mRNA-seq 分析

mRNA-seq Analysis

赵婧，刘南*

中国科学院生物与化学交叉研究中心，上海

*通讯作者邮箱: liunan@sioc.ac.cn

引用格式：赵婧，刘南．(2019)．mRNA-seq 分析．*Bio-101* e1010251．Doi: 10.21769/BioProtoc.1010251．

How to cite: Zhao, J and Liu, N. (2019). mRNA-seq analysis. *Bio-101* e1010251. Doi: 10.21769/BioProtoc.1010251. (in Chinese)

摘要：mRNA-seq 是通过逆转录过程将细胞产生的 RNA 转化为 DNA (cDNA，互补，并对获得的 cDNA 进行文库构建)。然后对得到的 DNA 进行测序，并从观察到的特定 DNA 丰度中，从中推断细胞中 mRNA 的原始量。大多数 mRNA-seq 分析的目标是找到在实验条件下转录水平发生变化的基因或转录本，即差异表达。通过找到这些基因和转录本，我们可以推断出不同条件的功能特征。mRNA-seq 通常建议的每种状态最小生物学重复次数为 3 次，5 次更好。如果预期到结果可能差异微妙或者生物变异显著时（例如对活体动物进行实验），需要更多的重复。所以，一组实验（对照组和实验组）意味着会有 6~10 个样品，产生 6~10 个或者更多 fastq 文件。所以能够明确以及流程化分析过程十分重要。

关键词：mRNA-seq analysis，序列比对，测序定量，差异表达

仪器设备

1. 服务器 (型号: I620-G20; 系统: centos7; CPU: Xeon(R) CPU E5-2620 v4 @2.10GHz, 3 核), 用于运行相关命令
2. 普通个人电脑 (需要已经安装好 R, 能调用个人终端。本教程是使用 Mac 系统进行操作的。)

软件

1. FastQC (v0.11.8)

2. mulitiQC (v1.6)
3. flexbar (v 3.3.0)
4. STAR (STAR_2.6.0a)
5. samtools (v1.9)
6. HTSeq (0.11.0)
7. R (v3.5.1)

实验步骤

注：本教程是基于已经在服务器上和个人电脑上安装好相关软件进行的。如果安装出现问题，请参考以下链接：

FastQC: <https://www.bioinformatics.babraham.ac.uk/projects/download.html>

mulitiQC: <https://multiqc.info/>

flexbar: <https://github.com/seqan/flexbar>

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

Samtools: <http://www.htslib.org/download/>

HTSeq: https://htseq.readthedocs.io/en/release_0.10.0/

1. 数据获得

本教程以我们实验室 2018 的 eLIFE 文章 (Epigenetic drift of H3K27me3 in aging links glycolysis to healthy longevity in *Drosophila*) 中的部分数据作为测试数据(Ma 等, 2018)。

注：一般已发表文章所包含的数据可以在 NCBI (SRA、GEO 等)、EMBL-EBI 等相关数据库获得。本文中的原始数据 (raw data) 可以在 eLIFE 文章 (Ma 等, 2018) 中获取。

2. 通过 FastQC 对数据进行质量评估

\$ fastqc 所在目录/fastqc -t 4 -o 输出文件夹 输入文件

在文件夹可以看到每个输入文件有两个输出结果，一个结果集合压缩包和一个网页型结果。一般通过查看网页结果即可知道基本的测序质量。但是具有多个样品逐一打开过于麻烦，所以可以通过 multiQC 将所有的结果综合到一起查看，输入命令：

\$ multiqc 目标文件夹

输出结果是网页形式，结果如图 1 所示：

图 1. MultiQC 输出结果

通过结果得知，测序样品结果稳定，测序质量很高。除了用 multiQC 查看多个 QC 结果以外，我们还用名为 [fastqc_summary.py](#) 的一个 python 脚本看每个样本的 reads 数量，GC 含量，Q20，Q30 的比例。脚本如图 2 所示：

```
import re
import zipfile

# read the zip file
def zipReader(file):
    #打开zip文件，读取xx_data.txt的数据
    qcfile = zipfile.ZipFile(file)
    #读取每一行的数据，用正则表达式模块re，找到目标行
    data_txt = [file for file in qcfile.namelist() if re.match(".*?_data\\.txt", file)][0]
    #根据分隔符对每一行进行分割，进行赋值。
    data = [bytes.decode(line) for line in qcfile.open(data_txt)]
    return data

#calculate
def fastqc_summary(data):
    module_num = 0
    bases = 0
    Q20 = 0
    Q30 = 0
    for line in data:
        if re.match('Filename', line):
            filename = line.split(sep="\t")[1].strip()
        if re.match('Total Sequence', line):
            read = line.split(sep="\t")[1].strip()
        if re.match('%GC', line):
            GC = line.split(sep="\t")[1].strip()
        if re.match("[^#](.*?\t){6}", line):
            bases = bases + 1
            if float(line.split("\t")[1]) > 30:
                Q20 = Q20 + 1
                Q30 = Q30 + 1
            elif float(line.split("\t")[1]) > 20:
                Q20 = Q20 + 1
        #每个碱基只读一次
        if re.match(">>END", line):
            module_num = module_num + 1
            if module_num >= 2:
                break
    Q20 = Q20 / bases
    Q30 = Q30 / bases
    summary = [filename, read, GC, str(Q20), str(Q30)]
    return summary

if __name__ == '__main__':
    import sys
    for arg in range(1, len(sys.argv)):
        data = zipReader(sys.argv[arg])
        summary = fastqc_summary(data)
        with open('summary.txt', 'a') as f:
            f.write('\t'.join(summary) + '\n')
```

图 2. fastqc_summary.py 脚本内容

这段脚本的逻辑为：

- 1) 用 python 的 zipfile 模块打开 zip 文件，读取 xx_data.txt 的数据。
- 2) 读取每一行的数据，用正则表达式模块 re，找到目标行。
- 3) 根据分隔符对每一行进行分割，进行赋值。
- 4) 由于只需要读取到>>Per base sequence quality pass 这一部分，所以设置一个>>END_MODULE 的计数器，数量超过 2，就停止。

运行命令为:

```
$ python3 ./fastqc_summary.py 目标文件
```

运行后会输出一个 txt 文本, 结果如图 3 所示:

```
muscle-10_S10_L001_R2_001.fastq 2568029 49 1.0 0.9736842105263158
muscle-10_S10_L002_R2_001.fastq 2516903 49 1.0 0.9736842105263158
muscle-10_S10_L003_R2_001.fastq 2635442 49 1.0 0.9736842105263158
muscle-10_S10_L004_R2_001.fastq 2554983 49 1.0 0.9736842105263158
muscle-9_S9_L001_R2_001.fastq 2649132 49 1.0 0.9736842105263158
muscle-9_S9_L002_R2_001.fastq 2589635 49 1.0 0.9736842105263158
muscle-9_S9_L003_R2_001.fastq 2710788 49 1.0 0.9736842105263158
muscle-9_S9_L004_R2_001.fastq 2624663 49 1.0 0.9736842105263158
```

图 3. 测序 Q30 结果

结果表明 Q20 指标在 1.0, Q30 指标在 0.97, 可以说测序质量非常好了!

3. 对测序文件进行比对

3.1 flexbar

在比对文件之前需要先要去掉 adapter, 得到不含有 adapter 的 fastq 文件。我们采用的是 flexbar 程序, 运行代码如下:

```
$ FLEXBAR -t 输出文件名 -qf i1.8 -r 输入文件 -a Adapters.fa
```

3.2 STAR

获得 trimmed 数据后, 就要与参考文件进行比对。目前大家所认同的比对算法有很多, 比如 Bowtie2, BWA, TopHat, HISAT2 和 STAR 等等。我们采用的是综合指标 (Sahraeian 等, 2017) 比较高的 STAR 进行比对。

注: 由于 STAR 的运算需要较高的配置, 当使用个人电脑进行学习分析流程的时候建议使用 TopHat 进行比对, 可参考 <https://www.jianshu.com/p/0abf87b3aca0> 进行操作。如果是基因组较小的物种 (果蝇、线虫等) 也可以使用个人电脑运行 STAR (测试过的笔记本电脑配置:

MacBook Pro, 2.3GHz Intel Core i5 处理器, 8GB 内存)

使用 STAR，首先需要对参考基因组建立索引，代码如下：

```
$ STAR --runThreadN 线程数 --runMode genomeGenerate
--genomeDir 存放参考基因的目录 --genomeFastaFiles 参考基因组的
fastq 文件 --sjdbGTFfile 参考基因组的 gtf 文件 --sjdbOverhang
索引长度
```

建立好索引就可以对数据进行比对了，比对命令如下：

```
$ STAR --genomeDir 存放参考基因的目录 --readFilesIn 输入文件
--runThreadN 线程数 --outSAMtype SAM --outFileNamePrefix 输出文件前缀
```

STAR 运行结束后会生成一个 `log.final.out` 文件，打开该文件可以看到比对结果如何。图 4 是本篇 protocol 中用到的示例数据的比对结果，特异比对率在 92.9%，说明比对结果很好，样品没有污染，可信度高。

注：一般情况下，*uniquely mapped reads %* 在 90% 以上就认为样品良好，没有污染。如果比对比例低于 70%，需要慎重对待。可以采取的办法是，只保留比对上的数据，如果数据量满足需求（比如重测序深度达 10X，则可以继续使用）；如果不满足要求，则需要重新准备样本。而对于很多医学样品可能是来源于感染的组织/细胞，那么它们的比对到其中一个基因组的效率低是正常的。

Number of input reads		2550987
Average input read length		146
UNIQUE READS:		
Uniquely mapped reads number		2371639
Uniquely mapped reads %		92.97%
Average mapped length		145.78
Number of splices: Total		562511
Number of splices: Annotated (sjdb)		554112
Number of splices: GT/AG		556185
Number of splices: GC/AG		3886
Number of splices: AT/AC		134
Number of splices: Non-canonical		2306
Mismatch rate per base, %		0.56%
Deletion rate per base		0.03%
Deletion average length		2.43
Insertion rate per base		0.02%
Insertion average length		2.40
MULTI-MAPPING READS:		
Number of reads mapped to multiple loci		64259
% of reads mapped to multiple loci		2.52%
Number of reads mapped to too many loci		21903
% of reads mapped to too many loci		0.86%
UNMAPED READS:		
% of reads unmapped: too many mismatches		0.00%
% of reads unmapped: too short		3.59%
% of reads unmapped: other		0.07%
CHIMERIC READS:		
Number of chimeric reads		16189
% of chimeric reads		0.63%

图 4. STAR 比对结果

3.3 Samtools

注: *Samtools* 的使用主要是为了生成可以在 *igv* 中可视化的 *bw* 文件, 后续 *htseq-count* 可以不使用此部分结果。

在 *STAR* 中, 它是可以直接将 *sam* 文件转换成 *bam* 文件并进行排序的。为了后续的分析的方便, 我们是采取 *samtools* 进行这一操作。

我们首先使用 *samtools* 将 *STAR* 比对后生成的 *sam* 文件转换成 *bam* 文件, 命令如下:

```
$ SAMTOOL view -b -S 输入 sam 文件 -o 输出 bam 文件
```

注: *Samtools* 的新版本已经可以直接不用转换的 *sam* 文件进行 *Samtools sort* 的排序操作。

然后对获得的 *bam* 文件进行排序, 命令如下:

```
$ SAMTOOL sort -o 输出排序后文件 输入 bam 文件
```

最后, 我们将已经排序的 *bam* 文件生成 *index*, 命令如下:

```
$ SAMTOOL index 输出文件
```

这时获得的 bam 文件可以直接在 igv 中进行可视化等其他操作。

3.4 HTseq

在 RNA-seq 中，我们最关心的是基因的表达量，所以我们通过 HTseq 程序获得每个基因的 reads 数，通过 reads 数可以进行后面的差异表达分析。当然根据问题需要，还可以通过 cufflinks 等程序选择计算 TPM, RPKM 等进行下游分析。使用 HTseq 的命令是：

```
$ htseq-count 输入比对后的 sam 文件 GTF 参考文件 -m
intersection-strict -s no > 输出 counts 文件
```

注：这里输入的 STAR 比对输出的 sam 文件，对于双端测序是必须输入排序后的 sam/bam 文件 (htseq-count 中含有排序命令，详情请在终端输入 htseq-count -h 查阅具体使用方法)，对于单端测序排序与否可以忽略。

4. 差异表达分析

HTseq 的输出文件可以直接与 R 包 [DESeq2](#) 对接，得到差异表达分析数据。将 R 的工作环境设置到含有 HTseq 输出的 counts 文件下，运行如下脚本 (图 5)：


```
library("DESeq2")

#Input counts file
sampleTable=data.frame(
  sampleName = c("3d_c1","3d_c2","3d_c3","3d_c4",
    "3d_e1","3d_e2","3d_e3","3d_e4"),
  fileName = c("C1.counts","C2.counts","C3.counts","C4.counts",
    "E1.counts","E2.counts","E3.counts","E4.counts"),
  condition = c("3d_control","3d_control","3d_control","3d_control",
    "3d_exprement","3d_exprement","3d_exprement","3d_exprement"))

#Build DESeqDataSet
ddsHTSeq <- DESeqDataSetFromHTSeqCount(sampleTable=sampleTable,
  directory=".",
  design=~condition)

#Pre-filtering
keep <- rowSums(counts(ddsHTSeq)) >= 10
ddsHTSeq1 <- ddsHTSeq[keep,]

#Differential expression analysis
dds <- DESeq(ddsHTSeq1)
res <- results(dds, contrast=c("condition","3d_exprement","3d_control"))

# Write the table out
write.csv(as.data.frame(res),file="3d_control_vs_3d_exprement.csv")

# Get normalized counts and write this to a file
nc = counts(dds,normalized=TRUE)

# Turn it into a dataframe to have proper column names.
dt = data.frame("id"=rownames(nc),nc)

# Save the normalize data matrix.
write.table(dt, file="norm-matrix-deseq2.txt", sep="\t", |row.name=FALSE, col.names=TRUE,quote=FALSE)

#PCA plot
rld<-rlog(dds)
plotPCA(rld)
```

图 5. DESeq2 差异表达分析 R 脚本

运行脚本后我们可以获得一个差异表达的表格，标准化了的 counts 表达矩阵，以及一张 PCA 图 (图 6)。

图 6. PCA 图

5. 下游分析

到目前为止，我们已经获得了实验组和对照的差异表达数据集。在这个数据集中包含了差异表达倍数、*P* 值、本底表达量等信息，后面可以基于这组数据进行下游分

析。比如对筛选出的差异基因做热图展示、GO 分析、KEGG 分析等。

参考文献

1. Ma, Z., Wang, H., Cai, Y., Wang, H., Niu, K., Wu, X., Ma, H., Yang, Y., Tong, W., Liu, F., Liu, Z., Zhang, Y., Liu, R., Zhu, Z. J. and Liu, N. (2018). [Epigenetic drift of H3K27me3 in aging links glycolysis to healthy longevity in *Drosophila*](#). *Elife* 7: e35368.
2. Sahraeian, S. M. E., Mohiyuddin, M., Sebra, R., Tilgner, H., Afshar, P. T., Au, K. F., Bani Asadi, N., Gerstein, M. B., Wong, W. H., Snyder, M. P., Schadt, E. and Lam, H. Y. K. (2017). [Gaining comprehensive biological insight into the transcriptome by performing a broad-spectrum RNA-seq analysis](#). *Nat Commun* 8(1): 59.