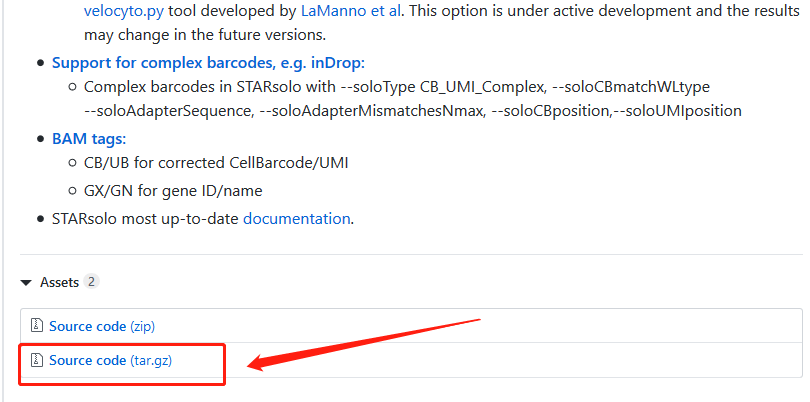
STAR比对、Salmon量化流程

一、软件下载：

STAR软件安装说明在https://github.com/alexdobin/STAR

下载地址：<https://github.com/alexdobin/STAR/releases>



下载之后直接解压就可以使用了。

tar zxvf 2.7.3a.tar.gz

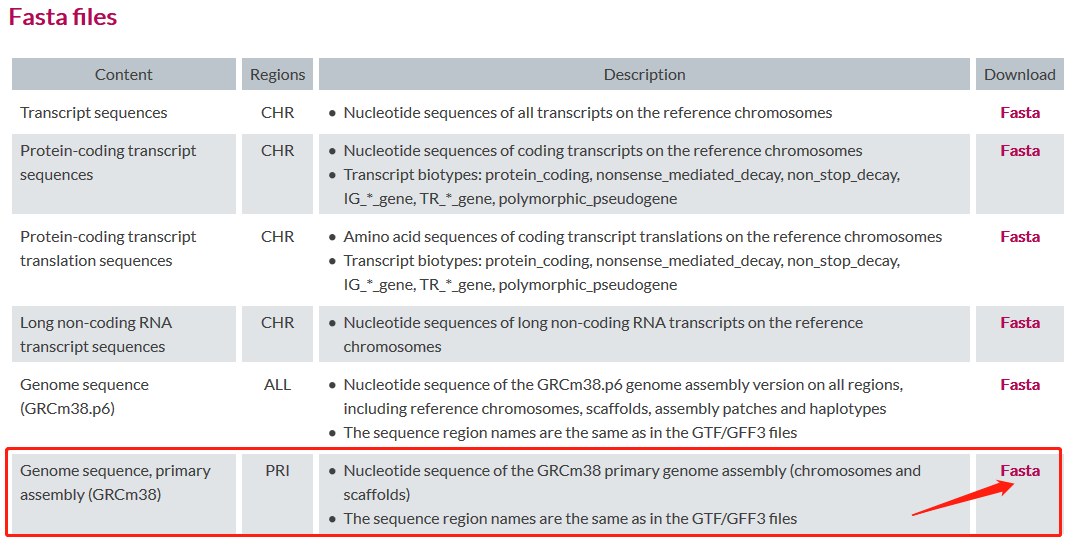
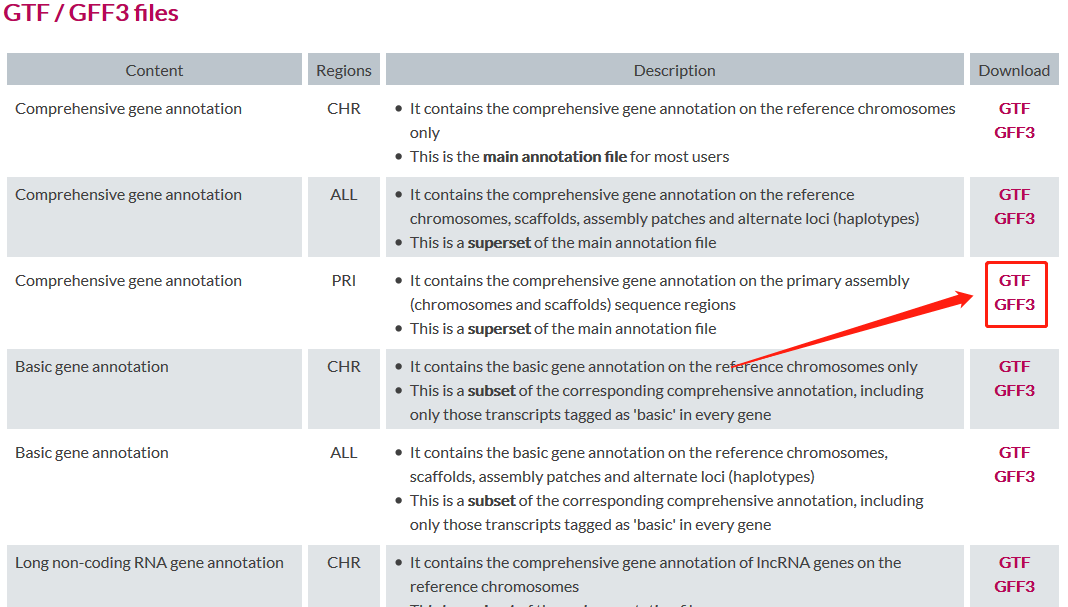
进入文件路径之后记得将bin目录下的执行文件写进环境变量bashrc，方便调用。

二、建立index：

1、文件下载：

STAR推荐使用注释较全面的基因组和注释文件，比如GENCODE的初级组装PRI (primary)数据。

<https://www.gencodegenes.org/mouse/release_M22.html>



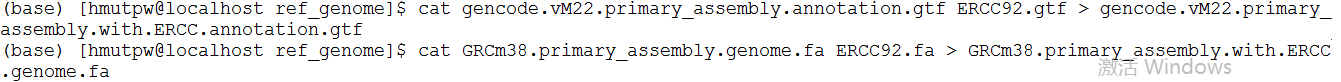
注意：1、如果是GFF3文件，需要使用—sjdbGTFtagExonParentTranscript参数。

2、下载的文件需要解压。

3、如果数据里面包含ERCC spike in，则可以将spike-in的序列和注释文件合并在一起。

2、建立index：

由于我的数据含有ERCC spike-in，所以先将spike-in合并到基因组序列中。



建立index：

ref\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/ref\_genome"

index\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/index/STARIndex/mouseGencodeVM22"

STAR --runMode genomeGenerate --runThreadN 16 --sjdbOverhang 100 --genomeDir $index\_dir --genomeFastaFiles $ref\_dir/GRCm38.primary\_assembly.with.ERCC.genome.fa --sjdbGTFfile $ref\_dir/gencode.vM22.primary\_assembly.with.ERCC.annotation.gtf

我使用16个线程大约花费23分钟建立了小鼠的index。

几个重要参数：

--runMod：运行模式，用于选择是建立index、比对或者是其他功能。

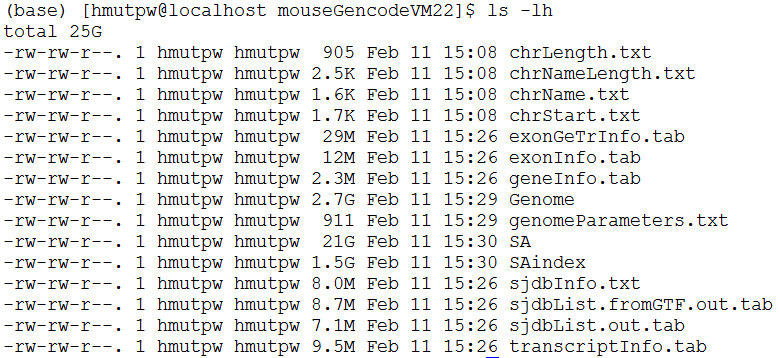
--runThreadN：线程数目，越多越快。

--sjdbOverhang：用于控制junction数据的长度，和测序长度有关，ReadLength-1，比如数据是illumina双端150bp测序，那么这个参数设置为149。如果测序数据中序列长短不一，那么我们可以设置为max(ReadLength)-1。

--genomeDir：基因组参考序列文件。

--sjdbGTFfile：基因的gtf注释文件。

建立完index之后的文件夹下面有这些文件：



注意：1、index文件存放位置至少准备100G的磁盘空间。

三、STAR比对：

#!/bin/bash

for sample in `cat /mnt/odisk/raw4T/HSC\_Autophagy/sample\_infor/mouse\_sample\_list.txt`

do

ref\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/ref\_genome"

index\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/index/STARIndex/mouseGencodeVM22"

fastq\_dir="/mnt/hdisk/main4T/tpw\_SC\_AS/clean\_data/$sample"

out\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/STAR\_res/alignment\_res/$sample"

mkdir -p $out\_dir;

echo '[perform read alignment with STAR] '$sample

STAR --runMode alignReads --runThreadN 16 --genomeDir $index\_dir --genomeLoad NoSharedMemory --readFilesCommand zcat --sjdbGTFfile $ref\_dir/gencode.vM22.primary\_assembly.with.ERCC.annotation.gtf --outFileNamePrefix $out\_dir/ --outSAMattributes All --outSAMtype BAM SortedByCoordinate --quantMode TranscriptomeSAM GeneCounts --twopassMode Basic --readFilesIn $fastq\_dir/$sample".R1.groomed.fq.gz" $fastq\_dir/$sample".R2.groomed.fq.gz"

done

几个重要参数：

--genomeDir：上一步建立好的index的路径。

--genomeLoad：STAR的index在内存中的管理，详细解释去看手册。

--readFilesCommand：STAR的输入文件是否需要解压，一般我们的fastq文件都是压缩为gz格式的，所以这里需要解压。

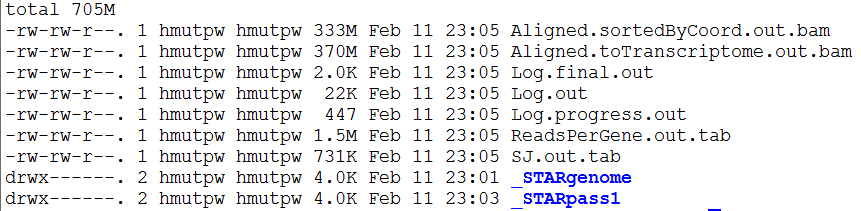
--outFileNamePrefix：这里是输出文件的路径，可以使文件夹，也可以是文件夹后面接输出文件的前缀名。

--outSAMattributes：输出的sam文件包含的tag信息。

--outSAMtype：输出的文件类型，为了节省硬盘空间，我们一般使用按照位置排序好的bam文件（这里STAR的排序可能和samtools的排序不一样，如果后续使，最好还是再用samtools对它再做一遍sort和index）。

--quantMode：是否需要把比对到转录本坐标上的bam输出出来以及是否需要计算基因的表达。

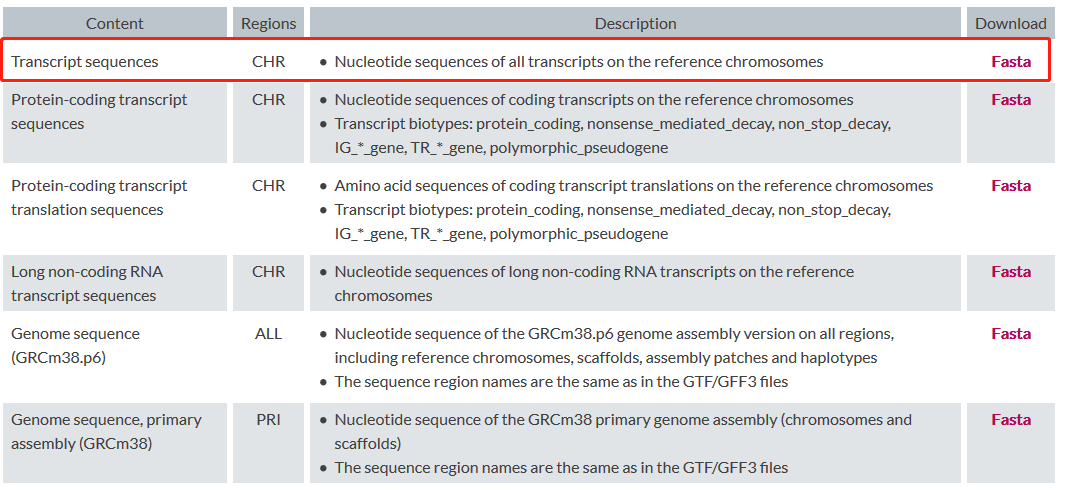
--twopassMode：是否需要进行两次比对，软件建议进行两次，具体原因见手册。



四、转录本表达量计算：

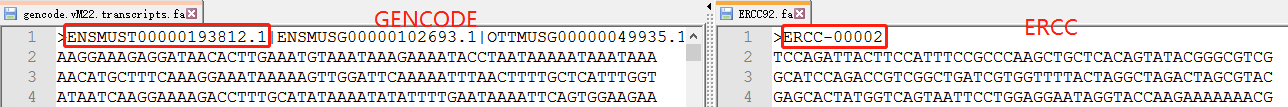
我们接下来利用salmon对上述比对结果进行转录本表达水平定量，因为salmon需要按照转录组比对结果进行计算，所以需要先下载转录本的序列。

1、下载gencode 转录本的fasta文件。



2、接下来将上述转录本序列与ERCC的序列合并成一个fasta文件备用

gencode.vM22.transcripts.fa文件里面header的第一个ID是转录本ID，而ERCC92.fa文件里面header的第一个ID是gene id，因此我们需要将ERCC92.fa的文件里header中的ID替换为转录本的，



#!/bin/bash

ref\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/ref\_genome"

#get ERCC transcript fasta.

perl ./03.1.2.get\_ERCC\_transcript\_fasta.pl $ref\_dir/ERCC92.gtf $ref\_dir/ERCC92.fa $ref\_dir/ERCC92.transcript.fa

#merge all wanted fasta files.

cat $ref\_dir/gencode.vM22.transcripts.fa $ref\_dir/ERCC92.transcript.fa > $ref\_dir/gencode.vM22.all.ERCC.transcripts.fa

此外，如果只想计算protein coding gene和lncRNA gene，只需要选择其中的pc和lncRNA的fasta文件合并就行。例如：

cat $ref\_dir/gencode.vM22.pc\_transcripts.fa $ref\_dir/gencode.vM22.lncRNA\_transcripts.fa $ref\_dir/ERCC92.transcript.fa > $ref\_dir/gencode.vM22.pc.lncRNA.ERCC.transcripts.fa

3、利用salmon计算转录本表达量：

因为salmon需要提供转录组比对得到的Aligned.toTranscriptome.out.bam文件和转录本序列gencode.vM22.all.ERCC.transcripts.fa文件，并且bam文件中的所有转录本**必须要在**fasta文件找到，因此我们要先对比对好的bam文件进行filter，去掉bam文件中在fasta文件里面没有的转录本。（注意：salmon会check sam/bam 文件的header的transcript，所以在filter的时候也需要filter sam/bam文件的header部分）

#!/bin/bash

for sample in `cat /mnt/odisk/raw4T/HSC\_Autophagy/sample\_infor/mouse\_sample\_list.txt`

do

ref\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/ref\_genome"

bam\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/STAR\_res/alignment\_res/$sample"

out\_dir="/mnt/odisk/raw4T/HSC\_Autophagy/STAR\_res/salmon\_res/$sample"

mkdir -p $out\_dir;

date;

echo "[Convert bam files to sam files]"

samtools view -@ 8 -h $bam\_dir/Aligned.toTranscriptome.out.bam > $out\_dir/Aligned.toTranscriptome.out.sam

date;

echo "[Filter sam files with transcripts included in fasta files]"

perl 03.2.2.filter\_bam\_for\_salmon.pl $ref\_dir/gencode.vM22.all.ERCC.transcripts.fa $out\_dir/Aligned.toTranscriptome.out.sam $out\_dir/Aligned.toTranscriptome.out.for.salmon.sam

echo '[Perform transcript quantification with Salmon] '$sample

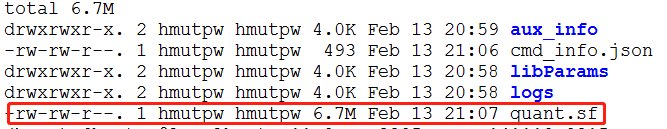
salmon quant -p 8 --libType IU --seqBias --gencode -t $ref\_dir/gencode.vM22.all.ERCC.transcripts.fa -a $out\_dir/Aligned.toTranscriptome.out.for.salmon.sam -o $out\_dir/

rm $out\_dir/Aligned.toTranscriptome.out.sam

rm $out\_dir/Aligned.toTranscriptome.out.for.salmon.sam

done

结果如下：



quant.sf文件就是我们得到的转录本表达数据，其中几列的解释：

Name：转录本ID

Length：转录本长度

EffectiveLength：有效地计算转录本丰度的长度

TPM：标准化的TPM值，这里因为有ERCC数据的存在，这个值应该不是很准确。下一步我们要去除ERCC的影响重新计算TPM值。

5、利用tximport将转录本map到基因上

5.1、首先我们要准备一个对应于自己版本gtf文件的一个转录本对应基因ID的csv文件。

perl 04.1.1.transcript\_to\_gene\_from\_gtf.pl ../ref\_genome/gencode.vM22.primary\_assembly.with.ERCC.annotation.gtf ../ref\_genome/gencode.vM22.transcript2gene.with.ERCC.csv

其中04.1.1.transcript\_to\_gene\_from\_gtf.pl文件是自己编写的文件。结果文件第一列是transcript ID，第二列是gene ID，列名可以随意，但是列的顺序不可以变。

5.2、将salmon中转录本的counts矩阵合并和计算基因的counts

脚本：05.1.1.merge\_transcript\_exp\_to\_gene\_from\_salmon.r

#!/usr/bin/env Rscript

#get salmon result path.

salmon\_dir = "/mnt/odisk/raw4T/HSC\_Autophagy/STAR\_res/salmon\_res"

samples = list.files(salmon\_dir)

files <- file.path(salmon\_dir, samples, "quant.sf")

names(files) <- samples

#get transcript to gene relation.

library(readr)

ref\_dir = "/mnt/odisk/raw4T/HSC\_Autophagy/ref\_genome"

tx2gene <- read\_csv(file.path(ref\_dir,"gencode.vM22.transcript2gene.with.ERCC.csv"))

head(tx2gene)

#get transcript/gene level counts.

library(tximport)

txi <- tximport(files, type = "salmon", txOut=TRUE)

transcript\_counts = txi$counts

transcript\_length = txi$length

txi.gene = summarizeToGene(txi, tx2gene = tx2gene)

gene\_counts = txi.gene$counts

gene\_length = txi.gene$length

#calculate transcript TPM.

enst = grep("^ENS",row.names(transcript\_counts),value=T)

spikes = setdiff(row.names(transcript\_counts), enst)

transcript\_enst\_counts = transcript\_counts[enst,]

transcript\_enst\_length = transcript\_length[enst,]

transcript\_spikes\_counts = transcript\_counts[spikes,]

library(data.table)

transcript\_TPM = data.frame(transcript\_id = enst)

for(i in samples){

count\_frm = data.frame(counts = transcript\_enst\_counts[,i],length = transcript\_enst\_length[,i])

count\_norm = apply(count\_frm,1,function(x){as.numeric(x[1])/as.numeric(x[2])/1000})

tpm\_frm = data.frame(transcript\_id = enst, TPM = (10^6\*count\_norm/sum(count\_norm)))

colnames(tpm\_frm)[2] = i

transcript\_TPM = data.frame(transcript\_TPM, tpm\_frm, by="transcript\_id", all=T)

}

out\_dir = "/mnt/odisk/raw4T/HSC\_Autophagy/STAR\_res/salmon\_merge\_counts"

dir.create(out\_dir)

write.table(transcript\_enst\_counts,file.path(out\_dir,"transcript\_raw\_counts.txt"),sep="\t",quote=F)

write.table(transcript\_spikes\_counts,file.path(out\_dir,"transcript\_ERCC\_counts.txt"),sep="\t",quote=F)

write.table(transcript\_TPM,file.path(out\_dir,"transcript\_normalized\_TPM.txt"),sep="\t",quote=F)

#calculate gene TPM.

ensg = grep("^ENS",row.names(gene\_counts), value=T)

spikes = setdiff(row.names(gene\_counts), ensg)

gene\_ensg\_counts = gene\_counts[ensg,]

gene\_ensg\_length = gene\_length[ensg,]

gene\_spikes\_counts = gene\_counts[spikes,]

gene\_TPM = data.frame(gene\_id = ensg)

for(i in samples){

count\_frm = data.frame(counts = gene\_ensg\_counts[,i],length = gene\_ensg\_length[,i])

count\_norm = apply(count\_frm,1,function(x){as.numeric(x[1])/as.numeric(x[2])/1000})

tpm\_frm = data.frame(gene\_id = enst, TPM = (10^6\*count\_norm/sum(count\_norm)))

colnames(tpm\_frm)[2] = i

gene\_TPM = data.frame(gene\_TPM, tpm\_frm, by="gene\_id", all=T)

}

write.table(gene\_ensg\_counts,file.path(out\_dir,"gene\_raw\_counts.txt"),sep="\t",quote=F)

write.table(gene\_TPM,file.path(out\_dir,"gene\_normalized\_TPM.txt"),sep="\t",quote=F)

write.table(gene\_spikes\_counts,file.path(out\_dir,"gene\_ERCC\_counts.txt"),sep="\t",quote=F)

注意：1、这一步计算时间较长。

2、gene和transcript的TPM值需要将ERCC spike in 去除之后按照EffectiveLength标准化。