2_out/diffExpr.P0.001_C3.matrix.RData.clusters_fixed_P_60/
subcluster_2_log2_medianCentered_fpkm.matrix
subcluster_1_log2_medianCentered_fpkm.matrix
__tmp_plot_clusters.R
my_cluster_plots.pdf
```

クラスタリングに基づいてグルーピングされた  
サブクラスターごとの遺伝子リストとプロット

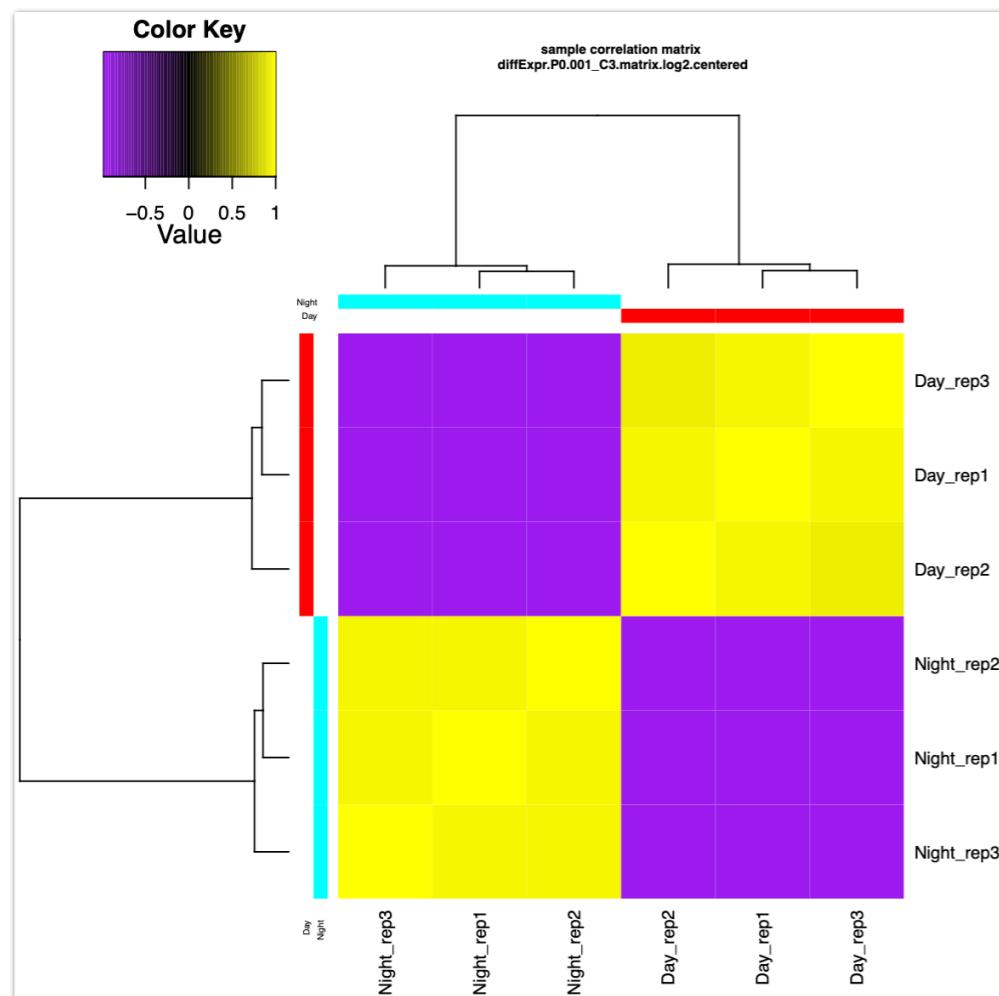
# Step 6. 発現変動するコンティグ配列の検出

昼に発現が有意に上がっている遺伝子リスト

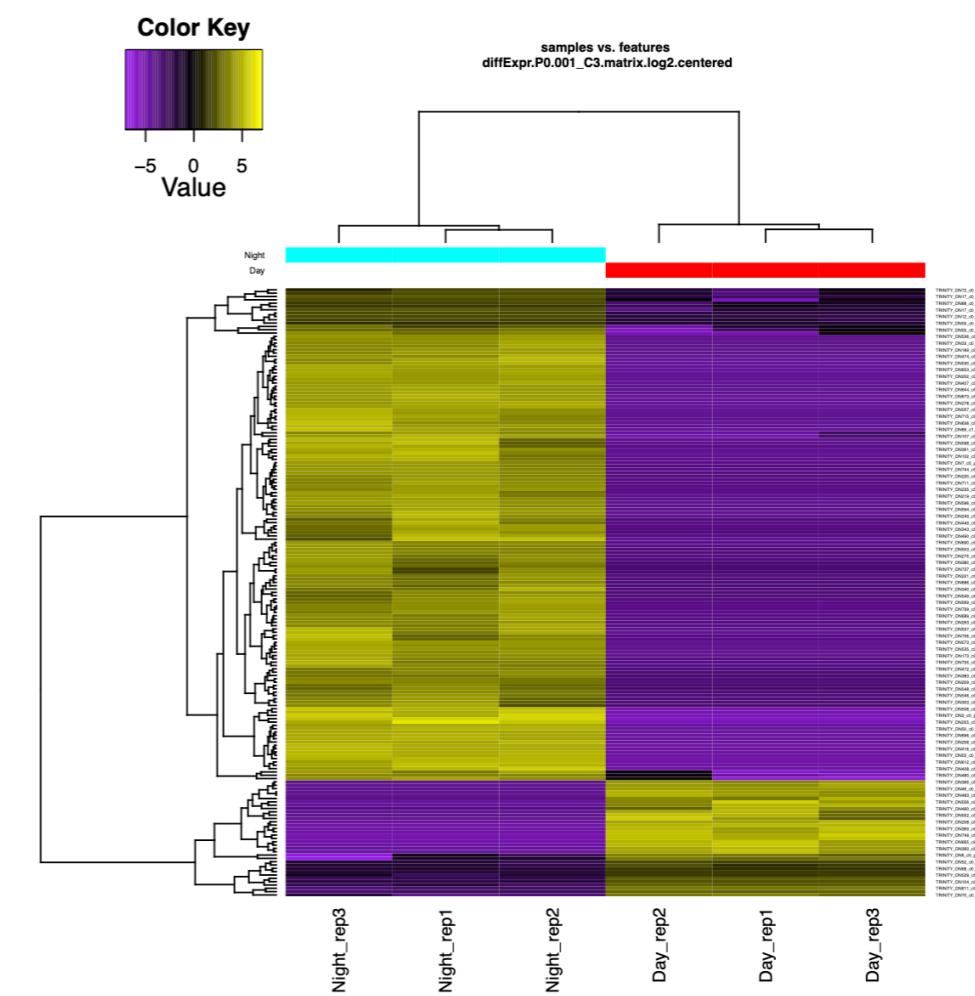
```
$ less DESeq2_out/kallisto.gene.counts.matrix.Day_vs_Night.DESeq2.DE_results.P0.001_C3.Day-UP.subset
```

sampleA	sampleB	baseMeanA	baseMeanB	baseMean	log2FoldChange	lfcSE	stat	pvalue
padj	Day_rep1	Day_rep2	Day_rep3	Night_rep1	Night_rep2	Night_rep3		
TRINITY_DN104_c0_g1	Day	Night	3289.19046522624	153.427845604144	1721.30915541519			
4.42205911412575	0.097957126664436		45.1428013938593	0	0	26805.137		
28264.439	26563.418	1101.813	1309.456	1338.269				
...								

サンプルのクラスタリング結果

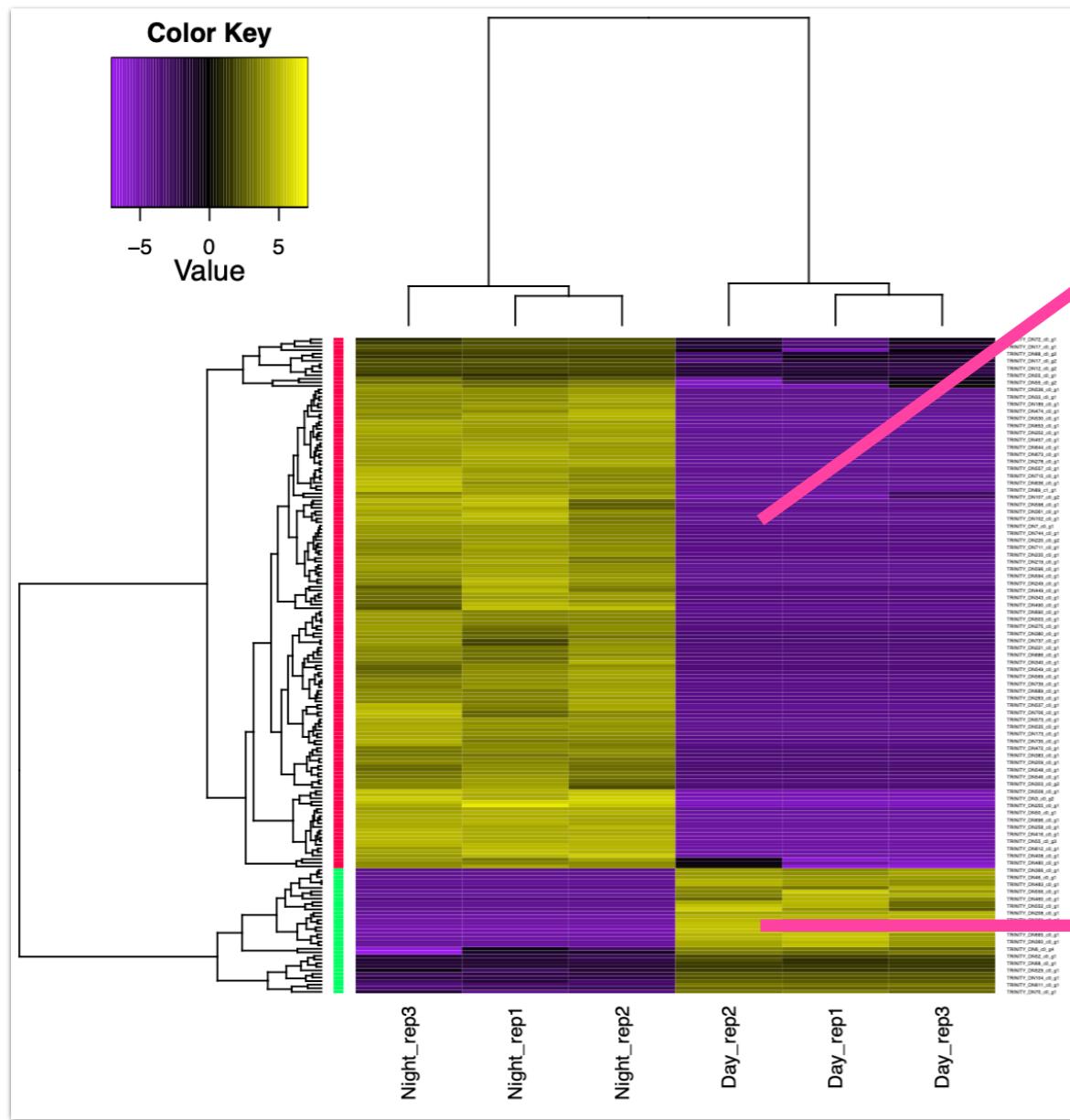


遺伝子のクラスタリング結果



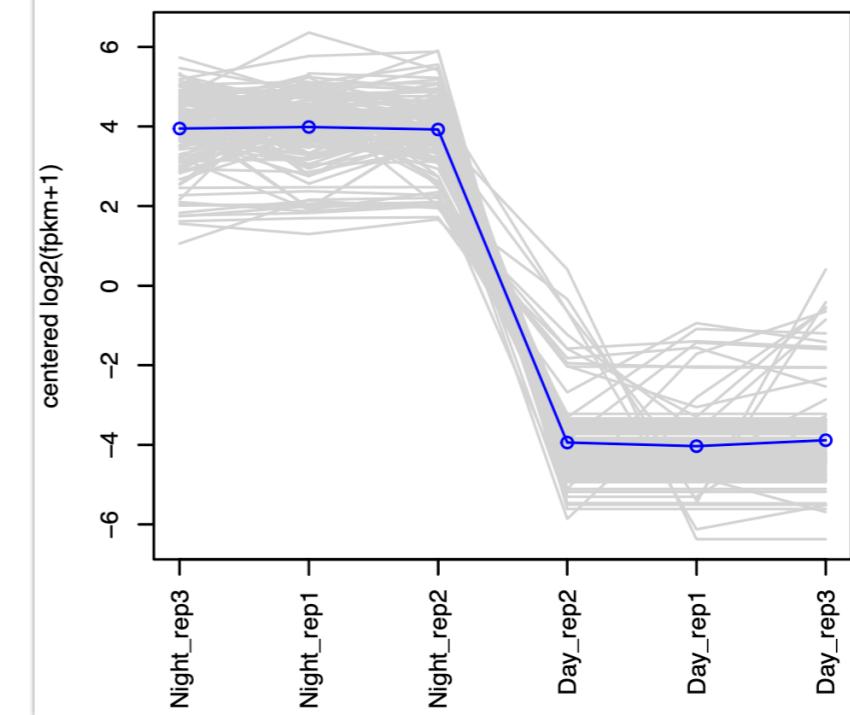
# Step 6. 発現変動するコンティグ配列の検出

遺伝子のクラスタリング結果  
(枝長によるグルーピング結果付き)

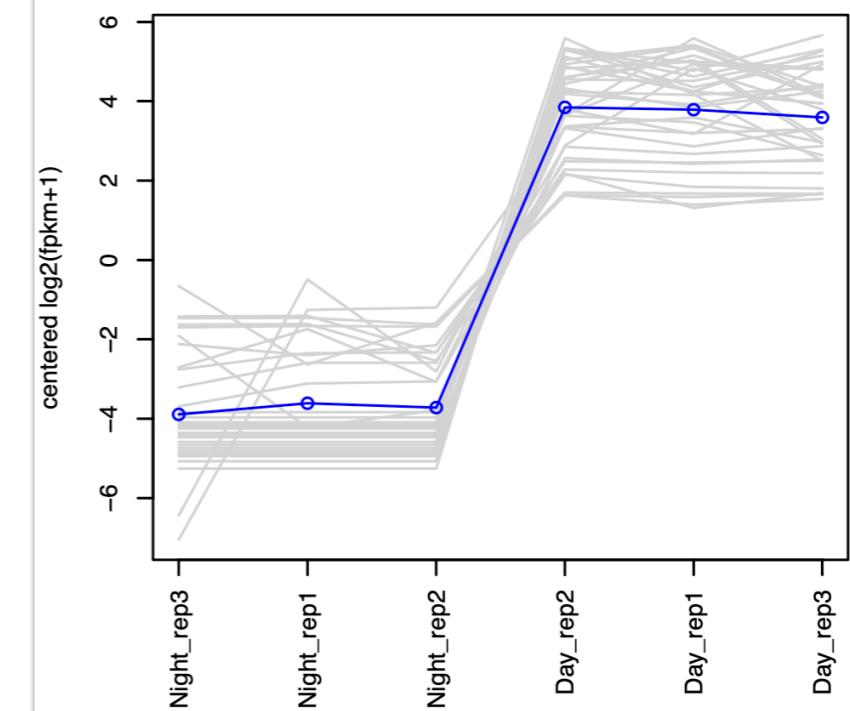


サブクラスターごとのプロット

subcluster\_2\_log2\_medianCentered\_fpkm.matrix, 148 tra



subcluster\_1\_log2\_medianCentered\_fpkm.matrix, 35 tra



# おまけ：ロングリードによる転写産物配列の解読

転写産物配列の全長をロングリードで解読可能  
アセンブルが不要なので精確な遺伝子構造が得られる

## Direct RNA / cDNA sequencing by ONT

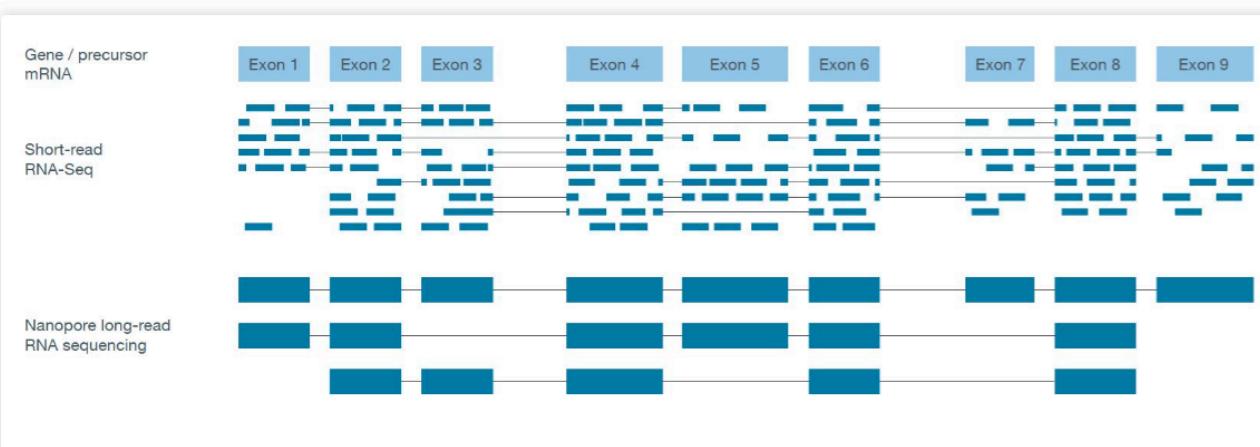
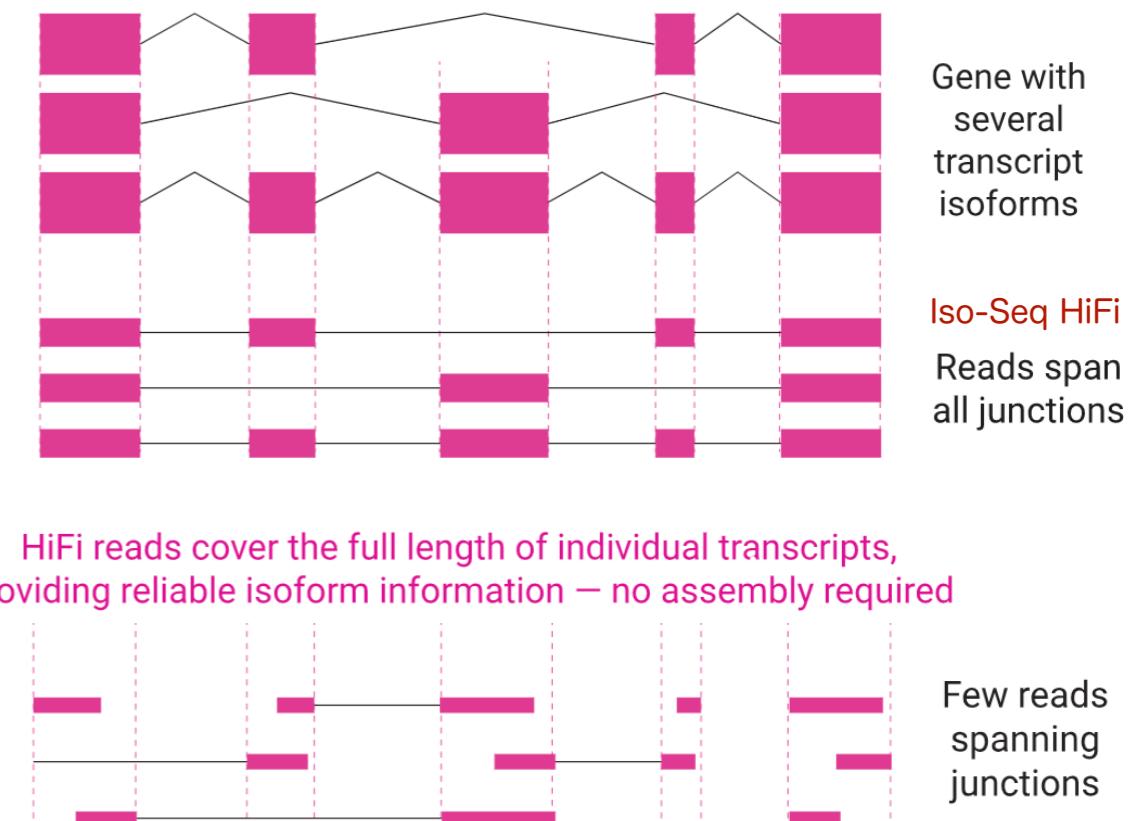


Figure 1: Alternative splicing can give rise to numerous mRNA isoforms per gene, which in turn can alter protein composition and function. The short reads generated by traditional RNA-Seq techniques lose positional information, making the correct assembly of alternative mRNA isoforms challenging. Long nanopore RNA sequencing reads can span full-length transcripts, simplifying their identification and quantification.

<https://nanoporetech.com/applications/investigation/gene-expression>



## Iso-Seq by PacBio HiFi



Computational assembly of fragmented short reads cannot resolve complex isoforms



<https://www.pacb.com/products-and-services/applications/rna-sequencing/>

- ▶ 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
- ▶ 遺伝子機能アノテーション

# 得られた転写産物配列が何者なのか知りたい

II

## コンティグ配列の機能アノテーション

### タンパク質コード領域の予測 TransDecoder

The screenshot shows the GitHub repository page for TransDecoder. The top navigation bar includes links for Product, Solutions, Open Source, Pricing, and a search bar. Below the header, there are buttons for Notifications, Fork (55), Star (236), Sign in, and Sign up. The main content area is titled "TransDecoder (Find Coding Regions Within Transcripts)". It contains a brief description of what TransDecoder does, a list of criteria for identifying coding sequences, and a sidebar with links to various documentation pages.

**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 2 forward 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).
- a PSSM is built/trained/used to refine the start codon prediction.
- optional the putative peptide has a match to a Pfam domain above the noise cutoff score.

TransDecoder の詳細は以下を参照

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

### 遺伝子機能の予測 Trinotate

The screenshot shows the Trinotate homepage. The title "Trinotate" is prominently displayed at the top. Below it are three circular icons representing the stages of the pipeline: "Inchworm" (containing a DNA helix), "Chrysalis" (containing a caterpillar), and "Butterfly" (containing a butterfly). To the right of these are logos for NCBI BLAST, HMMER, Pfam, UniProt, eggNOG, SQLite, and the Gene Ontology. A flowchart at the bottom illustrates the process: RNA-Seq → Trinity → Transcripts/Proteins → Functional Data → Discovery. A large blue button at the bottom is labeled "Automated Higher Order Biological Analysis".

**Trinotate**

Inchworm Chrysalis Butterfly

NCBI BLAST HMMER Pfam UniProt

eggNOG SQLite the Gene Ontology

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

Automated Higher Order Biological Analysis

Trinotate の詳細は以下を参照

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

# Step 1. タンパク質コード領域 (CDS) の予測

Trinityによってアセンブルされた転写産物配列（塩基配列）上のタンパク質コード領域候補を探し、各転写産物ごとに最もそれらしいものを1つ選ぶ。

```
$ less step1_TransDecoder.sh
```

- step1\_TransDecoder.sh -

```
WorkingDir=$HOME/RNA-Seq  
ToolDir=$WorkingDir/tool  
TransDecoderImage=$ToolDir/transdecoder.v5.7.1.simg
```

使用するツールのパスを変数に格納

```
### Step 1: Extract the long open reading frames
```

```
ln -s ../../denovo/trinity_out.Trinity.fasta Trinity.fasta
```

Trinityが出力した  
転写産物配列を準備

```
singularity exec -e $TransDecoderImage \  
 TransDecoder.LongOrfs \  
 -S \  
 -t Trinity.fasta
```

各転写産物配列ごとにCDSの候補  
を探索 (-S で+鎖のみを探索)

```
### Predict the likely coding regions
```

```
singularity exec -e $TransDecoderImage \  
 TransDecoder.Predict \  
 -t Trinity.fasta \  
 --single_best_only
```

coding likelihood scoreに基づいて、  
CDS候補から最もそれらしいものを  
1つ選択

# Step 1. タンパク質コード領域 (CDS) の予測

シェルスクリプトの実行

(実行時間：約1分)

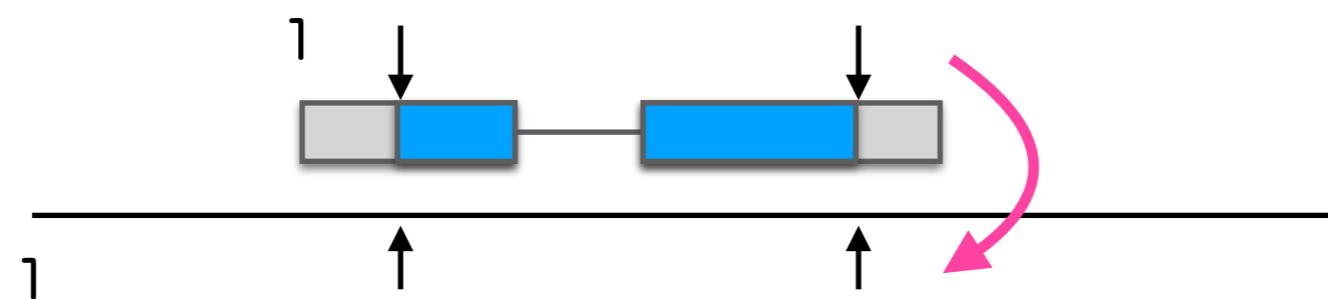
```
$ bash step1_TransDecoder.sh 2>&1 | tee step1_TransDecoder.log
```

出力ファイルの確認

```
$ ls -tr
Trinity.fasta
Trinity.fasta.transdecoder_dir/
Trinity.fasta.transdecoder.gff3
Trinity.fasta.transdecoder.bed
Trinity.fasta.transdecoder.pep
Trinity.fasta.transdecoder.cds
```

様々な形式で出力された  
CDS予測の結果

- Pfam Search (機能ドメイン探索) やBLAST (既知遺伝子との相同性検索) の結果を入力し、  
そのような情報をもつCDSを優先的に選ばせることもできる。
- StringTieのGTFをから転写産物配列を作成して TransDecoder にかけたり、転写産物上での  
CDS位置情報をゲノム上での位置情報に変換するスクリプトなども用意されている。



# Step 1. タンパク質コード領域 (CDS) の予測

## 出力ファイルの確認

```
$ less Trinity.fasta.transdecoder.pep
>TRINITY_DN0_c0_g1_i1.p1 TRINITY_DN0_c0_g1~~TRINITY_DN0_c0_g1_i1.p1 ORF
type:3prime_partial (+),score=160.60 len:748 TRINITY_DN0_c0_g1_i1:483-2726(+)
MHGWREGGEGCGKSRRLVRHMWPVTRVEAAAPPAQGQASPPPRSSVPPPLTTSYPPAP
TPPAAAHKKERVDSPRPASSDSFLKDGREFRVGDCALFQAVEVPPFIGLIRWIEKKEEG
...
SDQGNGHRLIVRFPNPVRSPARSASGGSFEDPSFTGSRASSPVLADKHEQSDRRVKMKTE
NSNPHLGNDTNAESWHSNAVTGASVSEE
>TRINITY_DN0_c0_g1_i10.p1 TRINITY_DN0_c0_g1~~TRINITY_DN0_c0_g1_i10.p1 ORF
type:5prime_partial (+),score=264.00 len:1180 TRINITY_DN0_c0_g1_i10:1-3543(+)
LKSPVSQQLSSSKALTSPVAADAAKSSPVISGSSKLQHMQPGNAVTNLKEQPSKSTGGT
CGSELPNAVKEEKSQQSLNNSQSCSSEHAKTIGSSKEDARSSTAASGVAYKTSGSSSR
...
HTRQYAINLPEGSSTVGHDRNRKWGRQGLDLNSGPGSVDVEVKDDRVTLVRQNFIAAPPH
AFVDEHTRMYQMPPGVGIKRKEPEGSWDAERSSYKQLSWQ*
>TRINITY_DN0_c0_g1_i2.p1 TRINITY_DN0_c0_g1~~TRINITY_DN0_c0_g1_i2.p1 ORF
type:complete (+),score=343.94 len:1619 TRINITY_DN0_c0_g1_i2:474-5333(+)
MHGRRQGGGEGCGNRRLLVRGMWPATRVEAATPPAQGPASPPRLPVPPPLTTPCLPAPT
TPPAAAHNQDWVDSPRPVSPDSFVKDGREFRVGDCALFRAVDVPPFIGLIRWIEKQEEGY
...
PYFPSIAPTLGPAGALPAQHTRQYAINLPEGSSTVGHDSNRKWGRQGLDLNSGPGSVDA
EIKDERVSLPVRQNLITPPHAFGEEHTRMYQMPSVGIKRKEPEGSWDAERSSYKQLSWQ*
```

「TransDecoder.LongOrfs」を実行する際に、「--complete\_orfs\_only」を付けると、M-Stopの complete ORF のみが出力される。

# Step 1. タンパク質コード領域 (CDS) の予測

## 出力ファイルの確認

```
$ less Trinity.fasta.transdecoder.gff3
```

```
...
```

```
TRINITY_DN0_c0_g1_i2 transdecoder gene 1 6011 . + .
```

```
ID=TRINITY_DN0_c0_g1~~TRINITY_DN0_c0_g1_i2.p1;Name="ORF type:complete (+),score=343.94"
```

```
TRINITY_DN0_c0_g1_i2 transdecoder mRNA 1 6011 . + .
```

```
ID=TRINITY_DN0_c0_g1_i2.p1;Parent=TRINITY_DN0_c0_g1~~TRINITY_DN0_c0_g1_i2.p1;Name="ORF type:complete (+),score=343.94"
```

```
TRINITY_DN0_c0_g1_i2 transdecoder five_prime_UTR 1 473 . + .
```

```
ID=TRINITY_DN0_c0_g1_i2.p1.utr5p1;Parent=TRINITY_DN0_c0_g1_i2.p1
```

```
TRINITY_DN0_c0_g1_i2 transdecoder exon 1 6011 . + .
```

```
ID=TRINITY_DN0_c0_g1_i2.p1.exon1;Parent=TRINITY_DN0_c0_g1_i2.p1
```

```
TRINITY_DN0_c0_g1_i2 transdecoder CDS 474 5333 . + 0
```

```
ID=cds.TRINITY_DN0_c0_g1_i2.p1;Parent=TRINITY_DN0_c0_g1_i2.p1
```

```
TRINITY_DN0_c0_g1_i2 transdecoder three_prime_UTR 5334 6011 . + .
```

```
ID=TRINITY_DN0_c0_g1_i2.p1.utr3p1;Parent=TRINITY_DN0_c0_g1_i2.p1
```

```
...
```

----- : 行の区切り

GFF3ファイルには、各転写産物配列上のUTRやCDSの位置が記載されている。

上では1つの転写産物配列 (TRINITY\_DN0\_c0\_g1\_i2) に関する部分を抜き出している。

## Step 2. 機能アノテーションの付与

遺伝子の機能アノテーションには様々な方法があり、これが正解というものはない。

Trinotateは転写産物配列を対象に、既知遺伝子配列に対する相同性検索、タンパク質機能ドメインの探索、Gene Ontologyの付与といった一連の解析をまとめて行ってくれる便利なツールです。

```
$ less step2_Trinotate.sh
```

- step2\_Trinotate.sh -

```
WorkingDir=$HOME/RNA-Seq  
ToolDir=$WorkingDir/tool  
TrinotateImage=$ToolDir/trinotate.v4.0.2.simg
```

使用するツールのパスを変数に格納

```
CPU=2
```

```
### Step 2: Run Trinotate for functional annotation  
# create Trinotate DB
```

```
singularity exec -e $TrinotateImage \  
/usr/local/src/Trinotate/Trinotate \  
--create \  
--db myTrinotate.sqlite \  
--trinotate_data_dir TrinotateDB
```

機能アノテーションに使う様々なデータを取得して、データベース (myTrinotate.sqlite) を構築する。

このステップは外部のウェブサイトから大きなデータをダウンロードしてDB構築をするため、非常に時間がかかります（約1時間）。今回は構築済みのDBが存在している状態から始めるので、このステップはスキップします。

## Step 2. 機能アノテーションの付与

- step2\_Trinotate.sh のつづき -

```
# initialize Trinotate DB  
  
export TRINOTATE_DATA_DIR=$WorkingDir/annot/TrinotateDB  
  
ln -s ../denovo/trinity_out.Trinity.fasta.gene_trans_map Trinity.fasta.gene_trans_map  
  
singularity exec -e $TrinotateImage \  
/usr/local/src/Trinotate/Trinotate \  
--db myTrinotate.sqlite \  
--init \  
--gene_trans_map Trinity.fasta.gene_trans_map \  
--transcript_fasta Trinity.fasta \  
--transdecoder_pep Trinity.transdecoder.pep
```

TrinotateのDBディレクトリのパスの設定  
Trinityがoutputする遺伝子座と転写産物IDの対応リストを準備  
Trinityがアセンブルした転写産物配列や遺伝子座と転写産物IDの対応リスト、TransDecoderが予測したタンパク質配列をDBに格納

Trinityによるトランск립トームアセンブル解析でなくとも

- 転写産物配列（塩基配列）
- タンパク質配列
- 遺伝子座と転写産物IDの対応リスト

があれば、Trinotateを用いた機能アノテーションの付与が可能。

## Step 2. 機能アノテーションの付与

- step2\_Trinotate.sh のつづき -

```
# make and store functional annotation
singularity exec -e $TrinotateImage \
/usr/local/src/Trinotate/Trinotate \
--db myTrinotate.sqlite \
--CPU $CPU \
--transcript_fasta Trinity.fasta \
--transdecoder_pep Trinity.fasta.transdecoder.pep \
--trinotate_data_dir TrinotateDB \
--run "swissprot_blastp swissprot_blastx pfam EggnoGMapper"
```

アノテーションを実行

```
# output functional annotation
singularity exec -e $TrinotateImage \
/usr/local/src/Trinotate/Trinotate \
--db myTrinotate.sqlite \
--report \
> Trinotate_report.tsv
```

機能アノテーション情報をタブ区切りの  
テキストファイルに出力

今回「--run」で指定している解析以外にも

- Infernal :Noncoding RNA Identification
  - signalP: signal peptide prediction
  - Tmhmm: Prediction of transmembrane helices in proteins
- といった解析を追加することも可能。

# Step 2. 機能アノテーションの付与

シェルスクリプトの実行

(実行時間：約40分)

```
$ bash step2_Trinotate.sh 2>&1 | tee step2_Trinotate.log
```

出力ファイルの確認

```
$ ls -tr
...
uniprot_sprot.ncbi.blastp.outfmt6
uniprot_sprot.ncbi.blastx.outfmt6
TrinotatePFAM.out
pfam.log
eggnoG_mapper.emapper.hits
eggnoG_mapper.emapper.seed_orthologs
eggnoG_mapper.emapper.annotations
myTrinotate.sqlite
Trinotate_report.tsv
```

A diagram illustrating the dependencies between the files listed above. A vertical dotted green line connects the bottom two files, `Trinotate_report.tsv` and `myTrinotate.sqlite`. To the right of this line, three labels are positioned: "個々のアノテーション解析の結果ファイル" (Individual annotation analysis results file) next to the top-most dependency, "Trinotate用のDB" (Trinotate database) next to the middle dependency, and "アノテーション結果のテキストファイル" (Annotation results text file) next to the bottom-most dependency.

個々のアノテーション解析の結果ファイル  
Trinotate用のDB  
アノテーション結果のテキストファイル

## Step 2. 機能アノテーションの付与

「Trinotate\_report.tsv」はExcel等で開くことができる

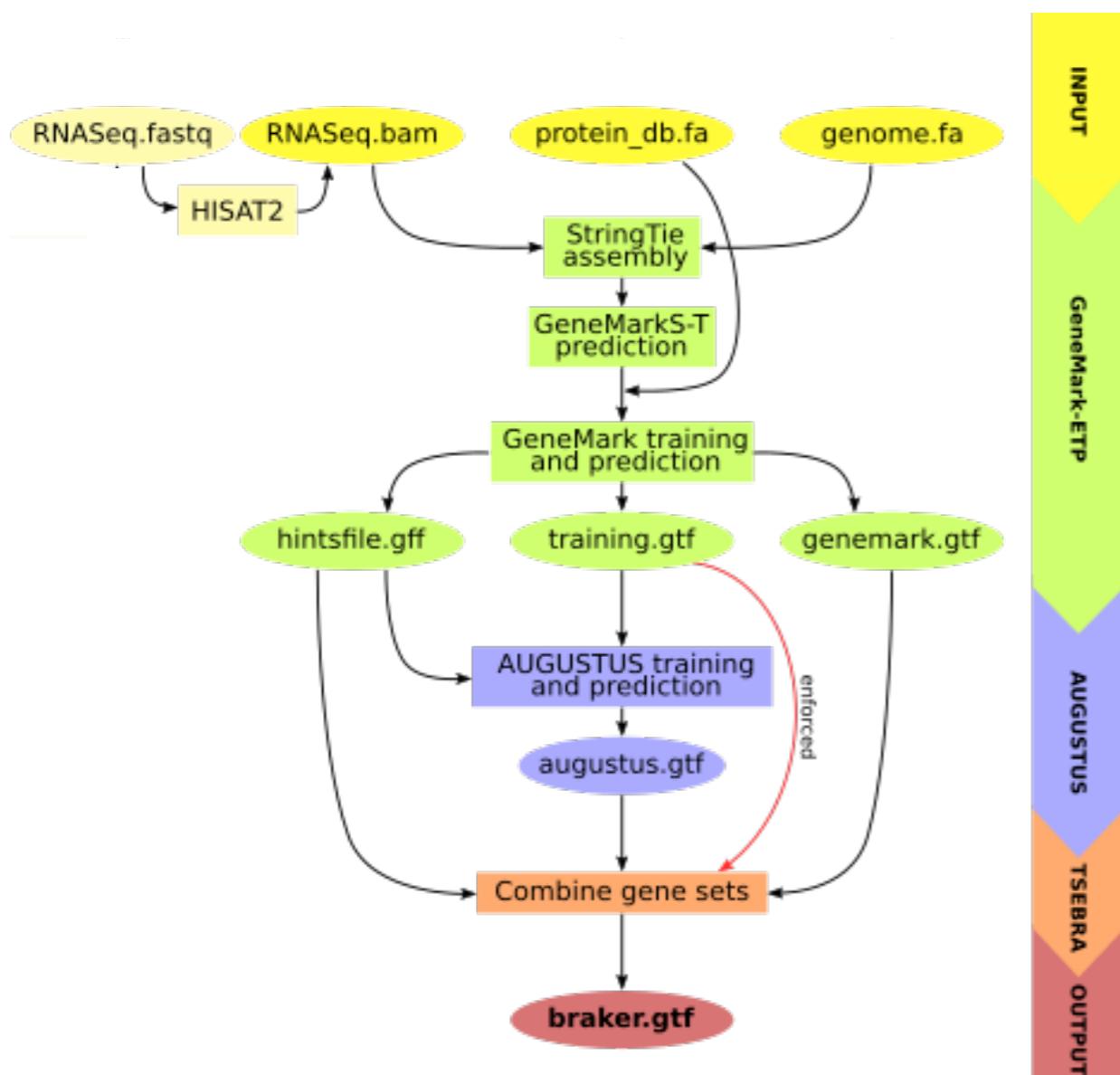
各転写産物配列ごとに様々なアノテーションが付与されている

C22532	A	B	C	G	H	M	N
1	#gene_id	transcript_id	sprot_Top_BLASTX_hit	sprot_Top_BLASTP_hit	Pfam	gene_ontology_BLASTX	gene_ontology_BLASTP
22532	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i1	COL9_ARATH^COL9_ARATH^Q:364-1584,H:1-372^50.238%ID^E:4.13e-86^RecName: Full=Zinc finger protein CONSTANS-LIKE 9;^Eukaryota; Viriplantae; Streptophyta; Embryophyta; Tracheophyta;	COL9_ARATH^COL9_ARATH^Q:PF06203.18^CCT^NULL^4	GO:0005634^cellular_component	GO:0005634^cellular_component	
22533	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i2	COL9_ARATH^COL9_ARATH^Q:.	.	.	GO:0005634^cellular_component.	
22534	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i3	COL9_ARATH^COL9_ARATH^Q:COL9_ARATH^COL9_ARATH^Q:PF06203.18^CCT^NULL^5	GO:0005634^cellular_component	GO:0005634^cellular_component		
22535	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i4	COL9_ARATH^COL9_ARATH^Q:COL9_ARATH^COL9_ARATH^Q:PF06203.18^CCT^NULL^1	GO:0005634^cellular_component	GO:0005634^cellular_component		
22536	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i5	COL9_ARATH^COL9_ARATH^Q:COL9_ARATH^COL9_ARATH^Q:PF06203.18^CCT^NULL^6	GO:0005634^cellular_component	GO:0005634^cellular_component		
22537	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i6	.	.	.	.	.
22538	TRINITY_DN999_c0_g3	TRINITY_DN999_c0_g3_i7	COL9_ARATH^COL9_ARATH^Q:COL9_ARATH^COL9_ARATH^Q:PF06203.18^CCT^NULL^3	GO:0005634^cellular_component	GO:0005634^cellular_component		
22539	TRINITY_DN999_c1_g1	TRINITY_DN999_c1_g1_i1	.	.	.	.	.
22540	TRINITY_DN986_c0_g1	TRINITY_DN986_c0_g1_i6	HT1_ARATH^HT1_ARATH^Q:11	.	.	GO:0005737^cellular_component.	
22541	TRINITY_DN986_c0_g2	TRINITY_DN986_c0_g2_i1	HT1_ARATH^HT1_ARATH^Q:58:HT1_ARATH^HT1_ARATH^Q:55-PF07714.21^PK_Tyr_Ser-1	GO:0005737^cellular_component	GO:0005737^cellular_component		
22542	TRINITY_DN986_c0_g2	TRINITY_DN986_c0_g2_i2	STY13_ARATH^STY13_ARATH^Q:STY13_ARATH^STY13_ARATH^Q:PF07714.21^PK_Tyr_Ser-1	GO:0005737^cellular_component	GO:0005737^cellular_component		
22543	TRINITY_DN986_c0_g2	TRINITY_DN986_c0_g2_i3	HT1_ARATH^HT1_ARATH^Q:58:HT1_ARATH^HT1_ARATH^Q:55-PF07714.21^PK_Tyr_Ser-1	GO:0005737^cellular_component	GO:0005737^cellular_component		
22544	TRINITY_DN986_c0_g2	TRINITY_DN986_c0_g2_i4	HT1_ARATH^HT1_ARATH^Q:58:HT1_ARATH^HT1_ARATH^Q:55-PF07714.21^PK_Tyr_Ser-1	GO:0005737^cellular_component	GO:0005737^cellular_component		
22545	TRINITY_DN986_c0_g2	TRINITY_DN986_c0_g2_i5	HT1_ARATH^HT1_ARATH^Q:58:HT1_ARATH^HT1_ARATH^Q:55-PF07714.21^PK_Tyr_Ser-1	GO:0005737^cellular_component	GO:0005737^cellular_component		
22546	TRINITY_DN900_c0_g1	TRINITY_DN900_c0_g1_i1	FDM1_ARATH^FDM1_ARATH^Q:FDM1_ARATH^FDM1_ARATH^Q:PF03469.18^XH^NULL^23	GO:0009506^cellular_component	GO:0009506^cellular_component		
22547	TRINITY_DN900_c0_g1	TRINITY_DN900_c0_g1_i2	.	.	.	.	.
22548	TRINITY_DN900_c0_g1	TRINITY_DN900_c0_g1_i3	FDM1_ARATH^FDM1_ARATH^Q:FDM1_ARATH^FDM1_ARATH^Q:PF03469.18^XH^NULL^23	GO:0009506^cellular_component	GO:0009506^cellular_component		
22549	TRINITY_DN900_c0_g1	TRINITY_DN900_c0_g1_i4	FDM1_ARATH^FDM1_ARATH^Q:FDM1_ARATH^FDM1_ARATH^Q:PF03469.18^XH^NULL^23	GO:0009506^cellular_component	GO:0009506^cellular_component		
22550	TRINITY_DN900_c0_g1	TRINITY_DN900_c0_g1_i5	.	.	.	.	.
22551	TRINITY_DN900_c1_g1	TRINITY_DN900_c1_g1_i3	H3_NARPS^H3_NARPS^Q:704-2.	.	.	GO:0005730^cellular_component.	
22552	TRINITY_DN900_c1_g1	TRINITY_DN900_c1_g1_i4	H3_NARPS^H3_NARPS^Q:523-1.	.	.	GO:0005730^cellular_component.	
22553	TRINITY_DN900_c1_g1	TRINITY_DN900_c1_g1_i9	H3_NARPS^H3_NARPS^Q:704-2.	.	.	GO:0005730^cellular_component.	
22554	TRINITY_DN900_c1_g2	TRINITY_DN900_c1_g2_i1	H3_NARPS^H3_NARPS^Q:112-5:H3_NARPS^H3_NARPS^Q:1-136PF00125.28^Histone^NUL	GO:0005730^cellular_component	GO:0005730^cellular_component		
22555	TRINITY_DN900_c1_g2	TRINITY_DN900_c1_g2_i2	H3_NARPS^H3_NARPS^Q:111-5:H3_NARPS^H3_NARPS^Q:37-17PF00125.28^Histone^NUL	GO:0005730^cellular_component	GO:0005730^cellular_component		
22556	TRINITY_DN900_c1_g2	TRINITY_DN900_c1_g2_i3	H3_NARPS^H3_NARPS^Q:1318-H3_NARPS^H3_NARPS^Q:1-136PF00125.28^Histone^NUL	GO:0005730^cellular_component	GO:0005730^cellular_component		
22557	TRINITY_DN969_c0_g1	TRINITY_DN969_c0_g1_i3					

# おまけ：BRAKERによる遺伝子予測

BRAKERは、従来の「ab initio 遺伝子予測」に既知タンパク質配列やRNA-Seqデータを追加することで予測精度の向上を図っている。

ゲノム配列、タンパク質配列、RNA-Seqリードのアラインメントデータを入力として実行すると、予測遺伝子のGFFやFASTAファイルを出力してくれる。



```
singularity exec braker3.sif \
braker.pl \
--genome=genome.fa \
--softmasking \
--prot_seq=rice_protein.fa \
--workingdir=braker3_out \
--gff3 \
--bam=./rice_D_rep1.bam,./
rice_D_rep2.bam,./rice_D_rep3.bam,./
rice_N_rep1.bam,./rice_N_rep2.bam,./
rice_N_rep3.bam
```

ゲノム配列  
タンパク質配列

HISAT2による  
RNA-Seqリードの  
アラインメント

結果としては、

- ・タンパク質配列のFASTA
  - ・遺伝子やCDSの座標情報を含むGFF3ファイル
- などが出力される。

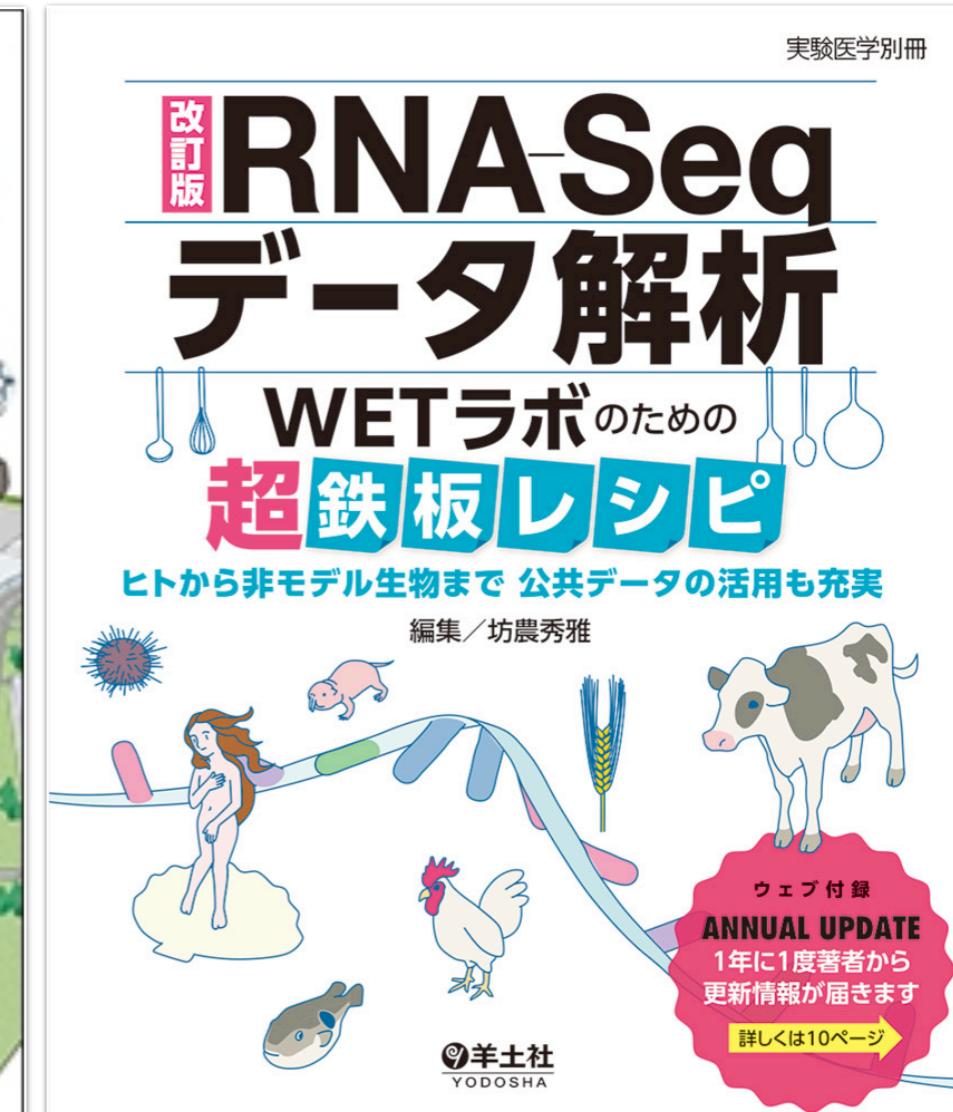
現状ではUTR情報は出力されない

詳しくは以下を参照

<https://github.com/Gaius-Augustus/BRAKER>

## 参考図書・ウェブサイト

# データ解析について本で勉強するなら



- 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。
- NGS解析全般について
- 実際の解析手順やコマンドまで書かれている
- RNA-Seq解析に特化している
- 実際の解析手順やコマンドまで書かれている

# RNA-Seqの実験法についても勉強したいなら

解析結果を正しく解釈するためには、サンプルやライブラリ調製、シーケンシング手法等については知っておいたほうが良いです。



# 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。

まずはGoogleで検索！

Google 検索結果: DESeq2 解析

約 17,000 件 (0.30 秒)

biopapyrus  
https://bi.biopapyrus.jp › de-analysis › 2g-deseq2 :  
二群間比較 (DESeq2) | 一般化線形モデルによる発現変動 ...  
DESeq2 は RNA-seq のリードカウントデータから発現変動遺伝子を検出するためのパッケージである。一般化線形モデルをサポートしているため複雑な解析にも柔軟に対応できる ...

macでインフォマティクス  
https://kazumaxeo.hatenablog.com › 2022/05/08 :  
DESeq2 - macでインフォマティクス - はてなブログ  
2022/05/08 — この論文では、分散とフォールドチェンジの収縮推定を使用して、推定値の安定性と解釈可能性を向上させた、カウントデータの差分解析手法であるDESeq2を ...

はてなブログ  
https://choron81.hatenablog.com › entry › 2023/03/13 :  
【R】遺伝子発現解析を学んでみたい-4② (DESeq2)  
2023/03/13 — お題：遺伝子発現の多寡を解析するパッケージとして、EdgeR、DESeq2、limmaなどいろいろあるらしい。今回もDESeq2をやってみたい。

OlvTools  
https://olvtools.com › documents › deseq2 :  
【DESeq2の使い方】発現変動遺伝子の検出  
次世代シーケンサーを用いてRNA-Seq解析を行うとそれぞれの遺伝子の発現量が得られます。複数のサンプルの遺伝子発現量の定量結果をもとに、グループ間比較を行うこと ...

Zenn  
https://zenn.dev › rchiji › books › viewer :  
DEG - DESeq2で2群間比較  
基本はDESeq2を推奨している。DESeq2; limma-voom; limma-trend. TPMやマイクロアレイデータの場合はlimmaによる解析のみ行える。(voom変換無し、trend指定無し) limmaは ...

shiokoji11235.com  
https://shiokoji11235.com › Bioinformatics :  
【RNA-seq】RNA-seq解析を徹底的に解説！Part2~発現変動 ...  
2023/03/12 — つまりは、発現量するソフトウェア (RSEMやSalmon、kallisto) と発現変動解析をするソフトウェア (edgeRやDESeq2) の架け橋となってくれるということ ...

- 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。

RNA-Seqに関する様々な情報が載せられているポータルサイト

- RNA-Seq Blog <https://www.rna-seqblog.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>

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

- Qiita <https://qiita.com>
- Hatena Blog <http://hatenablog.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>

他にも、GitHubのIssuesに投稿する、X (Twitter) で開発者に直接聞くなど