bio-101 www.bio-protocol.org/e1010638 DOI:10.21769/BioProtoc.1010638

高质量基因组组装

High-Quality Genome Assembly

王忠凯，黎浩榕，李永鑫*

西北工业大学生态环境学院，西安710072

*通讯作者邮箱：yxli28science@sina.com

引用格式：王忠凯，黎浩榕，李永鑫.（2021）.高质量基因组组装. Bio-101 e1010638. Doi: 10.21769/BioProtoc.1010638.

How to cite: Wang, Z. K., Li, H. R.andLi, Y. X. (2021). High-Quality Genome Assembly. Bio-101 e1010638. Doi: 10.21769/BioProtoc.1010638. (in Chinese)

摘要：上世纪50年代开始，DNA双螺旋结构的提出（1953），中心法则的确定（1958），和遗传密码的破译（1966）等一系列理论的突破推动了生命科学领域研究的飞速发展。直至1975年，以Sanger测序为代表的第一代DNA测序体系的建立使得科研工作者可以深入解读生命体的遗传奥秘，生命科学步入了基因组学时代。在后续的几十年里，测序技术有了非常迅猛的发展，出现了以Roche公司的454技术、Illumina公司的Solexa和Hiseq技术、BGI公司的DNB技术以及ABI公司的Solid技术等为代表的二代测序技术，而以Pacific Biosciences (PacBio)和Oxford Nanopore Technologies (Nanopore)公司为代表的单分子测序技术则发展为第三代测序技术。在今天这个信息快速爆发的后基因组时代，解析得到每个物种的高质量基因组是几乎所有研究中必不可少的关键因素，更是很多后续分析的基础所在。目前主流的高质量基因组组装主要是依据混合组装策略，即三代测序数据搭建contig，然后利用二代短片段数据进行纠错，继而通过BioNano和Hi-C等技术将其组装到scaffold或染色体水平。本文将对相关的主流软件和方法做一个系统性的介绍。

关键词：基因组，测序，组装，读长

一、三代测序数据的基因组组装

尽管三代单分子测序 (PacBio和Nanopore相关平台) 具有的长读长 (read) 优势为我们获取高连续性的基因组提供了很大的帮助，但是其高昂的价格和较高的错误率等缺点对其组装应用有所限制。如今PacBio常用的测序模式有两种，分别是Circular Consensus Sequencing (CCS，也称为HiFi)和Continuous Long Read (CLR) Sequencing。CCS通过滚环测序的方式，将单模板测序错误的随机性与多轮测序相结合获得一致性序列，大大提高了测序单碱基准确率。CLR测序模式和Nanopore测序类似，产出错误率较高但是读长较长的数据，无GC偏好性且可以同时记录碱基修饰信息。相比较而言，PacBio的CCS测序模式价格相对昂贵，但是reasds单碱基准确性较高 (99.9%)；而其CLR和Nanopore测序价格则相对便宜且测序读长较长，但是单碱基准确性不高 (~85%)。不同平台各有优缺点，研究人员可以根据自身需求选择不同测序平台和测序模式。基于PacBio单分子测序的基因组组装策略有以下几类：1) 三代测序数据深度较低 (<5x) 时，通常以二代测序数据基因组组装的contig之间的骨架为草图，低深度三代测序数据辅助补洞 (gap filling) 和提升基因组；2) 三代测序深度居中 (5-50x) 时，可以进行二代三代混合组装，以三代组装结果为草图，二代reads辅助纠错；3) 三代测序深度较高 (>50x) 时，可以进行纯三代reads组装。基于Nanopore测序的组装策略，则通常是高深度三代测序，然后结合二代数据进行单碱基校正的二三代结合方式。研究人员可以根据不同需求选择不同的策略，本文主要介绍相关的混合组装策略。

主流的三代基因组组装软件有 Canu、FALCON、NextDenovo 和 wtdbg2 等。各种软件层出不穷，根据不同的需求可以选择不同的软件。本文介绍的几个软件已经基本可以满足大部分基因组的组装需求。

1. Canu

Canu(Koren *et al.*, 2017)的组装核心来自于 Celera Assembler(Myers *et al.*, 2000)，该软件最早应用于果蝇和人类基因组组装，是一种基于 Overlap-Layout-Consensus (OLC)的从头组装软件。Canu 组装结果质量较高，但是其对计算资源和服务器配置有较高的要求。Canu 分为三个步骤：数据纠错 (read correction)，数据修剪 (read trimming) 和 contig 构建 (contig construction)。其中每一步都采用以下模式 (Koren *et al.*, 2017)：

•将 reads 载入 read 数据库 gkpStore；

•计算 k-mer count，为计算 overlap 做准备；

•计算 overlap；

•将 overlap 载入 overlap 数据库 ovlStore；

•针对不同的任务，进行不同的运算：

a. Read correction 步骤将用从 overlap 中计算出的共有序列替换 read 中原始的噪音序列。

b. Read trimming 步骤将使用 overlap 来确定每条 read 高质量序列区域以及低质量序列区域。trimming 后，最长的高质量序列片段将会被保留下来。

c. Contig construction 步骤将计算 overlap 之间的相互关系，生成 read layout，并最终生成 consensus contig。

Canu 可以单做数据纠错或者单做组装 (reads 纠错用其他软件)，也可以一步完成 *de novo* 组装所需的所有步骤。其运行方式既可以通过命令行参数控制，也可以通过指定配置文件 (-s) 来控制。

一步运行示例如下：

```
    canu-d out_dir\ #使用-d 指定输出路径，默认是在当前运行路径
    -p out_prefix\ #输出文件前缀
    genomeSize=1.5g \ #支持后缀 g/m/k，基因组大小可通过软件获得，如 genomescope
    useGrid=1 gridEngine=pbs gridEngineResourceOption="-l nodes=1:ppn=THREADS:mem=MEMORY" gridOptions="-q high " \#grid 提交选项
    rawErrorRate=fraction-error \ #未纠错 reads 之间的 overlap 允许的差异，若 reads 质量较低 (如由于物种高复杂度引起的测序质量低)，则可以使用较大的数值。PB 默认值为 0.300，Nanopore 默认值为 0.500
    correctedErrorRate=fraction-error\ #纠错 reads 之间的 overlap 允许的差异，若基因组覆盖度较低或生物学差异较大，则可以调整该差异值。PB 默认值为 0.045，Nanopore 默认值为 0.144
    -pacbio|-nanopore|-pacbio-hifi reads/*.fasta.gz#reads 类型及路径，可以是三者之一
```

也可限制运行某些特定步骤：

```
        -haplotype     # generate haplotype-specific reads
        -correct       # generate corrected reads
        -trim          # generate trimmed reads
        -assemble      # generate an assembly
        -trim-assemble # generate trimmed reads and then assem
ble them
```

其余参数及配置可参考官方说明：https://canu.readthedocs.io/en/latest/parameter-reference.html

2. FALCON

FALCON 是 PacBio 公司开发的一款用于三代测序数据的 *de novo* 组装软件，与之一起开发的是其下游分析软件 FALCON-Unzip(Chin *et al*., 2016)。与该公司开发的 HGAP(Chin *et al*., 2013)软件比较而言，FALCON 更适合于大基因组的组装，且能分析双倍体序列，并在基因组组装结果中将变异位点信息和基因组序列分别以 alternative contigs(a-contigs)和 primary contigs(p-contigs)的形式给出。每一条 a-contig 都有其对应的 p-contig 序列。FALCON-Unzip 是单倍型组装软件，它能在 FALCON 软件的基因组组装结果基础上，根据识别出的 SNPs 等信息对基因组进行单倍型分型。

FALCON 流程主要包括三个步骤：

•pre-assembly：对 raw subreads 进行纠错并预组装，结果存放在 0-rawreads 路径下；

•pread overlapping：对上一步得到的 preads 进行相互比对，构建 overlap，结果存放在 1-preads_ovl 路径下；

•contig assembly：使用 overlap 结果进行构图并搭建 contig，结果存放在 2-asm-falcon 路径下。

FALCON 运行命令如下：

```
    /PATH/To/Installation/fc_run.pyfc_run.cfg
```

其中，fc_run.cfg 是其配置文件，以下是一个简单的配置示例：

```
    [General]
    job_type =sge
```

```
    input_fofn = input.fofn
    input_type = raw

    #Data Partitioning
    pa_DBsplit_option = -x500 -s400
    ovlp_DBsplit_option = -x500 -s400

    ####Pre-assembly
    length_cutoff = -1
    genome_size = 2800000000
    seed_coverage = 40
    pa_HPCdaligner_option=-v -B128 -M24
    pa_daligner_option= -k18 -e0.80 -l1000 -h256 -w8 -s100
    falcon_sense_option=--output-multi --min-idt 0.70 --min-c
ov 4 --max-n-read 200
    falcon_sense_greedy=False

    #Pread overlapping
    ovlp_HPCdaligner_option=-v -B128 -M24
    ovlp_daligner_option= -k24 -e.92 -l1800 -h1024 -s100

    #Final Assembly
    length_cutoff_pr=1000
    overlap_filtering_setting=--max-diff 100 --max-cov 100 --
min-cov 2
```

部分参数说明如下：

•input_type：表示输入的数据类型，raw指原始的subreads；preads指经过了pre-assembly的reads，也就是纠错之后的reads，若使用该类型数据，运行过程会跳过pre-assembly步骤。FALCON可以使用自己纠错的preads，也可以使用其它软件纠错后的reads。

•length_cutoff = -1：设置对种子序列的长度筛选阈值。若该参数值设置为-1，则程序自动计算种子序列的长度筛选阈值，根据 genome_size 和 seed_coverage

两个参数挑选最长的 reads 序列作为种子序列，直到其数据量达到基因组指定覆盖度为止。

•seed_coverage = 40：设置seed覆盖度。推荐设置seed_coverage参数的值为20~40×。

•pa_HPCdaligner_option和pa_daligner_option：该选项设置比对过程 (daligner) 的参数，其中pa_HPCdaligner_option主要影响比对过程所用资源，详细比对原理及参数可参考：https://dazzlerblog.wordpress.com/command-guides/daligner-command-reference-guide/；针对不同情况数据，作者建议参数为：

a. -e：平均序列相似度，低质量数据建议0.70，高质量数据建议0.80。
b. -l：overlap最短长度，文库质量若较差、片段较短建议设置为1000，文库质量较好、片段较长建议设置为5000。
c. -k：kmer大小，低质量数据建议设置为14，高质量数据建议设置为18；较小的kmer数值，比对灵敏度 (sensitivity) 会更高，但是所需计算资源会更大，运行会比较慢，适用于低质量数据；反之，较大的kmer值，特异性 (specificity) 会更高，使用资源较低，运行较快，但是只适用于高质量数据。

•pa_DBsplit_option和ovlp_DBsplit_option：该选项设置组装过程中对reads进行分块的参数，其中-s表示每份数据含有的数据量 (Mb)，默认值为200；-x表示忽略长度低于该阈值的reads；对于大基因组，作者推荐参数为pa_DBsplit_option=-x 500 -s200，ovlp_DBsplit_option=-x500 -s200；对于小基因组 (<10Mb)，作者推荐参数则为pa_DBsplit_option = -x500 -s50，ovlp_DBsplit_option = -x500 -s50；

•ovlp_daligner_option和ovlp_HPCdaligner_option：该选项设置pread overlapping运行过程中的参数，请注意，该参数作用方式与上述pa_HPCdaligner_option和pa_daligner_option类似，但是不可混淆，针对不同情况数据，作者建议参数为：

a. -e：平均序列相似度，近交 (inbred) 物种数据建议0.93，远缘杂交 (outbred) 物种数据建议0.96。
b. -l：overlap最短长度，preassembly结果较差或文库质量较差、片段较短建议设置为1800，文库质量较好、片段较长建议设置为6000。
c. -k：kmer大小，低质量数据建议设置为18，多数情况下建议设置为24。

•length_cutoff_pr：该参数设置用来最终组装的preads的最短长度，该值通常允

许有15-30x的纠错数据用来最终组装；

•overlap_filtering_setting：该参数设置preadoverlap的过滤标准，其中--max-diff表示5’和3’端覆盖度差异的阈值，高于该指定值的overlap将会被过滤掉；--max-cov通常会过滤掉由于污染或重复序列引入的overlap；--min-cov则会指定overlap的最低覆盖度。

其余参数及配置可参考官方说明：https://github.com/PacificBiosciences/pb-assembly

3. NextDenovo

NextDenovo(胡江，武汉希望组)是一款基于图论的三代测序数据(CLR, CCS和Nanopore)的基因组*denovo*组装软件。它使用类似于Canu的“先纠错后组装”的策略(PacBio CCS数据(Wenger *et al.*, 2019)不需要纠错步骤)，需要的计算资源和存储相对较少，组装快慢多数情况下取决于I/O瓶颈。其组装序列的单碱基准确度约为98%-99.8% (https://github.com/Nextomics/NextDenovo)。

但是请注意，当前release的版本只能进行小于3.5G的基因组组装，大基因组组装版本为非公开版本，需要联系软件发布者。

该软件用法比较简单，和多数软件一样，需要指定配置文件，组装运行快慢的关键在于资源配置。某些步骤出错时，可以直接杀掉所有任务，然后重新进行投递，这也是该软件的优势之一。

```
/PATH/To/Installation/nextDenovorun.cfg -l log.txt
```

示例配置如下，详细参数可参考：https://nextdenovo.readthedocs.io/en/latest/OPTION.html

```
[General]
job_type = pbs#指定任务运行方式，pbs，sge，或者 local，或者其他集群类型
job_prefix = NextDenovo
task = all # 'all', 'correct', 'assemble'
rewrite = yes # yes/no
deltmp = yes
rerun = 3
parallel_jobs = 100
```

```
input_type = raw
input_fofn = ./input.fofn
workdir= ./01_rundir
cluster_options = -q high2 -l nodes=1:ppn={cpu}
read_type = ont#clr, ont, ccs

[correct_option]
read_cutoff = 1k#过滤原始数据中低于 1k 的 read
blocksize = 5g
genome_size = 1.74g#estimated genome size
seed_depth = 40#纠错 seed 深度，和 FALCON 等类似
pa_correction = 100
seed_cutfiles = 50
sort_options = -m 60g -t 30 -k 50
minimap2_options_raw = -x ava-ont -t 10
correction_options = -p 25

[assemble_option]
random_round = 10
minimap2_options_cns = -x ava-ont -t 6 -k17 -w17
nextgraph_options = -a 1
```

4. wtdbg2

wtdbg2(Ruan and Li, 2020)整体上遵循 OLC 算法，结合了 all-vs-all 比对和 fuzzy -Bruijn graph (FBG)的优点，该软件解决了通常软件难以解决的高 I/O 瓶颈和大量无效 k-mer 的问题 (Ruan and Li, 2020)，以资源消耗少速度快的优点著称，尤其对于大基因组组装十分友好。然而，该软件也有如下限制：

•Reads 最长限制是 0x0003FFFFU (256 Kb)，再长的 reads 会被切分开；

•Reads 数量最多是 0x03FFFFFFU (64 M)，如果数量过多，使用者需要过滤部分较短的 reads；

•可以设定的线程数 (threads) 最多是 4096；

•不可以跨节点并行运行，但是可以通过 kbm 和 wtdbg --load-alignments 在特

定步骤实现跨节点并行计算。

wtdbg2 运行简单，可以有一步运行也可以手动分步运行，用法如下：

```
#quick start with wtdbg2.pl
./wtdbg2.pl -t 16 -x rs -g 4.6m -o dbg reads.fa.gz

# Step by step commandlines
# assemble long reads
./wtdbg2 -x rs -g 4.6m -i reads.fa.gz -t 16 -fodbg

# derive consensus
./wtpoa-cns -t 16 -i dbg.ctg.lay.gz -fodbg.raw.fa

# polish consensus, not necessary if you want to polish th
e assemblies using other tools
minimap2 -t16 -ax map-pb -r2k dbg.raw.fa reads.fa.gz | sam
tools sort -@4 >dbg.bam
samtools view -F0x900 dbg.bam| ./wtpoa-cns -t 16 -d dbg.ra
w.fa -i - -fodbg.cns.fa

# Addtionalpolishment using short reads
bwa index dbg.cns.fa
bwa mem -t 16 dbg.cns.fa sr.1.fa sr.2.fa | samtools sort -
O SAM | ./wtpoa-cns -t 16 -x sam-sr -d dbg.cns.fa -i - -fodbg.
srp.fa
```

详细用法及参数可参考：https://github.com/ruanjue/wtdbg2/blob/master/README-ori.md

二、二代短片段reads纠错

三代测序具有长读长、错误率高的特点。虽然组装软件会进行纠错，但是其组装结果的单碱基准确率仍然不及二代测序数据的组装结果。因此，通常会利用二代短片段文库测序 reads 对三代基因组组装结果进行纠错。常用的软件有 Pilon 和 NextPolish 等。

1. Pilon

Pilon(Walker *et al.*, 2014)可以对三代基因组组装结果进行纠错，要求输入基因组的 FASTA 文件以及二代测序数据比对到组装结果的 BAM 文件，其根据比对结果对输入的基因组质量进行提高。Pilon 可以使用三种不同二代数据的 BAM 文件：

```
    --frags <frags.bam> #paired-end 测序比对结果，例如 Illumina p
aired-end reads(insert size<1000bp).
    --jumps <jumps.bam> #mate-pair 测序比对结果，例如 Illumina ma
te-pair reads (insert size>1000bp).
    --unpaired <unpaired.bam> #unpaired sequencing reads.
```

Plion 整体用法比较简单，主要是需要根据不同情况准备输入的 BAM 文件：

```
    bwa index -p index/draft draft.fa
    bwa mem -t 20 index/draft read_1.fq.gz read_2.fq.gz | samt
ools sort -@ 10 -O bam -o align.bam
    samtools index -@ 10 align.bam

    java -Xmx${MEMORY}G -jar pilon-1.22.jar --genome draft.fa
 --frags align_filer.bam --fix snps,indels --output pilon_pol
ished --vcf&> pilon.log
```

其中，内存可以根据基因组大小来调整，1Mb 基因组需要分配至少 1GB 资源。实际使用来说，小基因组可以直接运行；但是对于大基因组，直接运行不太现实，可以采用如下切分的思想。首先，在比对阶段，将基因组按照 contig 进行切分，并基于所有 reads 比对到基因组上的 SAM 文件分别提取每条 contig 的比对信息，再进一步将提取结果转化成 SAM 和 BAM 文件。然后针对每一条提取的 contig 和 BAM 文件，分别运行 Pilon 进行纠错。最后将纠错结果合并。另外，Pilon 运行一般需要迭代至少 2-3 轮反复纠错，但是不可迭代次数太多。

详细参数及用法请参考：https://github.com/broadinstitute/pilon/wiki/Requirements-&-Usage

2. NextPolish

与 NextDenovo 一样，NextPolish(Hu*et al.*,2020)也是武汉希望组开发的一款三代基因组纠错工具。与 Pilon 相比，NextPolish 是基于 k-mer 运算的软件，所以运行速度和资源消耗都优于 Pilon；Plion 只能运用二代短读长序列进行纠错，而 NextPolish 可以使用二代短读长序列、三代长读长序列、或者两者结合来纠正组装时导致

的碱基错误，加之运行方法简单，因此目前 NextPolish 的实用性和适用面更广。

NextPolish 的运行方法如下：

```
/PATH/To/Installation/nextPolishrun.cfg
```

其中，run.cfg 为配置文件，示例配置如下：

```
[General]
job_type = local
job_prefix = nextPolish
task = best
rewrite = yes
rerun = 3
parallel_jobs = 6
multithread_jobs = 5
genome = ./raw.genome.fasta
genome_size = auto
workdir= ./01_rundir
polish_options = -p {multithread_jobs}

[sgs_option] #optional
sgs_fofn = ./sgs.fofn
sgs_options = -max_depth 100 -bwa

[lgs_option] #optional
lgs_fofn = ./lgs.fofn
lgs_options = -min_read_len 1k -max_depth 100
lgs_minimap2_options = -x map-ont
```

部分参数说明如下：

job_type: 任务投递方式，可选 local，sge，pbs，slurm，若使用非 local 参数运行，则需要安装对应系统的 DRMAA 来完成任务投递。

task = best：纠错任务模式，可选 all, default, best, 1, 2, 5, 12, 1212 等，其中，1 和 2 是针对短 reads 的算法模型；5 是针对长 reads 的算法模型。all=[5]1234, default=[5]12, best=[55]1212. (默认参数: best)。

sgs_options: 该选项设置二代测序 polish 的参数。

lgs_option：该选项设置三代测序 polish 的参数。

其余参数及用法请参考：https://nextpolish.readthedocs.io/en/latest/OPTION.html

三、Bionano 光学图谱组装

Bionano Saphyr 光学图谱作为基因组学研究的重要工具，其关键技术 DLS 标记 (Direct Label and Stain) 能够在不产生任何物理损伤的情况下，对基因组 DNA 进行荧光标记，进而通过结合芯片 Saphyr Chip 处理和高分辨率荧光成像系统，获得完整的 DNA 物理图谱，分子 N50 长度平均可达 300kb(grandomics.com)。超长的基因组物理图谱为基因组组装提供了染色体尺度的框架，并能高效检测到大片段纯合子和杂合子的结构变异。光学图谱测序数据的主要分析工具包含 Linux 下的运行软件 Solve (https://bionanogenomics.com/support/software-downloads/) 以及 Windows 下的可视化软件 Access (https://bionanogenomics.com/support/download-form?file=http://bnxinstall.com/access/BionanoAccessWindowsInstall.html&title=Bionano%20Access%20for%20Windows)，以下为详细运行步骤：

第一步：过滤短于给定阈值的分子。

```
   $Bionano/RefAligner/11442.11643rel/RefAlinger -iall.bnx -minlen 120 -merge -o output -bnx
```

第二步：基因组序列模拟酶切，得到 CMAP 文件。

```
   perl $Bionano/HybridScaffold/11162020/scripts/fa2cmap_multi_color.pl -i genome.fa -e cttaag 1
```

第三步：用 align_bnx_to_cmap.py 进行比对，Bionano 光学图谱比对的基本原理是基于标记的相对位置。需要根据输出结果 contigs/alignmolvref/alignmol_log_simple.txt 里的“Fraction of aligned molecules”和“Effective coverage of reference (×)”，判断数据是否符合最低的比对率。比对结果可以下载到本地，然后上传至 Access 进行可视化。

```
   python $Bionano/Pipeline/11162020/align_bnx_to_cmap.py  \
       --prefix  human \
       --mol input.bnx  \
       --ref reference.cmap \
```

```
        --ra $Bionano/RefAligner/11442.11643rel \
        --nthreads80  \
        --output prealign \
        --snrFilter 2 \
        --color 1
```

第四步：从头组装 Bionano 图谱，其中-a 参数最为重要 (其指定的 xml 文件配置了流程中每一步的具体控制参数)。

```
    python $Bionano/Pipeline/11162020/pipelineCL.py \
        -T 96 -N 4 -f 1 \
        -i 5 \
        -b input.bnx \
        -y -r reference.cmap \
        -l Assembly \
        -t $Bionano/RefAligner/11442.11643rel/RefAlinger \
        -a  $Bionano/RefAligner/11442.11643rel/RefAlinger/optArgu
ments_nonhaplotype_saphyr_human.xml
```

第五步：从头组装结果图谱与参考基因组图谱比对，生成三个比对文件 (.xmap, _q.cmap，_r.cmap)。这些比对文件可上传至 Access 进行可视化。

```
    python $Bionano/Pipeline/11162020/runCharacterize.py \
        -t $Bionano/RefAligner/11442.11643rel/RefAlinger \
        -q <query CMAP> \
        -r <sequence reference CMAP>\
        -p $Bionano/Pipeline/11162020 \
        -a  $Bionano/RefAligner/11442.11643rel/RefAlinger/optArgu
ments_nonhaplotype_saphyr_human.xml \
        -n <number of threads to use, default 4>
```

第六步：混合组装 (Hybrid Scaffold)，以单酶系统 (Single-Enzyme) 为例。

```
    perl $Bionano/HybridScaffold/11162020/hybridScaffold.pl
        -n <sequence file in FASTA format> \
        -b <Bionano CMAP file> \
        -u CTTAAG \
```

```
        -c  $Bionano/HybridScaffold/11162020/hybridScaffold_confi
g_DLE1.xml \
        -r $Bionano/RefAligner/11442.11643rel/RefAlinger \
        -o <output directory> \
        -f \ #是否覆盖之前的输出
        -g \
      -B 2 \ #决定如何处理光学图谱的冲突，可选 1、2 或 3，其中 1 表示不过滤；
2 表示在冲突处分割 contig；3 表示删除冲突的 contig；
        -N 2 \ #决定如何处理物理图谱的冲突，可选 1、2 或 3，其中 1 表示不过
滤；2 表示在冲突处分割 contig；3 表示删除冲突的 contig；
        -p $Bionano/Pipeline/11162020
```

第七步：输出目录下的 agp.fasta 即为组装结果，hybridScaffold_archive.tar.gz 可上传至 Access 进行查看。混合组装作为 BioNano 重要的一步，可以发现基因组组装结果中不正确的部分，从而提高基因组的准确性。组装过程中会检测到光学图谱和物理图谱上的标记存在过多无法匹配的情况，该信息会保留在 conflicts.txt 中，最大允许错配数可以在配置 XML 文件的 assignAlignType.max_overhang 中进行修改。基于上一步中的-B 和-N 参数，软件会进一步通过检查分子的覆盖度(molecule coverage)和冲突标记附近的嵌合质量得分(chimeric quality scores)，对冲突作出对应的处理，如保留该 contig 或者在冲突处分割 contig，亦或者删除该 contig，并将结果最终保存到 conflicts_cut_status.txt 中。进一步，通过在 Access 上可视化后，可以通过肉眼查看冲突位点的图谱信息，检查对比软件处理结果，判断软件处理结果是否正确，不正确的部分可以通过手动修改 conflicts_cut_status.txt 中的状态 (okay 表示不作处理，cut 表示切割， exclude 表示删除该 contig)，保存后重新运行，导出 Access 处理结果即可。具体可以查看：https://bionanogenomics.com/wp-content/uploads/2018/04/30073-Bionano-Solve-Theory-of-Operation-Hybrid-Scaffold.pdf；https://bionanogenomics.com/wp-content/uploads/2017/03/30166-Hybrid-Scaffold-Conflict-Cut-Status-File-Format-Specification-Sheet.pdf

四、Hi-C 测序数据组装

高通量染色质构象捕获技术 (High-throughput chromosome conformation capture，

简称 Hi-C) 为基因组组装过程中 scaffolds 快速锚位提供了契机，该技术通过将空间结构上临近的基因组片段进行甲醛交联等一系列处理，从而得到染色质在空间结构上的交互信息，进而使我们得以研究染色质三维构象。相比于传统的基因组组装方法，基于染色质交互作用组装基因组的策略能够实现染色体水平的基因组组装。其具有实验操作简易、实验和时间成本较低、准确性及分辨率高等特点。因此，其在基因组相对复杂的多倍体和高度杂合的物种中有着更大的应用前景。Hi-C 和 Bionano 光学图谱相似，但是由于 Bionano 很难处理 Hi-C 数据组装结果中包含的朝向/排序错误，且 Bionano 可以高效检测大片段纯合子和杂合子结构变异，所以通常是先进行 Bionano 组装，后运行 HiC 组装。本文档介绍的流程为 Juicer + 3D-DNA + Juicebox。

Juicer(Durand*et al*., 2016)是可以处理上 T 数量级 Hi-C 数据的流程，可以由原始 FASTQ 数据处理得到不同分辨率的 Hi-C 图谱。在使用对应的工具 Juicer tools 下，可以自动 annotate loops 和 contact domains。其可以兼容很多运行平台，包括 LSF, Univa, SLURM, PBS，甚至是个人计算机。下载解压后的软件目录下有对应不同平台的运行目录，所以相比较而言比较友好。对于辅助基因组组装而言，运行路径下需要提前建立 references 和 restriction 目录，分别存放建立好索引的基因组文件和酶切位点文件，运行方法及部分参数如下，其余参数及用法可参考：https://github.com/aidenlab/juicer/wiki/Usage

```
   python $workdir/scripts/get_inres.py $enzyme $species $refer
ence.fa
   bwa index -a bwtsw $reference.fa
   python $workdir/scripts/get_length.py -i $reference.fa -o $r
eference.fa.length

   bash $workdir/scripts/juicer.sh \
    -d $workdir \ ##the top level directory，其中，fastq 目录应包含
fastq 文件，splits 目录运行中会被建立，以存放临时切割的文件，aligned 目录
会被建立，以存放最终比对结果
   -D $workdir \ ##Juicer scripts directory
   -S early \ ##表示运行的不同阶段，必须是"merge", "dedup", "final",
 "postproc", "early"之一，此流程采用 early 对接下一步 3D-DNA。
```

```
    -g $species \
    -z $workdir/references/$reference.fa \ ##reference 基因组
    -t $thread \ ##线程数
    -s $enzyme \ ##限制酶
    -y $workdir/restriction/$reference.enzyme.txt \ ##酶切位点文件
    -p  $workdir/references/$reference.fa.length  ##reference 基因组 contig 长度
```

Juicer 运行完成后，aligned 目录下生成的 merged_nodups.txt 即为下一步 3D-DNA(Dudchenko *et al*., 2017)运行的输入文件之一。3D-DNA 通过运行一系列迭代过程从而达到消除 misjoins 的目的，继而通过 Polish, Split, Seal 和 Merge，生成最终的基因组文件。示例运行脚本如下，详细参数可参考 https://github.com/aidenlab/3d-dna/wiki

```
    bash /soft/3d-dna-master/run-asm-pipeline.sh \
    -m haploid \
    -i 15000 \    #忽略该长度以下的 contig，默认值是 15k
    -r 0 \  #设置迭代次数，如果不打断的话就是 0，打断的话就看情况设置，一般 2-3 轮
    $workdir/references/$reference.fa \
    $workdir/aligned/merged_nodups.txt
```

3D-DNA 运行结果中，包含几部分重要的文件。首先是“.fasta”文件，其中”FINAL.fasta”为最终的染色体形式基因组文件。另一部分是“.hic”文件，可导入可视化软件进行可视化，与之需要一起导入可视化软件 Juicebox 的是对应的 “.assembly” 文件，其中内容包含了输入 contig 在组装不同阶段的 tracks 及 modifications。

Juicebox(Durand*et al*., 2016)是一款采用 java 语言编写的图形化界面工具，既有网页版工具，也可以在本地个人计算机使用。通过一系列操作，可以实现 Hi-C 图谱的动态修改，实现高质量染色体级别基因组的组装。软件整体使用比较简单，具体使用说明可参照: https://github.com/aidenlab/Juicebox/wiki。修改完成后可导出 modified.assembly，并通过该脚本生成最终基因组文件和 agp 文件：https://github.com/phasegenomics/juicebox_scripts。

致谢

衷心感谢武汉未来组李净净和青岛华大基因研究院范广益及其同事对本文的修改和建议。

竞争性利益声明

作者声明没有利益冲突。

参考文献

1. Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., Turner, S. W. and Korlach, J. (2013). Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. Nature Methods 10(6): 563-569.
2. Chin, C.-S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., O'Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., Cramer, G. R., Delledonne, M., Luo, C., Ecker, J. R., Cantu, D., Rank, D. R. and Schatz, M. C. (2016). Phased diploid genome assembly with single-molecule real-time sequencing. Nature Methods 13(12): 1050-1054.
3. Dudchenko, O., Batra, S. S., Omer, A. D., Nyquist, S. K. and Hoeger, M. (2017). De novo assembly of the *Aedes aegypti* genome using Hi-C yields chromosome-length scaffolds. *Science*356(6333): 92-95.
4. Durand, N. C., Robinson, J. T., Shamim, M. S., Machol, I., Mesirov, J. P., Lander, E. S. and Aiden, E. L. (2016). Juicebox provides a visualization system for Hi-C contact maps with unlimited zoom. *Cell Syst* 3(1): 99-101.
5. Durand, N. C., Shamim, M. S., Machol, I., Rao, S. S., Huntley, M. H., Lander, E. S. and Aiden, E. L. (2016). Juicer provides a one-click system for analyzing loop-resolution Hi-C experiments.*Cell Syst* 3(1): 95-98.
6. grandomics.com. Retrieved fromhttps://www.grandomics.com/news/bionano_gxtpzszm/
7. Hu, J. Retrieved from https://github.com/Nextomics/NextDenovo
8. Hu, J., Fan, J., Sun, Z. and Liu, S. (2020). NextPolish: a fast and efficient genome polishing tool for long-read assembly. *Bioinformatics* 36(7): 2253-2255.

9. Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H. and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation.*Genome Res* 27(5): 722-736.
10. Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H., Remington, K. A., Anson, E. L., Bolanos, R. A., Chou, H. H., Jordan, C. M., Halpern, A. L., Lonardi, S., Beasley, E. M., Brandon, R. C., Chen, L., Dunn, P. J., Lai, Z., Liang, Y., Nusskern, D. R., Zhan, M., Zhang, Q., Zheng, X., Rubin, G. M., Adams, M. D. and Venter, J. C. (2000). A whole-genome assembly of Drosophila. *Science* 287(5461): 2196-2204.
11. PacificBiosciences. Bioinformatics-Training. Retrieved from https://github.com/PacificBiosciences/Bioinformatics-Training/wiki/Large-Genome-Assembly-with-PacBio-Long-Reads
12. Ruan, J. and Li, H. (2020). Fast and accurate long-read assembly with wtdbg2. *Nat Methods* 17(2): 155-158.
13. Walker, B. J., Abeel, T., Shea, T., Priest, M., Abouelliel, A., Sakthikumar, S., Cuomo, C. A., Zeng, Q., Wortman, J., Young, S. K. and Earl, A. M. (2014). Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One* 9(11): e112963.
14. Wenger, A. M., Peluso, P. and Rowell, W. J. (2019). Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat Biotechnol*37(10): 1155-1162.