Guidelines for Running Bionano Solve on the Command Line

Document Number: 30205

Document Revision: G

Table of Contents

Legal Notice

For Research Use Only. Not for use in diagnostic procedures.

This material is protected by United States Copyright Law and International Treaties. Unauthorized use of this material is prohibited. No part of the publication may be copied, reproduced, distributed, translated, reverse-engineered or transmitted in any form or by any media, or by any means, whether now known or unknown, without the express prior permission in writing from Bionano Genomics. Copying, under the law, includes translating into another language or format. The technical data contained herein is intended for ultimate destinations permitted by U.S. law. Diversion contrary to U. S. law prohibited. This publication represents the latest information available at the time of release. Due to continuous efforts to improve the product, technical changes may occur that are not reflected in this document. Bionano Genomics reserves the right to make changes in specifications and other information contained in this publication at any time and without prior notice. Please contact Bionano Genomics Customer Support for the latest information.

BIONANO GENOMICS DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS DOCUMENT, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TO THE FULLEST EXTENT ALLOWED BY LAW, IN NO EVENT SHALL BIONANO GENOMICS BE LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY, OR UNDER ANY STATUTE OR ON ANY OTHER BASIS FOR SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, MULTIPLE OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THIS DOCUMENT, INCLUDING BUT NOT LIMITED TO THE USE THEREOF, WHETHER OR NOT FORESEEABLE AND WHETHER OR NOT BIONANO GENOMICS IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Patents

Products of Bionano Genomics® may be covered by one or more U.S. or foreign patents.

Trademarks

The Bionano Genomics logo and names of Bionano Genomics products or services are registered trademarks or trademarks owned by Bionano Genomics in the United States and certain other countries.

Bionano Genomics®, Saphyr®, Saphyr Chip®, and Bionano Access® are trademarks of Bionano Genomics, Inc. All other trademarks are the sole property of their respective owners.

No license to use any trademarks of Bionano Genomics is given or implied. Users are not permitted to use these trademarks without the prior written consent of Bionano Genomics. The use of these trademarks or any other materials, except as permitted herein, is expressly prohibited and may be in violation of federal or other applicable laws.

Introduction

Bionano Solve is a set of tools for analyzing Bionano data (generated from Saphyr and Irys). They have been tested and optimized for Saphyr Compute and Bionano Compute servers. You can utilize Bionano Solve through Bionano Access® (Bionano user interface for data analysis) or via the command-line. In this document, we describe how to utilize Bionano Solve through the command-line.

The latest Bionano Solve package (folder "tools") can be downloaded from the "Software Downloads" webpage at the Bionano Genomics website. Please refer to Bionano Solve Software Installation Guide (PN 30182). Once installed, the pipeline folder is located either by itself on custom servers or in the "tools" folder on the Saphyr/Bionano Compute server. During each installation of Bionano Solve at the Saphyr/Bionano Compute server, the symbolic link (1.0) is updated to reference the newly released version of the Bionano Solve pipeline for Access. Pre-existing Solve pipeline packages are also kept.

Below is a table of folders in the Bionano Solve package.

Folder	Description**
/bionano_packages	R and python packages used for Bionano Solve pipeline
/cohortQC	Scripts for generating MQR and other cohort-based metrics
/FSHD	Scripts for analyzing regions relevant to FSHD
/HybridScaffold	Scripts for single-enzyme and two-enzyme hybrid scaffold
/Pipeline	Scripts for *de novo* assembly pipeline and other utilities
/Process_Control_Datasets	Scripts for processing control datasets
/RefAligner	Binary tools for alignment and assembly*
/RefGenome	CMAP files for human and various commonly studied species
/SMSV	Scripts for Rare Variant Pipeline
/SVviewer	Script for visualizing inversions in Bionano Access
/VariantAnnotation	Tools for annotating and validating SV calls
/VCFConverter	Scripts for converting SV and CNV calls to VCF format

*The main RefAligner binary in this folder is for CPUs with avx2. When utilizing RefAligner, it firstly checks the hardware, if needed, it will automatically switch to the alternative binary, either sse or avx in the subdirectories. A warning message may be printed: "WARNING: host ypsilon4 does not support Intel avx2 instruction set."

**As of version 3.6, Bionano Solve utilizes Docker to encapsulate all the dependencies required to run the pipelines. Please refer to P/N 20192 Bionano Solve Installation Guide for more details. You could also find Docker command lines in this documentation.

Manipulate a Molecule File (BNX)

In silico representations of molecules generated from Bionano Saphyr and Irys instruments are contained in BNX files. BNX files can be imported into Access for further analysis, including merging with other BNX files, MQR (short for Molecule Quality Report, a data quality summary based on molecule alignment to a reference CMAP) and *de novo* assembly.

The following describes some of the commonly used commands for working with .BNX files.

   a) Filtering

   - **Label SNR filter**

      o   BNX files generated from a Saphyr instrument with ICS 3.1.3.4 or older

Human samples: Label SNR threshold computation and label SNR filtering are included and run as part of the assembly pipeline (with human reference provided). No additional filtering step is needed.

Non-human samples: A separate label SNR filtering step is needed before running the assembly pipeline using a Perl script. The script uses a histogram-based method to differentiate between noise and real labels.

The script is located in the "Pipeline" directory. Please use the following command to apply label SNR filter.

Note 1: RefAligner needs to be added into the bin path, please use the command line below to achieve that. BNX files generated by the new image detection algorithm in Bionano Access 1.2 typically do not need to be SNR-filtered.
$ export PATH=$PATH:/home/bionano/tools/pipeline/1.0/RefAligner/1.0
perl /home/bionano/tools/pipeline/1.0/cohortQC/1.0/filter_SNR_dynamic.pl -i <input BNX> -o <output file full name including .bnx suffix>

Here is an example Docker command to run SNR filtering:

*docker run --network=host --user 1000:1000 --rm --privileged *
        *-v /home/bionano:/home/bionano *
        *--env PATH=$PATH:/home/bionano/tools/pipeline/1.0/RefAligner/1.0 *
        *-w ${PWD} *
        *bionanodocker/bionano-docker-pipeline:latest *
        */bin/bash -c "perl /home/bionano/tools/pipeline/1.0/cohortQC/1.0/filter_SNR_dynamic.pl *
         -i <input.bnx> -o <output.bnx>"

This step only needs to be applied to BNX files generated directly from an instrument. In AutoDetect or Irysview, a label SNR filter is applied to the RawMolecules.bnx file. The label SNR cutoff is either determined dynamically or statically, according to instrument types,. Starting with the resulting file (Molecules.bnx file), there is no need to determine the label SNR threshold or apply any additional label SNR filter before running the assembly pipeline.

Note 2: Label SNR filter is no longer needed for BNX files generated from a Saphyr instrument with ICS 4.8 or greater.

- **Applying molecule minimum length filter (the -minlen parameter)**

The RefAligner binary is located in the "RefAligner" directory.

RefAligner -i <input BNX file> -minlen <minlen threshold, unit kbp> -merge -o <output prefix> -bnx

Here is an example Docker command to filter BNX:

*docker run --user 1000:1000 --rm –privileged *
    -v /home/bionano:/home/bionano
    *bionanodocker/bionano-docker-pipeline:latest *
    */home/bionano/tools/pipeline/1.0/RefAligner/1.0/avx/RefAligner *
    -i <input BNX file> -minlen <minlen threshold, unit kbp> -merge -o <output prefix> -bnx

- **Subsampling (the -randomize and -subset parameters)**

RefAligner -i <input BNX file> -randomize 1 -subset 1 <number of desired molecules> -merge -o <output prefix> -bnx

For example, to generate a BNX file named "output.bnx" by randomly subsampling 1,500,000 molecules from a BNX file named "input.bnx"

RefAligner -i input.bnx -randomize 1 -subset 1 1500000 -merge -o output -bnx

Here is an example Docker command to subsample molecules:

*docker run --user 1000:1000 --rm –privileged *
    -v /home/bionano:/home/bionano
    *bionanodocker/bionano-docker-pipeline:latest *
    */home/bionano/tools/pipeline/1.0/RefAligner/1.0/avx/RefAligner *
    -i input.bnx -randomize 1 -subset 1 1500000 -merge -o output -bnx

b) **Merging**

Compile the paths of molecule BNX files you would like to merge in a text file (e.g. list.txt; each line contains the path to one input BNX file). Then, merge them using the following command.

The RefAligner binary is located in the "RefAligner" directory.

RefAligner -if <text file with molecules paths> -merge -o <output prefix> -bnx -stdout -stderr

Alternatively, one could supply RefAligner with a list of –i arguments for individual BNX files:

RefAligner -i input1 -i input2 -i input3 -merge -o <output prefix> -bnx -stdout –stder

Here is an example Docker command to merge BNX:

*docker run --user 1000:1000 --rm –privileged *
*   -v /home/bionano:/home/bionano*
*   bionanodocker/bionano-docker-pipeline:latest *
*   /home/bionano/tools/pipeline/1.0/RefAligner/1.0/avx/RefAligner *
*   -i input.bnx -i input2 -i input3 -merge -o <output prefix> -bnx -stdout –stder*

Besides, you could also use merco to merge bnx files and here is an example command:

/home/bionano/tools/access/1.0/merco -o output.bnx input1.bnx input2.bnx

c) Alignment to Bionano assembly or sequence reference CMAP

The align_bnx_to_cmap.py script is located in the "Pipeline" directory. Please use the following command to align BNX to assembly or reference cmap.

python /home/bionano/tools/pipeline/1.0/Pipeline/1.0/align_bnx_to_cmap.py --mol <input BNX file> -- pipeline <path to pipeline> --ra <path to RefAligner> --nthreads 56 --output <output folder> --ref <path to assembly or ref cmap> --optArgs <path to optArgs.xml> --prefix <prefix>

For example,

*python /home/bionano/tools/access/1.0/../../pipeline/1.0/Pipeline/1.0/align_bnx_to_cmap.py *
*--mol /home/bionano/access/local/jobs/JOBID/input.bnx *
*--pipeline /home/bionano/tools/pipeline/1.0/Pipeline/1.0 *
*--ra /home/bionano/tools/pipeline/1.0/RefAligner/1.0 *
*--nthreads 56 *
*--output /home/bionano/access/local/jobs/JOBID/output *
*--ref /home/bionano/access/share/jobs/JOBID/reference.cmap *

*--optArgs /home/bionano/access/share/jobs/JOBID/optArguments_nonhaplotype_noES_saphyr.xml *

--prefix exp

Here is an example Docker command to align BNX to reference CMAP:

*docker run --network=host --user 1000:1000 --rm --privileged *

*   -v ${SGE_ROOT}:/home/sge *

*   -v /home/bionano:/home/bionano *

*   -v /home/bionano/tools/pipeline/1.0/Pipeline/1.0:/pipeline *

*   -w ${PWD} *

*   bionanodocker/bionano-docker-pipeline:latest *

*   /bin/bash -c "source activate bionano_minimal;*

*   python /home/bionano/tools/access/1.0/../../pipeline/1.0/Pipeline/1.0/align_bnx_to_cmap.py *

*   --mol /home/bionano/access/local/jobs/JOBID/input.bnx *

*   --pipeline /home/bionano/tools/pipeline/1.0/Pipeline/1.0 *

*   --ra /home/bionano/tools/pipeline/1.0/RefAligner/1.0 *

*   --nthreads 56 *

*   --output /home/bionano/access/local/jobs/JOBID/output *

*   --ref /home/bionano/access/share/jobs/JOBID/reference.cmap *

*   --optArgs*

*/home/bionano/access/share/jobs/JOBID/optArguments_nonhaplotype_noES_saphyr.xml *

*   --prefix exp"*

d) Generate Molecule Quality Report

The MQR.pl script is located in the "cohortQC" directory. Please use the following command to align BNX to assembly or reference CMAP.

perl /home/bionano/tools/pipeline/1.0/cohortQC/1.0/MQR.pl -i <input BNX file> -l <Minimum length of the molecule in kb> -n <Minimum nicks in each channel of the molecule> -o <path to output>

For example,

*perl /home/bionano/tools/pipeline/1.0/cohortQC/1.0/MQR.pl *
*-P /home/bionano/access/local/jobs/JOBID/MQR/snr.pdf *
*-i /home/bionano/access/local/jobs/JOBID/MQR/input.bnx *
-j
*'[{"minLengthKb":150,"minLabels":9,"subSamplingRange":10000,"subSamplingSeed":17,"Population":0,"usechannel":1,"samplename":"SampleName","jobid":JOBID,"RCmap":"/home/bionano/access/share/jobs/JOBID/hg38_DLEI_0kb_0labels.cmap"}]' *

-o /home/bionano/access/local/jobs/JOBID/MQR

Here is an example Docker command to run MQR:

*docker run --user 1000:1000 --rm –privileged *

     -v /home/bionano:/home/bionano

     *-w /home/bionano/access/local/jobs/JOBID/MQR --name jobs_JOBID *

     *bionanodocker/bionano-docker-mqr:latest *

     *perl /home/bionano/tools/pipeline/1.0/cohortQC/1.0/MQR.pl *

     *-P /home/bionano/access/local/jobs/JOBID/MQR/snr.pdf *

     *-i /home/bionano/access/local/jobs/JOBID/MQR/input.bnx *

     -j

*'[{"minLengthKb":150,"minLabels":9,"subSamplingRange":10000,"subSamplingSeed":17,"Population":0,"usechannel":1,"samplename":"SampleName","jobid":JOBID,"RCmap":"/home/bionano/access/share/jobs/JOBID/hg38_DLEI_0kb_0labels.cmap"}]' *

     -o /home/bionano/access/local/jobs/JOBID/MQR

In silico Digestion of a FASTA File (.fa, .fasta)

A FASTA file can be converted to a CMAP file through *in-silico* digestion with a specific enzyme motif. Once converted (*in-silico* digested), the output CMAP file can be imported into Access for further data analysis, such as alignment.

The script fa2cmap_multi_color.pl is located in this directory:
"/home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/scripts/".

Please use "fa2cmap_multi_color.pl -h" for further details and options.

perl fa2cmap_multi_color.pl -i <input FASTA file> -e <Enzyme Name (s) or Sequence (s), followed by channel #> -m <minimum nick sites filter of input contigs, default 0> -M <minimum length filter of input contigs, default 0>

By default, the output files are saved in the same folder as the input FASTA file.

For example, to digest an input FASTA file "a.fa" using the enzyme BspQI for channel 1, and another motif CAAAA for channel 2, please use the following command-line.

perl fa2cmap_multi_color.pl -i a.fa -e BspQI 1 CAAAA 2

For example, to digest an input FASTA file "b.fa" using the enzyme BssSI only and filter out sequence contigs shorter than 20 kbp, please use the following command-line.

perl fa2cmap_multi_color.pl -i b.fa -e BssSI 1 -M 20

Here is an example Docker command to generate *in silico* digestion using the enzyme DLE-1 after filtering out sequence contigs shorter than 150 kbp and less than 9 labels:

*docker run --user 1000:1000 --rm --privileged *
     *-v /home/bionano:/home/bionano *
     *-w /home/bionano/access/local/jobs/JODID --name jobs_JOBID *
     *bionanodocker/bionano-docker-scaffold:latest *
     *perl /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/scripts/fa2cmap_multi_color.pl *
     -i input.fasta. -e DLE1 -m 9 -M 150

To optionally filter the FASTA file to a subset of chromosomes or contigs prior to digestion, or to rename contigs, please use the following command line.

python fasta_filter.py -f <input FASTA file> -i <include_chrs.txt> -o <output FASTA file>

The file include_chrs.txt should list the chromosomes to keep, and what they should be renamed to. Please use "python fasta_filter.py -h" for further details on the format of this file.

*docker run --user 1000:1000 --rm --privileged *
     *-v /home/bionano:/home/bionano *
     *-w /home/bionano/access/local/jobs/JODID --name jobs_JOBID *
     *bionanodocker/bionano-docker-scaffold:latest *
     *python /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/scripts/fasta_filter.py *
     -i input.fasta. -e DLE1 -m 9 -M 150

Generate *de novo* Assembly

The following section describes how to launch a *de novo* assembly. The resulting assembly output package (.tar.gz or .zip) can be imported into Access for visualization.

a) With reference CMAP provided

The script pipelineCL.py is located in the "Pipeline" folder. Please use "python pipelineCL.py -h" for more details and options.

python pipelineCL.py -T 240 -j 60 -N 4 -f 0.2 -i 5 -y -b <input BNX file> -l <output directory > -t < RefAligner binary tools directory> -a <optArgs.xml> -r <reference CMAP> -C <clusterArgs.xml>

Note3: *<output directory>* must end with *<./output>*

For example, to assemble a genome from the molecule file "*Molecules.bnx"*, with human hg19 reference provided on an Saphyr Compute Server.

*python /home/bionano/tools/pipeline/1.0/Pipeline/1.0/pipelineCL.py -T 56 -j 48 -N 4 -f 0.2 -i 5 -y *

*-b /home/bionano/test/Molecules.bnx *

*-l /home/bionano/test/output *

*-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0/ *

*-a /home/bionano/tools/pipeline/1.0/RefAligner/1.0/optArguments_haplotype_saphyr_human.xml *

*-r /home/bionano/tools/pipeline/1.0/RefGenome/hg19_BSPQI_0Kb_0labels.cmap *

-C /home/bionano/tools/pipline/1.0/Pipeline/1.0/clusterArguments_saphyr.xml

-T, -j, -N are related to computation job settings and -f is the fraction of host jobs.

Here is an example to generate a human *de novo* assembly using Docker:

*docker run --network=host --user 1000:1000 --rm --privileged *

   *-v /home/sge:/home/sge *

   *-v /home/bionano:/home/bionano *

   *-w /home/bionano/access/local/jobs/JOBID --name jobs_JOBID *

   *bionanodocker/bionano-docker-pipeline:latest *

   */bin/bash -c "source activate bionano_minimal; *

   *python /home/bionano/tools/access/1.0/../../pipeline/1.0/Pipeline/1.0/pipelineCL.py *

   *-l /home/bionano/access/local/jobs/JOBID/output *

   *-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0 *

   *-C /home/bionano/tools/pipeline/1.0/Pipeline/1.0/clusterArguments_saphyr.xml *

   *-b /home/bionano/access/local/jobs/JOBID/input.bnx -y -d -U -i 5 -F 1 -W 1 -c 1 *

   *-a /home/bionano/access/share/jobs/JOBID/optArguments_haplotype_saphyr_human.xml *

   *-r /home/bionano/access/share/jobs/JOBID/hg38_DLE1_0kb_0labels.cmap *

   *-cm /home/bionano/access/share/jobs/JOBID/hg38_cnv_masks.bed *

   --compute-confidence human_hg38 -f 0.1 -J 56 -TJ 112 -j 56 -jp 48 -je 48 -T 288 -Tp 96 -Te 288 -

N 4

   --docker --autoRestart"

For a human sample assembly, you can add option: *-G <translocation masking bed file for hg19/hg38>* to the command. Custom bed files may be generated and used. Please see document [30110 Theory of Operation, Structural Variant Calling](#) for how to create a custom mask.

All default optArgument xml files (-a) for assembly are located in the folder "RefAligner". A description of each file is available in document 30110 Theory of Operation, Structural Variant Calling.

A few cluster argument xml files (-C) are available for use in the folder "Pipeline".

Cluster argument xml files	Compatible Cluster	Notes
clusterArguments_saphyr.xml	Saphyr Compute Server	
clusterArguments_3100.xml	IrysSolve Compute Server	Xeon Phi servers, but not configured by Bionano for Bionano Solve and Access.
clusterArguments_saphyr_phi.xml	IrysSolve Compute Server	Xeon Phi servers, which have been configured by Bionano for Bionano Solve and Access.

Please see **Appendix** for more details about running *de novo* assembly on a custom server.

The -B (bypass) option of the *de novo* assembly pipeline can be used to skip certain assembly steps and continue assembly using previous results, when an assembly fails in the middle, due to network, hardware or power supply issues. This option is useful to restart an assembly from a specified stage.

a) Without reference or poor quality reference

In the absence of a high-quality reference, the *de novo* assembly pipeline provides an option "-R" for users to first run a preliminary *de novo assembly with default noise parameters, and then use the initial rough assembly as the reference for auto-noise in a subsequent full assembly.*

-R [R]   *Rough assembly: denovo assembly used as autoNoise*

reference for re-assembly; sequence may be optionally

specified as -r, if supplied, will be used for global

scaling and SV calls, but not for autoNoise (optional,

default off, optional argument 0.2-0.9 for fraction of

rough assembly which must align to sequence for

rescaling)

Generate a Genome Map (.cmap) to Reference/Genome Map Alignment

A CMAP (Bionano assembly consensus genome map) file can be aligned to a sequence reference CMAP or another assembly CMAP. The output alignment (trio files, including .xmap, _q.cmap and _r.cmap) can be imported into Access for visualization.

The script runCharacterize.py is located in the "Pipeline" folder.

python runCharacterize.py -t <RefAligner binary> -q <query CMAP> -r <sequence reference CMAP> -p

<Pipeline directory> -a <assembly opt argument used to generate the consensus map> -n <number of threads to use, default 4>

For example, use the following command to align assembly consensus genome map "input.cmap" with reference "hg38.cmap", using 64 nodes. The assembly CMAP was generated earlier using "optArgument_haplotype_saphyr.xml" argument. By default, the output "alignref" folder is saved in the same directory where the query CMAP is stored. In the computation log, it also outputs the alignment statistics.

*python /home/bionano/tools/pipeline/1.0/Pipline/1.0/runCharacterize.py *

*-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0/ *

*-q /home/bionano/cmapalignment/input.cmap *

*-r /home/bionano/tools/pipeline/1.0/RefGenome/hg38_DLE1I_0kb_0labels.cmap *

*-p /home/bionano/tools/pipeline/1.0/Pipeline/1.0/ *

*-a /home/bionano/tools/pipeline/1.0/RefAligner/1.0/optArguments_haplotype_saphyr.xml *

-n 64 1>alignmentstatistics.out

Here is an example command to generate an alignment of genome map against reference map:

*docker run --network=host --user 1000:1000 --rm --privileged *

*-v /home/bionano:/home/bionano *

*-w /home/bionano/access/local/jobs/JOBID/output/.. *

*--name jobs_4465 *

*bionanodocker/bionano-docker-pipeline:latest *

/bin/bash -c "source activate bionano_minimal;

*python /home/bionano/tools/pipeline/1.0/Pipline/1.0/runCharacterize.py *

*-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0/ *

*-q /home/bionano/cmapalignment/input.cmap *

*-r /home/bionano/tools/pipeline/1.0/RefGenome/hg38_DLE1I_0kb_0labels.cmap *

*-p /home/bionano/tools/pipeline/1.0/Pipeline/1.0/ *

*-a /home/bionano/tools/pipeline/1.0/RefAligner/1.0/optArguments_haplotype_saphyr.xml *

-n 64 1>alignmentstatistics.out"

Generate Hybrid Scaffold

a) Single-Enzyme Hybrid Scaffold Pipeline

The Hybrid Scaffold pipeline can be run using the command:

perl hybridScaffold.pl

-n <sequence file in FASTA format>

-b <Bionano CMAP file>

-c <hybrid scaffold configuration file in XML format>

-r <RefAligner binary file>

-o <output directory>

-B <conflict filter level genome maps: 1, 2, or 3>

-N <conflict filter level for sequences: 1, 2, or 3>

-f <a flag to overwrite existing files; optional>

-x <a flag to align molecules to hybrid scaffolds and genome maps>

-y <a flag to generate chimeric quality score for the input genome maps>

-M <a conflict resolution text file; optional>

-m <molecule BNX file to align molecules to genome maps and hybrid scaffolds; optional> -p <de novo assembly pipeline script; optional but needed by the -x option>

-q <de novo assembly optArguments XML files; optional but needed for the -x option>

-e <de novo assembly noise parameter ERRBIN or ERR file; recommended for -y option>

-v <a flag to print the pipeline version>

Here is an example:

*perl /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/hybridScaffold.pl *

*-n /home/bionano/data/seq/MG1655.fa *

*-b /home/bionano/assembly/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap *

*-u CTTAAG *

*-c /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/hybridScaffold_config_DLE1.xml *

*-r /home/bionano/tools/pipeline/1.0/RefAligner/1.0/RefAligner *

*-o /home/bionano/data/hybridscaffold/output *

*-f *

*-g *

*-B 2 *

*-N 2 *

*-p /home/bionano/tools/pipeline/1.0/Pipeline/1.0 *

*-z cur_results_JOBID.zip *

*-w status_JOBID.txt *

-o /home/bionano/access/local/jobs/JOBID/output

Here is an example to generate a single-enzyme hybrid scaffold using Docker:

*docker run --user 1000:1000 --rm --privileged *

       *-v /home/bionano:/home/bionano *

*-w /home/bionano/access/local/jobs/JODID --name jobs_JOBID *

*bionanodocker/bionano-docker-scaffold:latest *

*perl /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/hybridScaffold.pl *

*-n /home/bionano/data/seq/MG1655.fa *

*-b /home/bionano/assembly/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap *

*-u CTTAAG *

-c /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0*/hybridScaffold_config_DLE1.xml *

-r /home/bionano/tools/pipeline/1.0/RefAligner/1.0/RefAligner \

*-o /home/bionano/data/hybridscaffold/output *

*-f *

*-g *

*-B 2 *

*-N 2 *

*-p /home/bionano/tools/pipeline/1.0/Pipeline/1.0 *

*-z cur_results_JOBID.zip *

*-w status_JOBID.txt *

-o /home/bionano/access/local/jobs/JOBID/output

b) Two-Enzyme Hybrid Scaffold Pipeline

The pipeline can be called by running the script runTGH.R using the following syntax and options:

Rscript /<hybrid scaffold path>/runTGH.R

runTGH.R [--] [--help] [--override] [--opts OPTS] [--BNGPath1 BNGPATH1] [--BNGPath2 BNGPATH2] [-- NGSPath NGSPATH] [--OutputDir OUTPUTDIR] [--RefAlignerPath REFALIGNERPATH] [--RunFlags RUNFLAGS] [--Enzyme1 ENZYME1] [--Enzyme2 ENZYME2] [- -ManualCut1 MANUALCUT1] [-- ManualCut2 MANUALCUT2] [--tar TAR] [--status STATUS]

An example command line is shown below:

*Rscript /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/runTGH.R *

-R /home/bionano/tools/pipeline/1.0/RefAligner/1.0/RefAligner

-b1 /home/bionano/assembly_BssSI/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap

-b2 /home/bionano/assembly_BspQI/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap

-N /home/bionano/data/seq/input.fa

-e1 CACGAG

-e2 GCTCTTC

-t cur_results.tar.gz

-s status.txt

-f

/home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/TGH/hybridScaffold_two_enzymes.xml

-O /home/bionano/hybridscaffold/output

Alternatively, all the command-line parameters can be also specified in an xml file and pass to the main control script by running:

Rscript runTGH.R hybridScaffold_two_enzymes.xml

Here is an example to generate a two-enzyme hybrid scaffold using Docker:

*docker run --user 1000:1000 --rm --privileged *

  *-v /home/bionano:/home/bionano *

  *-w /home/bionano/access/local/jobs/4483 --name jobs_4483 *

  *bionanodocker/bionano-docker-scaffold:latest *

  *Rscript /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/runTGH.R *

  -R /home/bionano/tools/pipeline/1.0/RefAligner/1.0/RefAligner

  -b1 /home/bionano/assembly_BssSI/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap

  -b2 /home/bionano/assembly_BspQI/output/contigs/exp_refineFinal1/EXP_REFINEFINAL1.cmap

  -N /home/bionano/data/seq/input.fa

  -e1 CACGAG

  -e2 GCTCTTC

  -t cur_results.tar.gz

  -s status.txt

  -f

  /home/bionano/tools/pipeline/1.0/HybridScaffold/1.0/TGH/hybridScaffold_two_enzymes.xml

  -O /home/bionano/hybridscaffold/output

Please refer to document [30073 Theory of Operation, Hybrid Scaffold](#) for details. The hybrid scaffold output package (.zip) can be imported into Access for visualization.

Generate Rare Variant Calls

The Rare Variant Pipeline interface is similar to the *de novo* assembly pipeline interface. The same command line parameters are used when applicable.

*python -u /home/bionano/tools/pipeline/1.0/SMSV/1.0/bionano_bin/single_molecule_cli.py *

*-b input.bnx -a /home/bionano/tools/pipeline/1.0/SMSV/1.0/SingleMoleculePipelineParameters.xml *

*-t home/bionano/tools/pipeline/1.0/RefAligner/1.0/ *

*-l output *

*-C /home/bionano/tools/pipeline/1.0/SMSV/1.0/SingleMoleculePipelineClusterParameters.xml *

-r /home/bionano/ref/Reference.cmap --log run_log.txt -T 64 -op 0.001 -c 1

Here is an example Docker command:

*docker run --network=host --user 1000:1000 --rm --privileged *

     *-v /home/sge:/home/sge -v /home/bionano:/home/bionano *

     *-e DRMAA_LIBRARY_PATH=/home/sge/lib/lx-amd64/libdrmaa.so *

     -e

PYTHONPATH=/home/bionano/tools/pipeline/1.0/SMSV/1.0/src:/home/bionano/tools/pipeline/1.0/SMSV/

*1.0/src/   bionano_pipeline *

     *-w /home/bionano/access/local/jobs/JOBID --name jobs_JBOID *

     *bionanodocker/bionano-docker-pipeline:latest *

     */bin/bash -c *

     'source activate bionano3; python

*/home/bionano/tools/pipeline/1.0/SMSV/1.0/bionano_bin/single_molecule_cli.py *

     -b input.bnx -a

*/home/bionano/tools/pipeline/1.0/SMSV/1.0/SingleMoleculePipelineParameters.xml *

     *-t home/bionano/tools/pipeline/1.0/RefAligner/1.0/ *

     *-l output *

     *-C /home/bionano/tools/pipeline/1.0/SMSV/1.0/SingleMoleculePipelineClusterParameters.xml *

     -r /home/bionano/ref/Reference.cmap --log run_log.txt -T 64 -op 0.001 -c 1'

Generate Variant Annotation

Please see document [30190 Bionano Solve Theory of Operation, Variant Annotation Pipeline](#) for more details. The following section describes the locations of necessary files to run the pipeline.

When configuring the relevant parameter files, please also note the following:

- The daughter_id, mother_id or father_id can be any random numbers. They are referred in the pipeline log file.

- Individual control variant database files are saved in "config" folder of the Variant Annotation Pipeline.

- In the step3 section of the parameter file,

  refAligner=<path to RefAligner binary>,

  alignmt_param_xml=<path to parentAlignMol.xml in "config" folder of Variant Annotation Pipeline folder>

- overlap_database=<input file which contains annotated gene position in the human reference, the default ones are available in the "config" folder of Variant Annotation Pipeline folder>

After configuring the relevant parameter file, in the shell script "env.sh", please put:

export PERL5LIB="$(pwd)/lib":$PERL5LIB

export PERL5LIB="$(pwd)/XML-Simple-2.22/lib":$PERL5LIB

perl <variant annotation perl script> <full path of the parameter file>

Please make sure the modified "env.sh" file is stored in the original folder in Variant Annotation Pipeline folder. Then users can run it with:

sh <your_env.sh>

Here is an example docker command line to run a trio variant analysis:

*docker run --network=host --user 1000:1000 --rm --privileged *
    -v /home/bionano:/home/bionano
    --env
*PERL5LIB=/home/bionano/tools/pipeline/1.0/VariantAnnotation/1.0/lib:/home/bionano/tools/pipeline/1.0/VariantAnnotation/1.0/XML-Simple-2.22/lib:/home/bionano/tools/pipeline/1.0/VariantAnnotation/1.0 *
    *-w /home/bionano/access/share/jobs/JOBID --name jobs_JOBID *
    *bionanodocker/bionano-docker-vap:latest *
    perl /home/bionano/tools/pipeline/1.0/VariantAnnotation/1.0/variant_annotation.pl
/home/bionano/tools/pipeline/1.0/VariantAnnotation/1.0/variant_annotation_param_db.txt

Generate Copy Number Profile

Human copy number profiling is part of the *de novo* assembly pipeline by default, when the sequence reference hg19 or hg38 CMAP with Nt.BspQI, Nb.BssSI and DLE-1 is provided. It can also be generated independently and separately from the *de novo* assembly pipeline:

Rscript CNV.r --args --cnvScriptsFolder [path containing CNV.R] --sampleName [string] --sampleRcmap [r.cmap]

For example,

/bin/Rscript
/home/bionano/tools/pipeline/1.0/Pipeline/1.0/Analysis/cnv/cnv_svd_outliers_reference_08292017/CNV.R
--args --cnvScriptsFolder

/home/bionano/tools/pipeline/1.0/Pipeline/1.0/Analysis/cnv/cnv_svd_outliers_reference_08292017 --
sampleName exp --sampleRcmap
/home/bionano/access/assembly/output/contigs/alignmolvref/merge/alignmolvref_merge_r.cmap --
resultsDir /home/bionano/access/assembly/output/contigs/alignmolvref/copynumber --outliersProbability
0.001

Here is an example using Docker to generate copy number profile:

*docker run --user 1000:1000 --rm --privileged *
     *-v /home/bionano:/home/bionano *
     *-w /home/bionano/access/local/jobs/JOBID --name jobs_JBOID *
     *bionanodocker/bionano-docker-cnv:latest *
     /bin/Rscript
/home/bionano/tools/pipeline/1.0/Pipeline/1.0/Analysis/cnv/cnv_svd_outliers_reference_08292017/CNV.R
**
     *--args --cnvScriptsFolder *
     /home/bionano/tools/pipeline/1.0/Pipeline/1.0/Analysis/cnv/cnv_svd_outliers_reference_08292017
**
     *--sampleName exp *
     --sampleRcmap
*/home/bionano/access/assembly/output/contigs/alignmolvref/merge/alignmolvref_merge_r.cmap *
     *--resultsDir /home/bionano/access/assembly/output/contigs/alignmolvref/copynumber *
     --outliersProbability 0.001

Please go to 30210 *Introduction to Copy Number Analysis* for more details.

Generate FSHD Analysis

The FSHD analysis pipeline first performs a local assembly of regions of interest and then analyzes the resulting genome maps in the chr4 and chr10 D4Z4 regions in order to size the repeats and assign haplotypes to the alleles. Other stable regions of the genome are also analyzed as part of the quality-control process.

Here is the command line to run guided assembly:

python pipelineCL.py -T 240 -j 60 -N 4 -f 0.2 -i 5 -y -U -b <input BNX file> -l <output directory > -t < RefAligner binary tools directory> -a <optArgs.xml> -r <reference CMAP> -C <clusterArgs.xml> -seed <seed cmap> -guidedB

For example,

*python /home/bionano/tools/pipeline/1.0/Pipeline/1.0/pipelineCL.py *

*-l /home/bionano/access/output *

-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0

-r /home/bionano/tools/pipeline/1.0/RefGenome/hg38_DLE1_0kb_0labels.cmap

 -b /home/bionano/input/RawMolecules.bnx

 -C /home/bionano/tools/pipeline/1.0/Pipeline/1.0/clusterArguments_saphyr.xml

 -a

/home/bionano/tools/pipeline/1.0/RefAligner/1.0/optArguments_haplotype_DLE1_saphyr_human_D4Z4.x

ml

 -seed

/home/bionano/tools/pipeline/1.0/FSHD/1.0/resources/guided_assembly/hg38_DLE1_D4Z4_with_control

*_regions.cmap *

-guidedB -y -U -T 288 -N 6 -j 144 -i 5 -f 0.2 -F 1 -d -W 0.4 -J 48

Here is an example Docker command line to run guided assembly:

*docker run --network=host --user 1000:1000 --rm --privileged *

> *-v /home/bionano:/home/bionano*

> *-w /home/bionano/access/local/jobs/JOBID --name jobs_JOBID *

> *bionanodocker/bionano-docker-pipeline:latest *

> */bin/bash -c *

> *"source activate bionano_minimal; *

> *python /home/bionano/pipeline/1.0/Pipeline/1.0/pipelineCL.py *

> *-l /home/bionano/access/output *

> *-t /home/bionano/tools/pipeline/1.0/RefAligner/1.0*

> *-r /home/bionano/tools/pipeline/1.0/RefGenome/hg38_DLE1_0kb_0labels.cmap*

> *-b /home/bionano/input/RawMolecules.bnx*

> *-C /home/bionano/tools/pipeline/1.0/Pipeline/1.0/clusterArguments_saphyr.xml*

> *-a*

/home/bionano/tools/pipeline/1.0/RefAligner/1.0/optArguments_haplotype_DLE1_saphyr_human_D4Z4.x

ml

> *-seed*

/home/bionano/tools/pipeline/1.0/FSHD/1.0/resources/guided_assembly/hg38_DLE1_D4Z4_with_control

*_regions.cmap *

> *-guidedB -y -U -T 288 -N 6 -j 144 -i 5 -f 0.2 -F 1 -d -W 0.4 -J 48"*

After that you can run FSHD data analysis using this command line:

Python run_FSHD_analysis.py [Sample Name] -a [final map directory] --log [name of a log file]

For example,

python /home/bionano/tools/pipeline/1.0/FSHD/1.0/run_FSHD_analysis.py 'Test' -a
/home/bionano/access/output/contigs/exp_refineFinal1_sv --verbose --log FSHD_analysis.log

Here is an example Docker command to run FSHD data analysis:

*docker run --network=host --user 1000:1000 --rm --privileged *
     -v /home/bionano:/home/bionano
     *-w /home/bionano/access/local/jobs/JOBID --name jobs_JOBID *
     *bionanodocker/bionano-docker-pipeline:latest *
     */bin/bash -c *
     *"source activate bionano_minimal; *
     *python /home/bionano/tools/pipeline/1.0/FSHD/1.0/run_FSHD_analysis.py 'Test' *
     *-a /home/bionano/access/output/contigs/exp_refineFinal1_sv *
     --verbose --log FSHD_analysis.log"

The "final map directory" should be the "exp_refineFinal1_sv" directory in a successfully completed assembly. The pipeline also looks for molecule-to-map alignment data under the directory "exp_refineFinal1/alignmol/" directory; the data are used for haplotype assignment and the calculation of statistics of the repeats. This directory should be part of the output from a successfully completed assembly.

Please go to 30321 *Bionano Solve Theory of Operation, Bionano EnFocus™ FSHD Analysis* for more details.

Convert SMAP and CNV (optional) to VCF Format

The SMAP to VCF converter tool converts insertion, deletion, duplication, and inversion and translocation breakpoint calls in an SMAP and duplication and deletion in optional CNV file to dbVar-compliant VCF v4.2 format.

There are three required inputs to the script:

1) the SMAP file to convert,

2) the reference (_r.cmap file) which is generated with the SMAP,

3) the map-to-reference alignment (.xmap file) which is generated with the SMAP.

The QUAL score is calculated as -10 times the log base 10 of (1 minus the confidence) where confidence is the SMAP confidence score for the given call. The QUAL ceiling is set at 20. The uncertainty of the start and end positions of the SV call are provided in the INFO column. This output VCF file can be used for further downstream analysis using any tools that take VCF files as input.

Here is the standalone script to convert Bionano smap file format to vcf:

bionano_vcf_converter.py [-h] [-s SMAPPATH] [-r REFCMAPPATH] [-x XMAPPATH]

[-n SAMPLE] [-o OUTPUT_PREFIX] [-a REF_ACCESSION]

[-b HUMAN_BOOL] [-i EXP_ID] [-m MASK] [-c CNVPATH]

For example,

*python /home/bionano/tools/pipeline/1.0/VCFConverter/ bionano_vcf_converter.py \
-s /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1.smap \
-r /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1_r.cmap \
-x /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1.xmap \
-n SampleName -o Output_Prefix -a GCA_000001405.1 -b True -i EXP_ID –m 1 \
-c /home/bionano/access/assembly/output/contigs/alignmolvref/copynumber/cnv_calls_exp.txt*

Here is an example Ddocker command to convert SMAP and CNV to VCF:

*docker run --network=host --user 1000:1000 --rm --privileged \
        -v /home/bionano:/home/bionano\
        -w /home/bionano/access/local/jobs/JOBID --name jobs_JOBID \
        bionanodocker/bionano-docker-pipeline:latest \
        /bin/bash -c \
        "source activate bionano_minimal; \
        python /home/bionano/tools/pipeline/1.0/VCFConverter/smap_to_vcf_v2.py \
        -s /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1.smap \
        -r /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1_r.cmap \
        -x /home/bionano/access/assembly/output/contigs/exp_refineFinal1_sv/exp_refineFinal1.xmap \
        -n SampleName -o Output_Prefix -a GCA_000001405.1 -b True -i EXP_ID –m 1 \
        -c /home/bionano/access/assembly/output/contigs/alignmolvref/copynumber/cnv_calls_exp.txt"*

Appendix: Running *de novo* Assembly on a Custom Server

Bionano Solve™ pipeline has been optimized and tested on Saphyr and Bionano Compute solutions. Currently, Bionano Genomics *does not* guarantee pipeline perfomance, support configuration, or troubleshoot pipeline performance issues on computer server, which is not provided by Bionano Genomics. Please refer to this documentation 30172 Bionano Solve Software Installation Guide for the prerequisites.When running the assembly command-line on a single node, user does not supply the -C <cluster argument.xml file> parameter. Please see section "Generate *de novo* Assembly" for more guidance.

The script "pipelineCL.py" is available in the "Pipeline" folder. Please use "*python pipelineCL.py –h*" for more details and options.

python /Pipeline/pipelineCL.py -l <path to output folder> -t <path to RefAligner directory> -b <path to input molecules BNX file> -r <path to reference CMAP> -y <perform autonoise> -i <# of extension and merge iterations, default is 1> -a <assembly parameters XML file>

-N 2 <# of split BNX files; number of pairwise jobs is N*(N-1)/2, default is 2> -f 0.10 <Run this fraction of largest grouped jobs on large memory host> -T 56 <Available threads per node> -j 56 <# of maximum threads per job>

Running the assembly command-line on a cluster requires a properly configured "clusterArgument.xml" file (-C parameter). In addition to previously mentioned hardware and dependency requirements, it also requires a batch system that supports the Distributed Resource Management Application API (DRMAA). The submission hosts for the cluster also require the Python DRMAA module.

Users need to configure the "clusterArguments_custom.xml" file (available at Bionano GitHub page) for their own cluster.

Basic configuration recommendation is listed below. Very likley, more configuration of the "clusterArgument_custom.xml" file is required to run the assembly command-line successfully on a custom cluster.

In the "clusterArguments_custom.xml" file, it assumes 3 types of nodes present in 3 SGE (Sun Grid Engine) parallel environments, and customers must assign all compute nodes to one (or more) of these 3 parallel environments:

a) smplargest: nodes with 256 GB memory (or more). User cluster must have at least 1 such node.

b) smplarge: nodes with 128 GB memory (or more).

c) smp: nodes with 32 GB memory (or more). It can include the same nodes as in smplarge.

d) The names of the parallel environments ("smp" or "smplarge" or "smplargest") can be changed by users. For example, the custom cluster may already have suitable configured parallel environments which include all 256 GB+ or 128 GB+ or 32 GB+ nodes, so changing the names in "clusterArgument_custom.xml" file to the existing ones on the customer cluster could be done, instead of defining new parallel environments on the customer cluster. With parallel environments, jobs that require 32 GB can run on nodes that have 128 GB and 256 GB as well.

e) Defining new parallel environments should be done in accordance with the DRMAA compatible batch system. For SGE compatible batch systems, this can be done using the "qconf" command. First the Linux command "qconf –ap smp" would be used to define a parallel environment name "smp" : The only default value that should be modified is the total number of slots available, which should be set to a very large value, eg 9999. Repeat for "smplarge" and "smplargest". The Linux command *"qconf –aq all.q"* would be used to create a new queue called "all.q". The command will pop up an editor, which allows the "qname" to be set to "all.q" and "pe_list" to be set to a list of compute nodes along with the parallel environments they support (eg [host1=smp],[host2=smp smplarge smplargest], ...), where host1 is the hostname or IP address of a compute node with 32 GB, host2 is a compute node with 256 GB etc. The "slots" entry should specify for each compute node the total number of threads (eg [host1=56],[host2=48],…). Similarly the Linux command *"qconf –mq all.q"* can be used to modify an existing SGE queue called "all.q", for example, to add additional parallel environments by modifying the "pe_list".

The clusterArguments_custom.xml assumes that each node can support jobs using 48 threads (56 threads for the 256 GB nodes). If the customer nodes support a different number of threads, improved performance can be obtained by adjusting the "-pe" SGE slots values in the "clusterArguments_custom.xml" file, by scaling the values 47, 27 and 12 up or down proportionately, for example, to replace 47 by one less than the number of slots per node on "smp", "smplarge" or "smplargest" nodes (whichever is least), 27 by one less than half the number of slots per node on "smplargest" nodes, and replace 12 by one quarter the number of slots per node on "smp" nodes.

In addition, all compute nodes added to "all.q" need to have their Linux OS configured with /proc/sys/vm/overcommit_memory set to 1 (unlimited memory overcommit), and vm.max_map_count increased to 256,000 on the 256 GB and 128 GB nodes. Otherwise memory allocations for some jobs may fail.

Technical Assistance

For technical assistance, contact Bionano Genomics Technical Support.

You can retrieve documentation on Bionano products, SDS's, certificates of analysis, frequently asked questions, and other related documents from the Support website or by request through e-mail and telephone.

Type	Contact
Email	**support@bionanogenomics.com**
Phone	**Hours of Operation:** **Monday through Friday, 9:00 a.m. to 5:00 p.m., PST** **US: +1 (858) 888-7663**
Website	**www.bionanogenomics.com/support**

Bionano Genomics, Inc.
9540 Towne Centre Drive, Suite 100
San Diego, CA 92121