第四次实验作业

26 APRIL 2021

> 实验背景
>
> > 不论是何种生物，生命活动都绕不开细胞内的中心法则的运行。蛋白作为形态组合最多样的一环承担了各类催化、结构、运输等功能，而 DNA 则是作为部分生物的遗传物质将信息一代代地传递下去，在两者之间，RNA 起到了相当关键的信息传递的作用。相比于 DNA，RNA包含了更多的包括表达量在内的信息，并且多个不同版本的参考之间存在选择的比较，且对于个体的突变尚未由非常完善的工具进行预测[1]；而由于 RNA 是核酸，因此相较于蛋白又可方便地提纯、扩增、测序等定量操作，并且可以避免蛋白质数据库往往随着实验更新、多个 ID 指向同一蛋白、注释信息完整性不一等问题[1]；故而我们将视野投向生物的转录组，通过逆转录 RNA 为 cDNA 并进一步测序，测得各个基因的转录本，可供后续计算基因表达量、差异表达基因分析等处理。
>
> 实验目的
>
> > 在高性能计算集群（High Performance Computing, HPC）环境下，使用并行任务处理，比对 RNA-seq 数据至参考序列，生成 BAM 格式通用序列比对文件供下游分析使用。
>
> 运行环境（简要见下，完整见附件）
>
> ```
> Linux version 3.10.0-957.el7.x86_64
> (mockbuild@kbuilder.bsys.centos.org)
> (gcc version 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC) )
> #1 SMP Thu Nov 8 23:39:32 UTC 2018
> ```

RNA-seq 原始数据比对与转录本计数

实验方法

依照处理转录组数据处理的最佳操作[2]，我们选用 `STAR`[3] 与 `htseq`[4] 的组合来对原始数据进行处理。`STAR` 工具负责对参考基因组构建索引，并将 `fastq` 格式的数据比对到参考基因组上。`htseq` 负责结合基因组注释信息，将比对结果进行过滤，滤除未比对到基因上的结果，并转为对应基因的转录本计数。其中我们还是用了 `samtools`[5] 来对 `STAR` 输出的 BAM 格式比对结果构建索引，以加快

`htseq` 的处理速度。具体实验流程如下。

数据准备

实验所用数据准备如下

```
.
└── in
    ├── genes -> /gpfs1/share/class4/data/chrX_data/genes
    ├── genome -> /gpfs1/share/class4/data/chrX_data/genome
    └── samples -> /gpfs1/share/class4/data/chrX_data/samples
```

其中：

- samples 文件夹下包含了 12 个样本 X 染色体区段中 RNA 的双端测序 fastq 结果文件
- genes 文件夹下包含了 X 染色体的注释信息
- genome 文件夹下存放了 X 染色体的参考序列。

脚本编写

单个样本处理

以下为所用脚本 `pipeline.sh` 的内容，将其写作 `bash` 程序的形式以便单独调用。可传入以下参数：

- `-h/--help` 参数打印帮助信息
- `-v/--verbose` 和 `-q/--quiet` 来指定是否输出额外信息（默认输出）
- `-s/--sample` 指定 in/samples 文件夹下样本 ID （只需输入双端测序结果文件名的共同部分）
- `-t/--threads` 指定最大线程数。

实际执行的流程代码与注释见脚本最后部分，此处使用 PKUHPC 计算集群环境下的 `pkurun-cnlong` 将任务挂载到其中一个 cnlong 节点下。

```
#!/usr/bin/bash

# # # # # # # # # # # # # #
# 以下为对输入参数的处理  #
# # # # # # # # # # # # # #

# 帮助信息
helpInfo="This script is written for processing single-sample RNA-seq fastq data\n\

Index files must be under 'out/genome' directory,\n\
and paired end sequencing result must be in 'in/samples' directory.\n\
This script relies on htseq-count, samtools and STAR,\n\
and they must be included in the path for the script to work properly\n\
\n\
Usage:\n\
\n\
./pipeline.sh [-h] [-v | -q] [-t <threadsNum>] -s <sampleID>\n\
\n\
-h | --help\n\
\t\tPrint this message\n\
-s | --sample\n\
\t\t<sampleID>: Sample ID for processing\n\
-t | --threads\n\
```

```
\t\t<threadsNum>: Max threads that can be utilized\n\
-v | --verbose\n\
\t\tVerbose output\n\
-q | --quiet\n\
\t\tPrint minimum runtime message\n\
\n\
by Nan Huang (huang_nan_2019@pku.edu.cn) April 27th, 2021
"

verboseOutput=0

# 循环处理输入参数
while [[ -n $1 ]]; do
    case "$1" in
    -h | --help)
        # 输出帮助信息
        echo -e ${helpInfo}
        exit
        ;;
    -s | --sample)
        # 指定样本 ID
        sampleID="$2"
        echo "Pending job for sample ${sampleID}"
        shift
        ;;
    -t | --threads)
        # 设定每个任务最大线程数
        threadsPerTask="$2"
        echo "Set threads limit to ${threadsPerTask}"
        shift
        ;;
    -v | --verbose)
        # 指定输出冗余信息
        verboseOutput=0
        shift
        ;;
    -q | --quiet)
        # 指定隐藏冗余信息
        verboseOutput=1
        shift
        ;;
    *)
        # 遇到位置参数则抛出错误
        echo "Unrecognized parameter name $1"
        echo -e ${helpInfo}
        exit
        ;;
    esac
    shift
done

# # # # # # # # # # # # # # # # # # # # # # # #
# 以下为在执行流程前对文件路径和程序的检查  #
# # # # # # # # # # # # # # # # # # # # # # # #

# 检查是否指定样本名
if [[ -z $sampleID ]]; then
    echo -e ${helpInfo}
    echo "Please set sample ID for processing with '-sample'"
    echo "FATAL ERROR: sample ID not set!"
    exit
fi

# 检查是否设置线程数
if [[ -z $threadsPerTask ]]; then
    echo "Using default thread limit of 5"
    threadsPerTask=5
fi
```

```
# 检查环境中是否有 htseq-count 程序
if hash htseq-count 2>/dev/null; then
    if [[ $verboseOutput -eq 0 ]]; then
        echo "Python environment loaded successfully!"
    fi
else
    echo -e ${helpInfo}
    echo "Please activate a Python virtual environment containing htseq-count before r

    exit
fi

# 检查环境中是否有 samtools 程序
if hash samtools 2>/dev/null; then
    if [[ $verboseOutput -eq 0 ]]; then
        echo "Samtools loaded successfully!"
    fi
else
    echo -e ${helpInfo}
    echo "Please include samtools executable file into your path before running this s

    exit
fi

# 检查基因组索引文件是否存在
if [[ ! -f out/genome/Genome ]]; then
    echo -e ${helpInfo}
    echo "Need genme index to run this script, plese generate it manually or using aut

    exit
fi

# 检查输出文件夹是否存在，如无则创建
if [[ ! -d out/samples/${sampleID} ]]; then
    if [[ $verboseOutput -eq 0 ]]; then
        echo "Creating temporary directory!"
    fi
    mkdir out/samples/${sampleID}
fi

# # # # # # # # # # # # # # # # # # # # #
# 以下为实际对样本进行比对等流程操作  #
# # # # # # # # # # # # # # # # # # # # #

# 使用 STAR 将 fastq 结果比对到基因组上
# 结果以排序后的 bam 文件形式呈现
pkurun-cnlong 1 $threadsPerTask \
    "STAR \
--runThreadN ${threadsPerTask} \
--genomeDir out/genome \
--readFilesIn in/samples/${sampleID}_chrX_1.fastq.gz in/samples/${sampleID}_chrX_2.fas

--readFilesCommand zcat \
--twopassMode Basic \
--outSAMattrRGline ID:${sampleID} \
--outSAMstrandField intronMotif \
--outSAMtype BAM SortedByCoordinate \
--outSAMunmapped Within \
--outFileNamePrefix out/samples/${sampleID}/ \
&>out/samples/${sampleID}/star_mapping.log \
&& \
samtools index out/samples/${sampleID}/Aligned.sortedByCoord.out.bam \
&>out/samples/${sampleID}/samtools.log \
&& \
htseq-count \
-m union \
-i gene_id \
```

```
-r pos \
-s no \
-n ${threadsPerTask} \
-p bam \
-o out/samples/${sampleID}/filtered.bam \
-f bam \
out/samples/${sampleID}/Aligned.sortedByCoord.out.bam \
in/genes/chrX.gtf \
&>out/samples/${sampleID}/htseq.log
"

# # 以下为挂载到计算节点上任务的详细注释信息
# STAR                                                                  # 调用 STAR 程序

# --runThreadN ${threadsPerTask}                                        # 指定最大线程数

# --genomeDir out/genome                                                # 基因组索引等相关

# --readFilesIn in/samples/${sampleID}...                               # 双端测序一个样本

# --readFilesCommand zcat                                               # 使用 zcat 来读

# --twopassMode Basic                                                   # 使用 2pass 模式

# --outSAMattrRGline ID:${sampleID}                                     # 在结果中增加额外

# --outSAMstrandField intronMotif                                       # 在结果中指明正反

# --outSAMtype BAM SortedByCoordinate                                   # 以 bam 格式输出

# --outSAMunmapped Within                                               # 在结果中包含未比

# --outFileNamePrefix out/samples/${sampleID}/                          # 指定输出路径

# &>out/samples/${sampleID}/star_mapping.log                            # 输出重定向至日志

# &&                                                                    # 成功执行后继续

# samtools index out/samples/${sampleID}/Aligned.sortedByCoord.out.bam # 为生成的 bam 文

# &>out/samples/${sampleID}/samtools.log                                # 输出重定向至日志

# &&                                                                    # 成功执行后继续

# htseq-count                                                           # 调用 htseq-co

# -m union                                                              # 指定以 union 方

# -i gene_id                                                            # 指定 gtf 文件中

# -r pos                                                                # 指明输入文件按位

# -s no                                                                 # 指明测序结果非链

# -n ${threadsPerTask}                                                  # 指定最大进程数

# -p bam                                                                # 指定输出文件格式

# -o out/samples/${sampleID}/filtered.bam                               # 指定输出文件路径

# -f bam                                                                # 指明输入文件格式

# out/samples/${sampleID}/Aligned.sortedByCoord.out.bam                 # 指明输入文件路径

# in/genes/chrX.gtf                                                     # 指明注释文件路径

# &>out/samples/${sampleID}/htseq.log                                   # 输出重定向至日志
```

循环/并行处理所有样本

在计算节点上，每个用户被允许挂载的任务有一定限制：

1. 每个用户同时运行不超过 20 个任务
2. 任务线程数之和不超过 40
3. 每个线程分配的内存不超过 3.2GB。

因此我们可以在不超过用户任务限制的情况下，结合上述对单个样本处理的脚本，循环挂载多个任务至计算节点，以加快实际运行时间。以下为自动化处理所有样本的脚本 `automate.sh`，包含了如下步骤：

- 检查输入文件夹与依赖脚本（完整性检查缺省）
- 获取样本 ID 列表
- 检查 / 创建输出文件夹
- 检查 / 激活依赖环境
- 构建基因组索引
- 循环挂载对单个任务的处理
- 成功运行

```
#!/usr/bin/bash

# # # # # # # # # # # # # # # # # # # # # # #
# 以下为在执行流程前对文件路径和程序的检查  #
# # # # # # # # # # # # # # # # # # # # # # #

# 帮助信息
helpInfo="This script is written for start processing multiple RNA-seq fastq data.\n\

Paired end sequencing result must be in 'in/samples' directory,\n\
genome reference must be in 'in/genome' directory,\n\
and annotataion data must be in 'in/genes' directory.\n\
This script relies on samtools and STAR,\n\
and they must be included in the path for the script to work properly\n\
\n\
Usage:\n\
\n\
./automate.sh\n\
\n\
Yes, this will process all data files in a loop,\n\
there is no extra operation needed!\n\
\n\
by Nan Huang (huang_nan_2019@pku.edu.cn) April 27th, 2021
"

if [[ -n $1 ]]; then
    echo -e ${helpInfo}
    exit
fi

# 程序运行计时
start=$(date +%s)

# 检查对每个样本处理的脚本是否在同一路径下
if [[ ! -f pipeline.sh ]]; then
    echo -e ${helpInfo}
    echo "This automation script relies on pipeline.sh under the same directory, pleas
```

```
    exit
fi

if [[ ! -d ./in ]]; then
    echo -e ${helpInfo}
    echo "All required files must be in directory named 'in'"
    exit
fi

# 获取全部样本文件 ID，供循环使用
sampleList=$(ls in/samples/ | grep ERR | cut -d "_" -f 1 | uniq)
sampleList=($sampleList)

# 样本文件计数
echo "=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+="
sampleNumber=${#sampleList[@]}
echo "${sampleNumber} sample files detected for process"
for sampleID in ${sampleList[@]}; do
    echo ${sampleID}
done
echo "=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+="

# 激活 Python 环境用于跑 htseq-count
if ! hash htseq-count 2>/dev/null; then
    source /appsnew/source/Python-3.8.6.sh
fi

# 设定每个任务最大线程数
threadsPerTask=20

# 检查输出文件夹是否存在
if [[ ! -d out ]]; then
    echo "Creating output directory!"
    mkdir out
fi

# 检查基因组索引文件夹是否存在
if [[ ! -d out/genome ]]; then
    echo "Creating genome index directory!"
    mkdir out/genome
fi

# 检查比对结果文件夹是否存在
if [[ ! -d out/samples ]]; then
    echo "Creating mapped result directory!"
    mkdir out/samples
fi

# # # # # # # # # # # # #
# 以下为实际流水线操作  #
# # # # # # # # # # # # #

# STAR 构建基因组索引
echo "=========================================="
echo "Start building index for the reference genome"
pkurun-cnlong 1 ${threadsPerTask} \
    "STAR \
--runThreadN ${threadsPerTask} \
--runMode genomeGenerate \
--genomeDir out/genome \
--genomeFastaFiles in/genome/chrX.fa \
--sjdbGTFfile in/genes/chrX.gtf \
--sjdbGTFtagExonParentTranscript Parent \
--sjdbOverhang 100 \
--genomeSAindexNbases 12 \
&>out/genome/star_index.log
"
```

```
echo "============================================="

# # 以下为挂载到计算节点上任务的详细注释信息
# STAR                                  # 调用 STAR 程序
# --runThreadN ${threadsPerTask}        # 指定最大线程数
# --runMode genomeGenerate              # 指明此步为构建基因组索引
# --genomeDir out/genome                # 指定基因组索引文件输出路径
# --genomeFastaFiles in/genome/chrX.fa  # 指明基因组序列文件路径
# --sjdbGTFfile in/genes/chrX.gtf       # 指明基因组注释文件路径
# --sjdbGTFtagExonParentTranscript Parent # 指明输入文件为 gff 式
# --sjdbOverhang 100                    # 指定构建连接数据库时所用两侧序列长度
# --genomeSAindexNbases 12              # 指定合适的后缀索引长度 min(14, log2(GenomeLen

# &>out/genome/star_index.log           # 输出重定向至日志文件

# 等待索引构建完毕
while ((($(sq | wc -l) - 1) > 0)); do
    sleep 5
done

# 重设每个任务最大线程数
threadsPerTask=5

# 任务计数
processedCount=0

# 循环处理每个样本
for sampleID in ${sampleList[@]}; do
    # 如果最大线程数或任务数超出上限，则等待计算资源的释放
    while ((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    echo "---------------------------------------------"
    ./pipeline.sh -s ${sampleID} -t ${threadsPerTask} -q
    processedCount=$((processedCount + 1))
    echo "Job No.${processedCount}/${sampleNumber} in total"
    echo "---------------------------------------------"
done

# 等待所有样本处理完毕
echo "Waiting for the processing of all data"
while ((($(sq | wc -l) - 1) > 0)); do
    sleep 5
done

# 成功处理所有样本
end=$(date +%s)
runtime=$((end - start))
hours=$((runtime / 3600))
minutes=$(((runtime % 3600) / 60))
seconds=$(((runtime % 3600) % 60))
echo "All samples finished! Time elapsed: ${hours}:${minutes}:${seconds} (hh:mm:ss)"
```

结果

完整运行 `automate.sh` 耗时约在 50 分钟左右，运行完成后可以得到如下结果：

```
.
├── in
│   ├── genes -> /gpfs1/share/class4/data/chrX_data/genes
│   ├── genome -> /gpfs1/share/class4/data/chrX_data/genome
│   └── samples -> /gpfs1/share/class4/data/chrX_data/samples
├── job.srp*
├── ┊
├── out
│   ├── genome
│   │   ├── chrLength.txt
│   │   ├── chrNameLength.txt
│   │   ├── chrName.txt
│   │   ├── chrStart.txt
│   │   ├── exonGeTrInfo.tab
│   │   ├── exonInfo.tab
│   │   ├── geneInfo.tab
│   │   ├── Genome
│   │   ├── genomeParameters.txt
│   │   ├── Log.out
│   │   ├── SA
│   │   ├── SAindex
│   │   ├── sjdbInfo.txt
│   │   ├── sjdbList.fromGTF.out.tab
│   │   ├── sjdbList.out.tab
│   │   ├── star_index.log
│   │   └── transcriptInfo.tab
│   └── samples
│       ├── ERR188044
│       │   ├── Aligned.sortedByCoord.out.bam
│       │   ├── Aligned.sortedByCoord.out.bam.bai
│       │   ├── filtered.bam
│       │   ├── htseq.log
│       │   ├── Log.final.out
│       │   ├── Log.out
│       │   ├── Log.progress.out
│       │   ├── samtools.log
│       │   ├── SJ.out.tab
│       │   ├── _STARgenome
│       │   │   ├── sjdbInfo.txt
│       │   │   └── sjdbList.out.tab
│       │   ├── star_mapping.log
│       │   └── _STARpass1
│       │       ├── Log.final.out
│       │       └── SJ.out.tab
│       └── ┊
├── STA*_*.err
├── STA*_*.out
└── ┊
```

其中：

1. `job.srp*` 文件非空，包含了挂载到计算节点上任务的信息
2. `out/genome/` 文件夹中包含了基因组索引与构建索引时的日志信息
3. `out/samples/` 文件夹中包含了多个以样本 ID 命名的文件夹，包含了对应各个样本的比对结果、索引与过滤后的比对结果、转录本数据和日志信息
4. `STA*_*.{err,out}` 文件为挂载到计算节点上任务的控制台错误和输出重定向，由于脚本中指定了重定向路径故此处这些文件均为空（除非人为终止任务）

结论

此实验中我们使用 `STAR`、`samtools` 以及 `htseq` 工具对 `fastq` 格式双端测序结果比对到基因组上并得到过滤后基因的转录本信息，并且保留了依照基因过滤后的转录本比对结果，可供后续差异表达分析等操作使用。

参考资料

[1] MANZONI C, KIA D A, VANDROVCOVA J, 等. Genome, transcriptome and proteome: the rise of omics data and their integration in biomedical sciences[J]. Brief Bioinform, 2018, 19(2): 286–302. DOI:10.1093/bib/bbw114.

[2] YANG I S, KIM S. Analysis of Whole Transcriptome Sequencing Data: Workflow and Software[J]. Genomics Inform, 2015, 13(4): 119–25. DOI:10.5808/gi.2015.13.4.119.

[3] DOBIN A, DAVIS C A, SCHLESINGER F, 等. STAR: ultrafast universal RNA-seq aligner[J]. Bioinformatics, 2013, 29(1): 15–21. DOI:10.1093/bioinformatics/bts635.

[4] ANDERS S, PYL P T, HUBER W. HTSeq—a Python framework to work with high-throughput sequencing data[J/OL]. Bioinformatics, 2014, 31(2): 166–169. https://doi.org/10.1093/bioinformatics/btu638. DOI:10.1093/bioinformatics/btu638.

[5] DANECEK P, BONFIELD J K, LIDDLE J, 等. Twelve years of SAMtools and BCFtools[J]. GigaScience, 2021, 10(2). DOI:10.1093/gigascience/giab008.