サンプルジョブの使用方法
(基礎生物学編)

基礎生物学向けのパッケージプログラム利用準備

基礎生物学向けのパッケージプログラムを使用する際は、
事前に設定ファイルを読み込む必要があります。

```
source /apl/bio/etc/bio.sh
```

パッケージプログラムの一覧については
RCCSHP: [利用者向け情報] → [パッケージプログラム一覧（基礎生物学）]
をご確認ください。

サンプルスクリプトの場所

基礎生物学向けの各アプリのサンプルスクリプトは原則として
以下のディレクトリに置いてあります。

- /apl/bio/package/(アプリ名)/samples
- /apl/bio/container/(アプリ名)/samples

例: salmonのサンプル

- /apl/bio/package/salmon/samples
  - sample_salmon_paired.sh
  - sample_salmon_single.sh

```
(base) [ebw@ccfep3 ~]$ cd /apl/bio/package/salmon/samples/
(base) [ebw@ccfep3 samples]$ ls
sample_salmon_paired.sh  sample_salmon_single.sh
(base) [ebw@ccfep3 samples]$
```

サンプルディレクトリ内のファイルについて

サンプルディレクトリ内には、以下のデータが含まれています。

- それを実行するためのジョブスクリプト
  - 入力データの形式（シングルエンド・ペアエンドなど)や計算条件に合わせて複数用意されている場合があります。
    - sample_single.sh: シングルエンド(Single-end)データ
    - sample_paired.sh: ペアエンド(Paired-end)データ
- テスト計算用のインプットデータ
  - 一部スクリプトにのみ存在

サンプルスクリプトが参照するデータについて

基礎生物学向けの各アプリのサンプルスクリプトは、
原則として /apl/bio/data_sample のデータをフルパスで参照しています。
各データについてはNCBIのデータを data_sample 下に保存してあります。

- ecoli
  - GCF_000005845.2
- yeast
  - GCF_000146045.2
- proteins
  - GCF_000001635.27 (Mouse) -> mouse_proteins.faa
  - GCF_000002945.1 (Pombe) -> pombe_proteins.faa
  - GCF_000001735.4 (Arabidopsis) -> arabidopsis_proteins.faa
  - GCF_000009045.1 (BS168) -> bs168_proteins.faa
  - GCF_000209795.2 (Natto) -> natto_proteins.faa
- misc
  - GCF_000005845.2 の一部データ

サンプルスクリプトが参照するデータについて

/apl/bio/data_sample下のデータについて

ecoli / yeast

REFERENCE & ANNOTATION

genome.fasta / transcriptome.fasta
.gff / .gtf (gene structure)

SEQUENCING READS

illumina_R1.fastq / R2.fastq
nanopore_longread.fastq

MAPPING DATA

single.sam / single_sorted.bam

PROTEINS

ecoli_proteins.faa / yeast_proteins.faa

proteins

PROTEINS

arabidopsis_proteins.faa
bs168_proteins.faa
mouse_proteins.faa
natto_proteins.faa
pombe_proteins.faa

misc

MISC / TEMPORARY

ecoli_single.sam
genome.fasta (redundant)

サンプルの実行: 一般的な手順(概要)

サンプルスクリプトの実行の大まかな流れは次のとおり。

1. サンプルスクリプトを自分の書き込める領域にコピー
2. コピーしたサンプルスクリプトを、コピー先で実行
    a. jsub sample.sh のようにしてジョブを投下
    b. sh ./sample.sh のようにしてログインノード上で実行

サンプルスクリプトをご自身のデータに合わせる場合は、
INPUT用のデータパスをご自身のデータパスに差し替えてください。

次のスライドでは、salmonを例にして実行方法手順を解説します

サンプルの実行: 一般的な手順(1)

サンプルスクリプトを自分の書き込める領域にコピー

1. salmon_test というディレクトリをホームディレクトリ(~)直下に作る
2. 1で作成した salmon_test に移動する
3. サンプルスクリプトを salmon_test にコピーする

```
[user@ccfep3 ~]$ mkdir -p ~/salmon_test ←1 ディレクトリ作成
[user@ccfep3 ~]$ cd ~/salmon_test       ←2 ディレクトリ移動
[user@ccfep3 salmon_test]$ cp /apl/bio/package/salmon/samples/* .   ←3 データコピー
[user@ccfep3 salmon_test]$ ls
sample_salmon_paired.sh  sample_salmon_single.sh
```

サンプルの実行: 一般的な手順(2)

コピーしたサンプルスクリプトを、コピー先で実行

1. コピーしたサンプルスクリプトをjsubコマンドで実行する
2. ジョブが終了するのを待ち、出力結果を確認する

```
[user@ccfep3 salmon_test]$ jsub sample_salmon_single.sh
11166909.ccpbs1
```

```
[user@ccfep3 salmon_test]$ ls
sample_salmon_paired.sh          sample_salmon_single.sh.o186015
sample_salmon_single.sh          yeast_salmon_idx
sample_salmon_single.sh.e186015  yeast_single_salmon_out
```

sample_salmon_paired.shの中身

スクリプト詳細は RCCS HPをご確認ください
https://ccportal.ims.ac.jp/node/3953

ご自身の
データに合わせて
赤枠などを
書き換えてください

```
#!/bin/sh
#PBS -l select=1:ncpus=4:mpiprocs=1:ompthreads=4
#PBS -l walltime=00:10:00

if [ ! -z "${PBS_O_WORKDIR}" ]; then
  cd ${PBS_O_WORKDIR}
  NCPUS=${OMP_NUM_THREADS}
else
  NCPUS=4
  export OMP_NUM_THREADS=${NCPUS}
fi

source /apl/bio/etc/bio.sh
module -s purge
module -s load salmon

# ------------------------------------------------------------------------------
# 入力・出力変数の定義 (ここを変更して再利用します)
# ------------------------------------------------------------------------------
# === 入力ファイル ===
# 転写産物(Transcriptome)のリファレンス配列 (FASTA形式)
# ※ Salmonのリファレンスには「ゲノム配列」ではなく「転写産物」を指定してください。
IN_REF="/apl/bio/data_sample/yeast/yeast_transcriptome.fasta"

# シングルエンドの入力 (FASTQ形式)
IN_FASTQ_SINGLE="/apl/bio/data_sample/yeast/yeast_illumina_single.fastq"

# === 出力ディレクトリ ===
# インデックスの出力ディレクトリ
OUT_DIR_IDX="./yeast_salmon_idx"

# 定量結果の出力ディレクトリ
OUT_DIR="./yeast_single_salmon_out"

# === 実行パラメータ ===
# CPUスレッド数
PARAM_THREADS=${NCPUS}

# k-mer長 (リード長に応じて調整してください。デフォルトは31)
PARAM_KMER_LEN=31
```

```
# ------------------------------------------------------------------------------
# 実行セクション
# ------------------------------------------------------------------------------

echo "--- Job started at $(date) ---"
echo "Running Salmon (Single-end mode) on ${PARAM_THREADS} threads (Allocated ${
NCPUS} cores)."
echo ""

# Step 1: インデックスの構築
echo "Step 1: Building Index..."

# -t: 参照配列
# -i: 出力するインデックスのディレクトリ
# -k: k-mer長
# -p: 使用するCPUスレッド数
if [ ! -d "${OUT_DIR_IDX}" ]; then
    salmon index \
        -t "${IN_REF}" \
        -i "${OUT_DIR_IDX}" \
        -k ${PARAM_KMER_LEN} \
        -p ${PARAM_THREADS}
    echo "Indexing finished. Saved to: ${OUT_DIR_IDX}"
else
    echo "Index directory '${OUT_DIR_IDX}' already exists. Skipping indexing ste
p."
fi

echo ""

# Step 2: 発現量の定量
echo "Step 2: Quantifying expression..."

# -i: 作成した(または既存の)インデックスディレクトリ
# -l: ライブラリタイプ (A: 自動検出)
# -r: シングルエンドの入力
# --validateMappings: マッピング精度の向上 (推奨オプション)
# -o: 結果の出力ディレクトリ
# -p: 使用するCPUスレッド数
salmon quant \
    -i "${OUT_DIR_IDX}" \
    -l A \
    -r "${IN_FASTQ_SINGLE}" \
    --validateMappings \
    -o "${OUT_DIR}" \
    -p ${PARAM_THREADS}

echo "Quantification finished. Results saved to: ${OUT_DIR}"
echo ""
echo "--- Job finished at $(date) ---"
```

実演