THE CENTER FOR GENOME ARCHITECTURE

Baylor College of Medicine & Rice University

Genome Assembly Cookbook

Table of Contents

Chapter 1

Overview and Quick Start

Generating a high-quality genome assembly is a critical foundation for the analysis of any organism. In this cookbook, we have tried to put together a recipe, based on our latest work in the field, for creating high-quality chromosome-length genome assemblies using publicly available tools. Our hope is that, using this recipe, anyone, even without prior experience in assembling genomes, will be able to produce high-quality chromosome-length genomic sequences.

The procedures described here rely on using Hi-C as a source of linking information during genome assembly. The methods used are robust with respect to input data and are compatible with the cheapest data types available at present, allowing, among other things, to produce mammalian genome assemblies for as little as $1000.

Find out more about our work on genome assembly at http://aidenlab.org/assembly.

Overview

The cookbook covers the following topics, in the order that mirrors a typical workflow for a Hi-C-based genome assembly project:

1) DNA-Seq library preparation, sequencing and assembly of the draft genome (Chapter 2);

2) Hi-C library preparation, sequencing and alignment to the draft genome assembly (Chapter 3);

3) Using a fully automatic pipeline to correct misjoins, order, orient and anchor scaffolds from the draft assembly into a candidate chromosome-length assembly (Chapter 4);

4) Manual review of the candidate assembly in Juicebox Assembly Tools for quality control and interactive correction of the automatic output (Chapter 5).

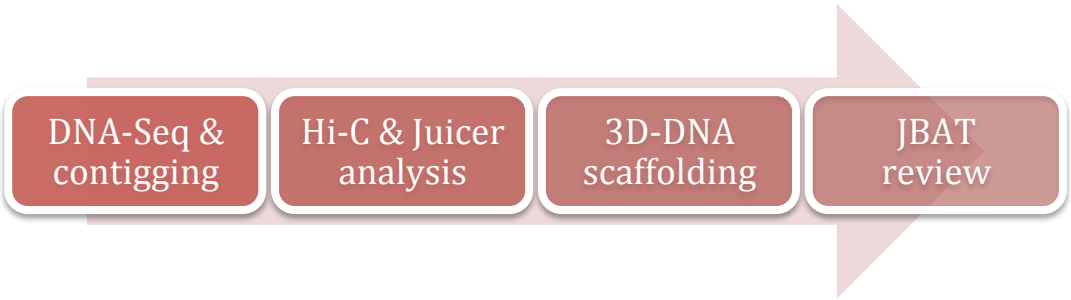

FIGURE 1. Overview of topics covered by this manual. The topics are discussed in the order reflective of a typical workflow for a Hi-C-based genome assembly project.

Quick Start Guide

In this section we will give a quick start guide to follow the $1000 chromosome-length mammalian genome assembly procedure introduced in (Dudchenko et al. 2018), see FIGURE 2. The workflow and respective commands are listed below:

1. DNA sequencing and draft genome assembly (contigging):

    a. Generate a short insert size PCR-free DNA-Seq library (insert size ~400bp) compatible with Illumina platform and sequence it with PE150 reads to collect at least 300 million reads.

    b. Install and assemble the output using w2rap-contigger (https://github.com/bioinfologics/w2rap-contigger), see (B. Clavijo et al. 2017).

w2rap example command

```
./w2rap-contigger -t 48 -m 900 -r R1.fq,R2.fq -p w2rap --dump_all 1

ln –sf a.lines.fasta draft.fa
```

    Skip this step if you already have a draft genome assembly. See Chapter 2 for some comments on DNA-Seq data generation and analysis.

2. Hi-C sequencing and data analysis with Juicer:

    a. Generate an *in situ* Hi-C library (see (Rao, Huntley et al. 2014) for a comprehensive protocol) and sequence it to at least 100 million paired-end reads.

    b. Align the resulting reads to draft genome assembly using Juicer (Durand, Shamim et al. 2016).

Steps to run Juicer on the Hi-C data

bwa index draft.fa

generate_site_positions.py MboI draft draft.fa

./juicer/scripts/juicer.sh –g draft –s MboI –z draft.fa –y draft_MboI.txt –p assembly

See Chapter 3 and https://github.com/theaidenlab/juicer for details on how to set up a working directory to run Juicer. The file necessary for downstream analysis is aligned/merged_nodups.txt, a duplicate-free list of paired alignments from an *in situ* Hi-C experiment.

3. Generate a candidate assembly with 3D *de novo* assembly (3D-DNA) pipeline:

Example command for running 3D-DNA on a draft genome assembly

./3d-dna/run-asm-pipeline.sh draft.fa merged_nodups.txt

See Chapter 4 and https://github.com/theaidenlab/3d-dna for details on how to run and tune 3D-DNA pipeline. The pipeline, among other output, will produce a candidate chromosome-length genome assembly sequence (*.FINAL.fasta*), an associated contact map (*.final.hic*), and a custom assembly file (*.final.assembly*) that stores, in a concise form, the relationship between the original draft sequences and the output.

You can skip this step if you already have a chromosome-length or near chromosome-length genome assembly that you believe you can handle manually. To visualize the draft assembly 'as is' run:

Visualize candidate assembly

awk –f ./3d-dna/utils/generate-assembly-file-from-fasta.awk draft.fa > draft.assembly

./3d-dna/visualize/run-assembly-visualizer.sh draft.assembly merged_nodups.txt

and use the produced *draft.assembly* and the *draft.hic* files in place of the *.final.assembly* and *.final.hic* with Juicebox Assembly Tools as described below.

4. Review the candidate assembly using Juicebox Assembly Tools (JBAT):

Download the latest distribution of desktop Juicebox (≥1.8.8) from https://github.com/theaidenlab/juicebox/wiki/Download and follow this tutorial video (https://www.youtube.com/watch?v=Nj7RhQZHM18) and instructions in Chapter 5 to open the *.final.hic* file ("Open…" menu item in the *File* menu), load the corresponding *.final.assembly* file ("Import Map Assembly" menu item in the *Assembly* menu), examine and rectify the assembly if necessary, and save changes ("Export Assembly" menu item in the *Assembly* menu). Changes will be stored in a form of a modified *.assembly* file. In order to convert it into a fasta sequence run:

Generate a fasta sequence after JBAT review

```
./3d-dna/ run-asm-pipeline-post-review.sh –r draft.final.review.assembly
draft.fasta merged_nodups.txt
```

In addition to the chromosome-length genome sequence (*.FINAL.fasta*), the pipeline will produce a new *.final.hic* file for concluding quality control.

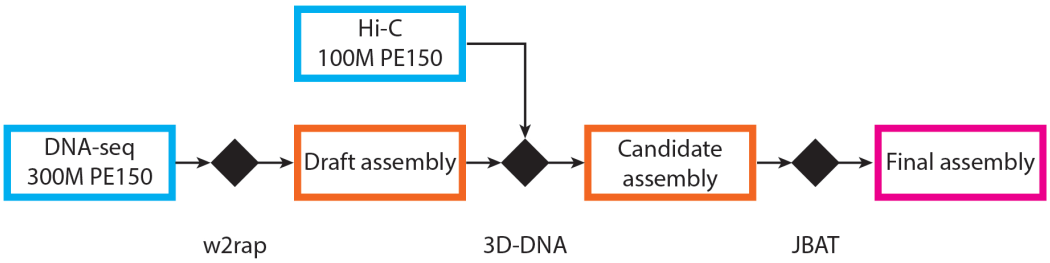

FIGURE 2. Workflow for $1000 chromosome-length mammalian genome assembly, from (Dudchenko et al. 2018).

Contact Information

Feel free to post your questions and comments, suggest improvements and submit bug reports on any of the tools developed by the lab at our forum:

http://www.aidenlab.org/forum.html

This document was assembled by Olga Dudchenko.

Chapter 2

DNA-Seq and Draft Assembly

The typical approach to 3D genome assembly is to generate two types of data: DNA-Seq and Hi-C. Here we use the term DNA-Seq to refer to any type of DNA sequencing that produces a genome-wide draft – fasta-file that contains all (most) of the sequences of the genome of interest but in an often highly fragmented form. (We will refer to the process of making a draft assembly as contigging though depending on the type of data used and the software employed the draft may consist of scaffolds rather then contigs, i.e. allow for presence of gaps in fragments rather than comprise perfectly contiguous individual pieces.)

Often various linking data are available in addition to standard DNA-Seq that can be used to piece the draft fragments together thus increasing the average fragment size and reducing the total number of fragments in the draft assembly. The Hi-C-based approaches described here are, as a rule, compatible with these additional linking methods, but do not require them to ensure whole-chromosome assembly.

Once the draft assembly is generated the Hi-C data are later used to anchor, order and orient the fragments into whole chromosomes. (We refer to the final result of this procedure as chromosome-length scaffolds.) If you already have a draft genome assembly you can skip this chapter. Note that when assembling small bacterial genomes it is sometimes possible to generate the draft sequence from contigging raw Hi-C data thus allowing one to skip DNA-Seq library construction and sequencing, see e.g. (Marbouty et al. 2014).

DNA-Seq Data Models

3D genome assembly is compatible with all DNA-Seq data types of which we are aware including Illumina, Pacific Biosciences, Oxford Nanopore, Sanger and Roche 454 data. The choice in favor of a particular data type is mostly a matter of available resources.

Very often it is possible to generate chromosome-length scaffolds from the cheapest data type available at the moment: Illumina PE150 sequencing (Dudchenko et al. 2018). Following this strategy we have assembled three new mammalian genomes *de novo* to chromosome-length for under $1000 (Dudchenko et al. 2018). The libraries were

prepared using Illumina's TruSeq DNA PCR-Free kit for short insert size DNA-Seq library preparation, following the manufacturer's protocols, aiming for insert size distribution centered at around 400bp. The sequencing was done on a HiSeq X instrument (maximal read length: 2x150).

More expensive data types such as Illumina PE250 sequencing, Pacific Biosciences and Oxford Nanopore sequencing have the capacity to boost contiguity and increase the percentage of anchored sequenced bases. See TABLE 1 for comparison of NA12878 genome assemblies generated using different DNA-Seq data models, from (Dudchenko et al. 2018).

Intermediate Linking Data

Often intermediate linking data are available for a sample that allow to perform some local joining of the draft sequence fragments produced from the original DNA-Seq experiment, thus increasing the average fragment size and reducing the number of fragments in the draft. Typical examples of such data include mate-pair sequencing, Fosmid and BAC sequencing, 10X and optical mapping, *in vitro* Hi-C (Chicago), genetic linkage and population data.

In our experience the algorithms described here are compatible with all of these methods though, as a rule, it is not necessary to employ them to achieve chromosome-length scaffolds if the draft scaffold N50 is ≥10-100kb (though increasing the draft scaffold N50 may allow to incorporate more bases into chromosome-length scaffolds). Note that starting with approximate scaffolds such as pseudomolecules created from linkage data or scaffolds that contain multiple large gaps (≥15kb) such as, for example, generated with optical mapping, can sometimes have detrimental effect on the final assembly and better results can be achieved with current methods by discarding the intermediate data for scaffolding purposes and using it for validation instead. When in doubt, examining the draft and output in Juicebox Assembly Tools is often helpful (see Chapter 5).

Draft Assembly

Just as 3D genome assembly using methods described in this manual is compatible with all DNA-Seq data types it is also compatible with all contiggers recommended for the aforementioned data types, of which we are aware. We have successfully used the output of DISCOVAR *de novo* (Weisenfeld et al. 2014; Love et al. 2016), ALLPATHS-LG (Gnerre et al. 2011) and long read assemblers Canu (Koren et al. 2017), FALCON and FALCON-Unzip (Chin et al. 2016), when generating chromosome-length scaffolds with our pipeline (Dudchenko et al. 2017; Matthews, Dudchenko, Kingan et al. 2017; Dudchenko et al. 2018). We have also used output from combinatorial pipelines that incorporate intermediate data types such as HiRise (Putnam et al. 2016), see (Dudchenko et al. 2018), Chromium scaffolding from 10X, BioNano Pipeline, and RefAligner (unpublished).

For our $1000 mammalian genome procedure we recommend using the w2rap contigger (B. Clavijo et al. 2017; B. J. Clavijo et al. 2017) based on DISCOVAR *de novo* (Weisenfeld et al. 2014; Love et al. 2016). The contigger can be run on hardware with less than 1Tb of RAM and typically finishes in under 48 hours for mammalian genome assemblies (with ~1Tb RAM). When run in high-memory environment such as on IBM's Large Memory Power System faster runtimes can be achieved, with the contigging step for hs-1k from (Dudchenko et al. 2018) taking less than 24 hours from 300M Illumina PE150 reads. See https://github.com/bioinfologics/w2rap-contigger for more details about w2rap.

w2rap example command

```
w2rap-contigger -t 48 -m 900 -r R1.fq,R2.fq -p w2rap --dump_all 1
```

References

Don't forget to cite the contiggers you are using in your research!

TABLE 1. Detailed statistics for several human genomes assembled using 3D-DNA and Juicebox Assembly Tools as compared to hg5 and hg38 reference genomes. The listed assemblies were created with 7X Hi-C data (same for all assemblies) and the following DNA-Seq data types: 300M Illumina PE150 reads from a short insert size library (assembled with w2rap package (B. Clavijo et al. 2017; B. J. Clavijo et al. 2017)); 372M Illumina PE250 reads from a short insert size library and assembled with DISCOVAR *de novo* (Weisenfeld et al. 2014) (note that this assembly was generated without Juicebox Assembly Tools, see (Dudchenko et al. 2017) for details); collection of short reads from libraries with varying insert sizes assembled with ALLPATHS-LG (Gnerre et al. 2011); Oxford Nanopore data polished with Illumina and assembled with Canu assembler from (Jain et al. 2017); Pacific Biosciences data assembled with FALCON assembler and shared by the McDonnell Genome Institute (NCBI accession number: GCA_002077035.2). Statistics listed include: (1) the percentage of 1kb sequences that are placed in chromosome-length scaffolds and corresponds to the "correct" chromosome in hg38 (identified by whole-genome alignment); (2) the percentage of randomly selected pairs of 1kb sequences assigned to the same chromosome-length scaffold in the assembly that are ordered in agreement with hg38; (3) the percentage of consecutive pairs of 1kb sequences that are ordered in agreement with hg38; (4) the percentage of 1kb sequences that are oriented in agreement with hg38. Only sequences uniquely aligning to hg38 (mapq≥60) were considered in all of the analyses.

Input: / Algorithms:	$1K Illumina (DNA-Seq & Hi-C) w2rap + 3D-DNA + JBAT	$7K Illumina (DNA-Seq & Hi-C) DISCOVAR + 3D-DNA	PE/MP/Fosmid draft & Hi-C ALLPATHS-LG + 3D-DNA + JBAT	Oxford Nanopore draft & Hi-C Canu + 3D-DNA + JBAT	Pacific Biosciences draft & Hi-C Falcon + 3D-DNA + JBAT	IHGSC (hg5)	hg38 (GRCh38.p12)
draft total seq bases (>1K)	2,712,354,371	2,819,306,710	2,614,901,442	2,646,010,004	2,858,827,405	n/a	n/a
draft number of contigs	164,715	80,223	231,190	2,886	3,628	n/a	n/a
draft contig N50	32,574	102,922	23,924	3,620,647	14,520,880	n/a	n/a
draft contig NG50*	28,415	94,232	19,559	3,024,148	13,176,815	n/a	n/a
draft longest contig	333,426	768,671	145,971	27,160,256	52,422,359	n/a	n/a
draft total length of scaffolds	2,717,536,071	2,819,952,110	2,786,254,569	2,646,010,004	2,858,827,405	n/a	n/a
draft number of scaffolds	112,925	73,770	11,389	2,886	3,628	n/a	n/a
draft scaffold N50	77,994	125,775	12,084,118	3,620,647	14,520,880	n/a	n/a
draft scaffold NG50*	67,475	115,268	10,404,037	3,024,148	13,176,815	n/a	n/a
draft longest scaffold	839,367	1,261,627	48,689,161	27,160,256	52,422,359	n/a	n/a
chr-length total seq bases	2,399,853,403	2,654,127,695	2,576,009,768	2,603,945,898	2,780,087,239	2,641,174,512	2,911,225,061
chr-length number of contigs	88,735	36,616	213,278	2,441	1,200	148,041	665
chr-length contig N50	36,914	108,937	24,318	3,517,161	14,518,630	80,184	57,879,411
chr-length contig NG50*	28,200	93,928	19,509	2,935,826	14,518,630	53,330	56,413,054
chr-length longest contig	333,426	768,671	260,490	27,039,813	47,391,326	28,515,322	141,414,041
chr-length total length of scaffolds	2,423,876,603	2,669,861,395	2,735,224,778	2,605,154,898	2,780,675,739	3,258,491,231	3,031,042,417
chr-length number of scaffolds	23	23	23	23	23	23	23
chr-length scaffolds N50**	125,170,150	141,244,516	146,426,750	138,911,586	149,363,931	162,599,930	156,040,895
chr-length scaffold NG50*	114,962,098	136,507,704	140,094,183	133,352,303	141,190,470	166,623,906	156,040,895
chr-length longest scaffold	207,780,125	225,222,252	229,098,500	219,596,205	232,958,391	282,193,664	248,956,422
full output total seq bases	2,712,354,371	2,819,306,710	2,614,901,442	2,646,010,004	2,858,827,405	2,692,861,938	3,095,979,984
full output number of contigs	165,597	80,725	231,751	3,749	4,668	150,750	1,502
full output contig N50	32,499	102,793	23,907	3,490,673	13,912,961	80,911	56,413,054
full output contig NG50*	28,350	93,976	19,531	2,935,826	47,391,326	57,055	56,413,054
full output longest contig	333,426	768,671	260,490	27,039,813	47,391,326	28,515,322	141,414,041
full output total length of scaffolds	2,736,505,371	2,835,055,510	2,784,512,041	2,647,366,504	2,859,645,405	3,400,374,049	3,257,347,282
full output number of scaffolds	75,863	44,065	11,071	1,036	3,032	42	595
full output scaffolds N50**	125,170,150	141,244,516	146,426,750	138,911,586	149,363,931	152,776,421	145,138,636
full output scaffold NG50*	114,962,098	136,507,704	140,094,183	133,352,303	141,190,470	166,623,906	156,040,895
full output longest scaffold	207,780,125	225,222,252	229,098,500	219,596,205	232,958,391	282,193,664	248,956,422
chr-length, % of draft assembly seq bases	88.48%	94.14%	98.51%	98.41%	97.25%	98.08%	94.03%
chr-length, % of hg38 chr spanned	99.34%	99.10%	99.33%	99.49%	99.42%	99.47%	100.00%
chr-length, % of hg38 chr-length seq bases	82.43%	91.17%	88.49%	89.45%	95.50%	90.72%	100.00%
% of IHGSC 2001 chr-length seq bases (hs5)	90.86%	100.49%	97.53%	98.59%	105.26%	100.00%	110.22%
1kb chunks, % of assigned to correct chr	99.88%	99.83%	99.84%	99.93%	99.91%	98.23%	n/a
1kb chunks, % of correctly ordered for randomly selected pairs	99.73%	99.13%	99.60%	99.83%	99.76%	96.02%	n/a
1kb chunks, % of correctly ordered for adjacent pairs	94.95%	95.73%	98.90%	99.10%	99.40%	77.55%	n/a
1kb chunks, % of correctly oriented	94.82%	95.58%	99.07%	99.06%	99.33%	76.93%	n/a

*Assumed genome size: 3031.04 Mb

**The scaffold N50 and NG50 for the output assemblies are determined almost entirely by the lengths of chromosomes

Chapter 3

Hi-C and Juicer Analysis

Hi-C is used to assemble genomes by using the 3D proximity relationships between short contigs and scaffolds (draft fragment sequences) to anchor, order and orient them. In this section we comment on Hi-C library preparation and sequencing, and analyzing Hi-C data with respect to the draft assembly to make a genome-wide record of those 3D proximity relationships.

In situ Hi-C

Although much of 3D genome assembly is conceptually compatible with other proximity-based ligation assays and other types of 1D-distance-dependent pairwise data such as mate pair, linked reads and linkage data, for best results we recommend generating an *in situ* Hi-C library.

In situ Hi-C protocol combines the original Hi-C protocol from 2009 (Lieberman-Aiden, van Berkum et al. 2009) with nuclear ligation assay (Cullen, Kladde, and Seyfred 1993), in which DNA is digested using a restriction enzyme, DNA-DNA proximity ligation is performed in intact nuclei, and the resulting ligation junctions are quantified.

The *in situ* Hi-C protocol involves crosslinking cells with formaldehyde, permeabilizing them with nuclei intact, digesting DNA with a suitable 4-cutter restriction enzyme (such as MboI), filling the 5'-overhangs while incorporating a biotinylated nucleotide, ligating the resulting blunt-end fragments, shearing the DNA, capturing the biotinylated ligation junctions with streptavidin beads, and analyzing the resulting fragments with paired-end Illumina sequencing (see FIGURE 3).

In situ Hi-C protocol has several major advantages over earlier proximity ligation assays. Crucial, *in situ* ligation reduces the frequency of spurious contacts due to random ligation in dilute solution. It also allows for more uniform coverage, enables higher resolution and more efficient cutting of chromatinized DNA through the use of a 4-cutter rather than a 6-cutter as in many earlier protocols. See (Rao, Huntley et al. 2014), Extended Experimental Procedures section (I.a.1: In situ Hi-C protocol) for a detailed description of method.

The protocol focuses on cultured cells but is easily adaptable to other types of tissues such as animal blood, animal tissue, bacteria, insects and plants.

As with many other applications quality control of sequencing libraries is very important for genome assembly. A lot of research has gone into developing useful QC metrics for genomic experiments including Hi-C (see, for example, section II.d in (Rao, Huntley et al. 2014)). It must be noted however that the majority of these metrics relies on aligning small amounts of sequencing data to a chromosome-length genome reference. The latter is of course unavailable unless performing a resequencing project.

In view of this it is advisable to monitor the critical enzymatic reactions during Hi-C library preparation by using, for example, gel electrophoresis. See also notes in Juicer Analysis of Hi-C Data section below.

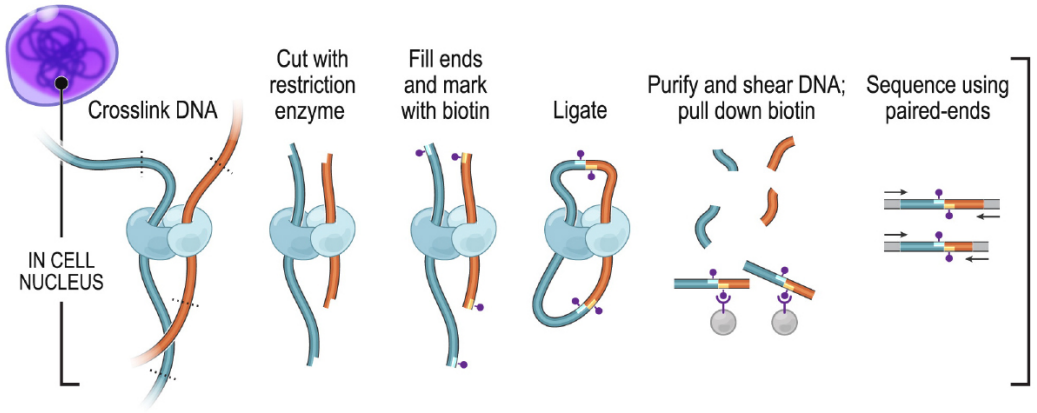

FIGURE 3. We recommend *in situ* Hi-C as a source of linking data to use during genome assembly. In the *in situ* Hi-C protocol, enzymatic reactions to capture DNA-DNA proximity are performed in intact nuclei, leading to considerably reduced frequency of spurious contacts as compared to older protocols. In brief, performing Hi-C entails the following steps: crosslinking the sample to glue DNA and proteins in place, and then cutting and religating the DNA based on physical proximity. The ligation junctions are marked with biotin, allowing for enrichment of the final library for 'interesting' chimeric sequences via biotin-streptavidin pull-down. Chimeras are then prepared for paired-end sequencing on the Illumina instrument. From (Rao, Huntley et al. 2014).

Juicer Analysis of Hi-C Data

Juicer is an open-source, one-click tool for processing large Hi-C datasets typical when studying chromatin architecture (Durand, Shamim et al. 2016). In order to meet the engineering challenge of handling large datasets, Juicer supports the use of parallelization and hardware acceleration, including CPU clusters, general-purpose graphics processing units (GP-GPUs), and field-programmable gate arrays (FPGAs). (When used for assembly purposes these are often not necessary as the datasets involved are, relatively, small and can often be run on a single machine.) Juicer is also compatible with a variety of cloud and cluster architectures. Juicer is closely based on the algorithms that were introduced in (Rao, Huntley et al. 2014).

Learn more about Juicer here: https://github.com/theaidenlab/juicer.

Juicer transforms raw sequence data into a list of Hi-C contacts (pairs of genomic positions that were adjacent to each other in 3D space during the experiment). To accomplish this, read pairs are aligned to the genome of interest, both duplicates and

near-duplicates are removed, and read pairs that align to three or more locations are set aside. The Hi-C contacts are listed in the merged_nodups.txt file (mnd file) and this is the file that serves as input for downstream analysis (running 3D-DNA and JBAT).

In order to generate the mnd file for a draft genome assembly the following steps should be performed.

1. Follow the Installation instructions to set up dependencies for alignment and creation of the Hi-C pairs merged_nodups.txt file (GNU CoreUtils and BWA). (Check also Juicer on a cluster or Juicer in the cloud sections of the Juicer wiki.)

2. Index the draft fasta sequence to prepare it for alignment. The default aligner used by Juicer is BWA (Li and Durbin 2009).

Index the draft fasta sequence using BWA

```
bwa index draft.fa
```

3. Pre-calculate the position of the enzyme restriction sites with respect to the draft genome sequence. You can use the following script from the Juicer package: generate_site_positions.py (https://github.com/theaidenlab/juicer/blob/master/misc/generate_site_positions.py).

Generate a restriction sites file for draft fasta

```
generate_site_positions.py MboI draft draft.fa
```

The latter command will create a draft_MboI.txt file that represents a sorted list of positions in the draft assembly where the fasta matches the MboI restriction enzyme recognition sequence (GATC). One line in the restriction site file corresponds to one sequence in the draft fasta.

4. Set up a working directory to contain a /fastq directory with the raw fastq files from sequencing a Hi-C library (the supported naming conventions are _*R[1,2]*.fastq and _*R[1,2]*.fastq.gz) and run Juicer.

Run juicer

```
./juicer/scripts/juicer.sh –g draft –s MboI –z draft.fa –y draft_MboI.txt –p assembly
```

(Note that some Juicer versions do not do not gracefully handle the assembly scenario. In such cases there will appear error messages reporting inter.hic and inter_30.hic files missing.) If you want to save time by skipping the statistics calculation run the juicer.sh command with a –S early flag to exit the pipeline immediately upon generation of the merged_nodups.txt file. Make sure to check the error logs in this case as the internal checks that are run at the end of the pipeline may not execute to help you access the results. (We are working to rectify this.)

5. Quality control/troubleshoot. Note that the majority of statistics developed as part of the Juicer pipeline are not robust with respect to the draft genome assembly quality and will differ dramatically when the same Hi-C library is processed against genome assemblies of varying levels of contiguity. As such they can have limited utility for quality control and troubleshooting, especially for highly fragmented genomes.

   In such cases it is often useful to pay attention to the percentage of reads containing ligation junctions in the raw fastq files of the Hi-C library. The exact number depends on the restriction enzyme, size selection protocols and the sequencing read length, but typically amounts to 20-40% for a reasonably good *in situ* Hi-C library. Low numbers may indicate poor Hi-C data quality.

   For relatively contiguous drafts standard Juicer statistics are easier to interpret and can be of help when accessing the sequencing results.

References

If you use Juicer in your analysis, please cite the following paper:

Neva C. Durand*, Muhammad S. Shamim*, Ido Machol, Suhas S. P. Rao, Miriam H. Huntley, Eric S. Lander, and Erez Lieberman Aiden. 2016. **"Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments."** Cell Systems 3 (1): 95–98. https://doi.org/10.1016/j.cels.2016.07.002. (*These authors contributed equally to this work).

Chapter 4

Candidate Assembly with 3D-DNA Pipeline

3D-DNA is a custom computational pipeline to correct misassembles, anchor, order and orient fragments of DNA based on Hi-C data. An overview of the workflow is schematically given in FIGURE 4.

We begin with a series of iterative steps whose goal is to eliminate misjoins in the input fragments. Each step begins with a scaffold pool (initially, this pool is the set of input fragments themselves); the scaffolding algorithm is used to order and orient these scaffolds; and the misjoin correction algorithm is applied to detect errors in the scaffold pool. Finally, the edited scaffold pool is used as an input for the next iteration of the misjoin correction algorithm. The ultimate effect of these iterations is to reliably detect misjoins in the input scaffolds without removing correctly assembled sequence.

After the iterations are complete, the scaffolding algorithm is applied to the revised input scaffolds, and the output – a single "megascaffold" which concatenates all the chromosomes – is retained for post-processing. This post-processing includes four step: (i) a polishing algorithm; (ii) a chromosome splitting algorithm, which is used to extract the chromosome-length scaffolds from the megascaffold; (iii) a sealing algorithm, which detects false positives in the misjoin correction process, and restores the erroneously removed sequence from the original scaffolds; and (iv) a merge algorithm, which corrects misassembly errors due to undercollapsed heterozygosity in the input scaffolds. Step (iv) is not run by default and is only required if the draft is known to contain substantial amounts of undercollapsed heterozygosity, see (Dudchenko et al. 2017; Matthews, Dudchenko, Kingan et al. 2017).

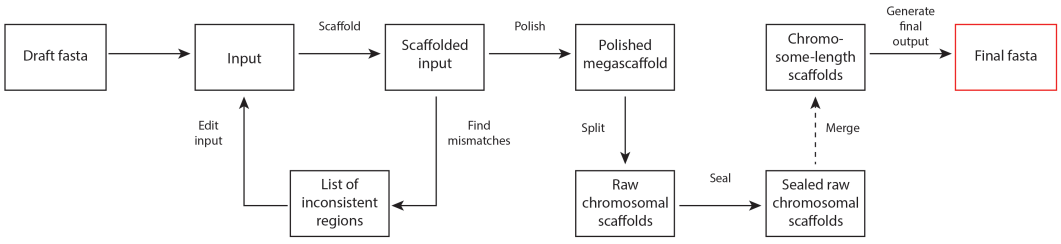

FIGURE 4. Workflow diagram for the 3D-DNA pipeline. The pipeline starts with setting aside very small scaffolds (threshold side defined by the --input option). The remaining scaffolds are ordered and oriented, and the output is used to detect and correct misjoins in

the input scaffolds. The corrected scaffolds are then again subject to ordering and orienting: the procedure (from scratch). This can be repeated several times (controlled by the --rounds option). Once the iterative scaffolding and misjoin detection are finished, the results are polished by running a coarse-grained misassembly detection and rescaffolding the resulting large pieces. The resulting megascaffold is then split into chromosomes, sealed to examine and restore false-positive edits introduced during misjoin detection and, if running in diploid mode) examined for overlaps (undergoing development, for stable version see the original 3D-DNA distribution). From (Dudchenko et al. 2017), with modifications.

For more details on the workflow of the 3D-DNA pipeline see (Dudchenko et al. 2017).

Prerequisites and Installation

- Bash >=4
- GNU Awk >=4.0.2
- GNU coreutils sort >=8.11
- Java version >=1.7
- LastZ (version 1.03.73 released 20150708) – only if planning to run the merge part of the pipeline;
- Python & Numpy for Python – only if planning to run the chromosome number-aware version of the chromosome splitting module.

We also highly recommend installing the GNU Parallel shell tool to speed up the pipeline:

- GNU Parallel >=20150322.

There is no need for installation. The pipeline consists of one main bash wrapper script (run-asm-pipeline.sh) that calls individual modules to assemble a genome. Check https://github.com/theaidenlab/3d-dna for the latest distribution.

Quick Start, Arguments and Options

As input, the pipeline requires:

1) a fasta file describing the draft assembly (*.fasta*, *.fa* and *.fna* files are allowed), and
2) a "merged_nodups.txt" file which is generated by the Juicer pipeline (see Chapter 3), and contains a duplicate-free list of paired alignments from an in situ Hi-C experiment to the draft assembly fasta file.

To run the pipeline with default parameters try the following.

Example run with default parameters

```
./run-asm-pipeline.sh draft.fa merged_nodups.txt
```

In many cases, running the 3D-DNA pipeline in combination with a review step in Juicebox Assembly Tools is sufficient to yield highly accurate assemblies. If the output is unsatisfactory, or to avoid the need for manual refinement in JBAT, it is also possible to examine the output of 3D-DNA (and/or individual 3D-DNA modules) in Juicebox to guide modification to default parameters for tuning performance to a particular draft genome.

Scenarios in which modifications of default parameters are typically required are: when working with small amounts of Hi-C data (<7X coverage) or with data generated from a low frequency restriction enzyme, when the Hi-C signal displays strong coverage biases or is noisy such as when working from in dilution Hi-C libraries or libraries from degraded samples, when assembling draft scaffolds with large gaps or badly polished long-read data, when assembling small genomes or drafts that contain large amounts of duplicated sequence due to undercollapsed heterozygocity, or when assembling from a partially diploid draft. These problems usually manifest as too much sequence being annotated as 'debris' and removed from the assembly process, and can be addressed by tuning the edit module (see some comments on this below).

Run help to see the list of parameters available for tuning pipeline performance. The most important parameters include

Options, short list: run-asm-pipeline.sh -h

-i|--input	the size threshold for the draft sequences to scaffold. Contigs/scaffolds smaller than input_size are going to be ignored by the algorithm and concatenated to the output without change. Default: 15000.
-r|--rounds	the number of iterative rounds for misjoin correction. Default: 2.
-m|--merge	the parameter to specify whether to run the merge module or not. Merging is not run by default and should only be employed if high levels of undercollapsed heterozygosity are expected to be present in the draft assembly.
-h|--help	shows this help. Use –h for main options and --help for a full set of options.

The 3D-DNA starts by putting aside the input fragments smaller than the value of parameter passed with the **–i|--input** flag. Due to their small size, these scaffolds (referred to as 'Tiny' in (Dudchenko et al. 2017)) have relatively few Hi-C contacts, making them more difficult to reliably analyze. Including them into analysis typically

results in some lowering of local ordering and orientation accuracy. Once separated, the tiny scaffolds are not processed further and are simply concatenated to the final output fasta without modification.

As such, the **–i|--input** parameter almost exclusively defines the percentage of the draft sequence that will enter into the assembly process and have a chance to be incorporated into chromosome-length scaffolds (referred to as 'Resolved' in (Dudchenko et al. 2017)). Consider lowering the default value for this parameter if wishing to increase the percentage of resolved sequence even at the expense of slightly reduced accuracy. For the accuracy estimates with default parameters see (Dudchenko et al. 2017, 2018).

An important part of 3D-DNA is misjoin correction. The misjoin correction is performed in a series of iterative steps (rounds) in which the 3D-DNA scaffolding algorithm is employed to create a preliminary genome-wide assembly and analyze it for discrepancies in the Hi-C signal (see (Dudchenko et al. 2017), Pipeline description section in Supplementary Online Material). Consider increasing the default number of rounds defined by the **–r|--round** option if dealing with a draft that contains an exceptionally large numbers of misjoins. When the draft is highly accurate however the candidate output without any error correction (-r 0) is often good enough for manual review in Juicebox Assembly Tools. Whichever the user-defined number, 3D-DNA will output files compatible with Juicebox Assembly Tools review and analysis for each of the individual rounds, so that the user can choose which one to use *a posteriori*.

An occasional error modality found in draft haploid genome assemblies is undercollapsed heterozygosity. This is when there exists a subset of the scaffolds such that each scaffold accurately corresponds to a single locus in the genome, but these loci overlap one another. Consequently, there are individual loci in the genome that are covered multiple times by different scaffolds. This error is typically caused by the presence of multiple haplotypes in the input sample material, which are sufficiently different from one another that the contiggers do not recognize them as emerging from a single locus. A typical manifestation of this error is larger-than-expected total haploid assembly size.

To address this class of misassembly error, run the pipeline with a **–m|--merge** flag. This will run the 3D-DNA algorithm for merging assembly errors due to undercollapsed heterozygosity (see (Dudchenko et al. 2017; Matthews et al. 2017)). Note that this section of the pipeline is currently under development. For stable release refer to the original 3D-DNA release.

For a more complete list of options pertaining to workflow and various individual blocks of the pipeline run the long form of help command line option.

Options, full list: run-asm-pipeline.sh - - help

**** workflow****

-s|--stage the name of any of the later stages of the pipeline to fast-forward to if necessary. Can be "polish", "split", "seal", "merge" and "finalize".

-e|--early-exit exit pipeline after "round 0" of scaffolding and map generation.

-f|--fast-start start pipeline assuming "round 0" of scaffolding is finished.

****scaffolder supplementary options****

-q|--mapq mapq threshold for scaffolding and visualization. Default: 1.

****misjoin detector supplementary options****

--editor-coarse-resolution

misjoin editor coarse matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 25000.

--editor-coarse-region

misjoin editor triangular motif region size. Default: 125000.

--editor-coarse-stringency

misjoin editor stringency parameter. Default: 55.

--editor-saturation-centil

Misjoin editor saturation parameter. Default: 5.

--editor-fine-resolution

misjoin editor fine matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 1000.

--editor-repeat-coverage

misjoin editor threshold repeat coverage. Default: 2.

****polisher supplementary options****

--polisher-input-size

polisher input size threshold. Scaffolds smaller than polisher_input_size are going to be placed into unresolved scaffolds. Default: 1000000.

--polisher-coarse-resolution

polisher coarse matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 25000.

--polisher-coarse-region

polisher triangular motif region size. Default: 3000000.

--polisher-coarse-stringency

polisher stringency parameter. Default: 55.

--polisher-saturation-centile

polisher saturation parameter. Default: 5.

--polisher-fine-resolution

polisher fine matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 1000.

****splitter supplementary options****

--splitter-input-size

splitter input size threshold. Scaffolds smaller than splitter_input_size are going to be placed into unresolved. Default: 1000000.

--splitter-coarse-resolution

splitter coarse matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 25000.

--splitter-coarse-region

splitter triangular motif region size. Default: 3000000.

--splitter-coarse-stringency

splitter stringency parameter. Default: 55.

--splitter-saturation-centile

splitter saturation parameter. Default: 5.

--splitter-fine-resolution

splitter fine matrix resolution, should be one of the following: 2500000, 1000000, 500000, 250000, 100000, 50000, 25000, 10000, 5000, 1000. Default: 1000.

****merged supplementary options****

--merger-band-size

band size for alternative haplotype search. Hardcoded: 1000000.

--merger-lastz-options

options to pass to LASTZ for alignment of draft fragments. Hardcoded: XXX.

--merger-sequence-identity

threshold sequence identity to classify fragments as alternative haplotype sequences. Hardcoded: XXX.

Note that the use of Juicebox Assembly Tools in the majority of cases alleviates the need to perform parameter sweeps on these with defaults working extremely well, requiring only very limited amount of polishing in Juicebox Assembly Tools. And even if the specific assembly project qualifies as one of the 'special' non-default cases such as those listed above, 3D-DNA provides output files that can be visually explored in Juicebox Assembly Tools to help with the parameter tuning. See below for a detailed discussion of output files.

Individual Pipeline Modules

The pipeline has a modular structure (see FIGURE 4). Wrapper script run-asm-pipeline.sh calls individual modules of the pipeline more or less in succession, starting with scaffold, visualize and edit and moving on to polish, split, seal, merge and finalize output. Code related to individual modules is organized into folders. The modules can be run as separate scripts. To skip earlier stages of the pipeline and fast-forward to later modules (when for example rerunning with different parameters) use –s|--stage flag (see Quick Start, Arguments and Options).

The list of individual modules with their core wrapper scripts is given below. Run the wrapper script with the –h flag to learn more about each individual module.

SCAFFOLD

Main script: ./scaffold/run-liger-scaffolder.sh

This module performs ordering and orientation of a given set of scaffolds, transforming a list of individual fragment sequences into a single megascaffold based on 3D proximity. The megascaffold is examined to help detect misjoins or retained for post-processing including splitting into individual chromosomes.

VISUALIZE

Main script: ./visualize/run-assembly-visualizer.sh

This module makes Juicebox Assembly Tools-compatible contact maps. The script can be used to visualize any assembly (draft or chromosome-length) and is used by 3D-DNA to generate contact maps as the pipeline moves from stage to stage to facilitate review and parameter tuning, if necessary.

EDIT

Main script: ./edit/run-misassembly-detector.sh

This module contains scripts to implement the 3D-DNA misjoin detection and correction algorithm from (Dudchenko et al. 2017). This is a module that is most likely to require parameter tuning when dealing with special use cases such as partially heterozygous drafts, older types of proximity data, noisy or biased Hi-C signal. Load the 1D-annotation files *.wig* and *.bed* output by this module to Juicebox Assembly Tools to check its performance (see also some comments on this below).

POLISH

Main script: ./polish/run-asm-polisher.sh

This module is designed to address an error modality in the basic 3D-DNA scaffolding algorithm when large-scale 3D interactions confine the signal from the diagonal – the 'core' signal associated with 1D proximity. This can sometimes happen when accumulating signal across large segments of the genome assembly, for example when working with very large initial fragments. The module attempts to address the issue by analyzing and splitting the assembly where near-diagonal signal discrepancies are detected (by running a modified version of the misassembly detection algorithm), and putting them back together based on the near-diagonal Hi-C signal. While tuning the parameters for polish is very easy using the *.wig* and *.bed* output files associated with this module, it is also true that, because such errors are rare and very obvious in Juicebox Assembly Tools, it is often easier and faster to address them in JBAT rather than in 3D-DNA.

SPLIT

Main script: ./split/run-asm-splitter.sh

This module attempts to split the megascaffold output by the previous sections of the pipeline into chromosomes. The module works by looking at large-scale discrepancies in the Hi-C signal near the diagonal (by running a modified misassembly detection script). Similar to polish, while it is easy to tune the parameters for split using the associated *.wig* and *.bed* output files, the signal associated with chromosome boundaries is, as a rule, apparent when examining the results in Juicebox Assembly Tools and, which allows to rectify any default output problems there.

Note that there is a chromosome-number-aware version of the splitter module that works by either (1) attempting to take advantage of the signal associated with chromosome territories or (2) attempting to take advantage of the telomere-to-telomere contact enrichment associated with genomes in Rabl configuration. See split_chrom_aware folder for associated scripts.

SEAL

Main script: ./seal-asm.sh

This module attempts to remove false-positive edits by reintroducing some 'debris' fragments (fragment labeled as internally inconsistent and excised from the chromosome-length portion of the sequence) back into the assembly if the pieces on either side of the debris region remain in the order and orientation consistent with the original scaffold even after the cut. This module is currently undergoing some development.

MERGE

Main script: ./merge/run-asm-merger.sh

This module is undergoing development as part of our work with the Aedes Genome Working Group, see (Matthews, Dudchenko, Kingan et al. 2017). We hope to share the updated version of the merge module soon.

FINALIZE

Main script: ./finalize/finalize-output.sh

This block generates the final fasta output from the custom formats used internally by the pipeline (*.assembly* and, deprecated, *.cprops* and *.asm*). During the final fasta generation the input scaffolds identified as part of the same chromosome-length scaffold are joined, taking into account the suggested order and orientation, while adding a gap of fixed size (500bp) in between each pair of input scaffolds. A supplementary ./finalize-output-w-stats.sh script in this module can be used to produce more comprehensive output including breakdown into various assembly components to facilitate calculating assembly statistics such as those included in (Dudchenko et al. 2017), Supplementary Online Material.

ADDITIONAL MODULES

Additional modules include:

- utils – several core scripts that are used across modules including for *.assembly* file generation from any *.fasta* file;
- lift – several core scripts to do liftover coordinates from the draft to the final assembly and vice versa;
- supp – several additional scripts including those for creating chromograms, see (Dudchenko et al. 2017, 2018). It also holds legacy scripts to match map data and generate initial conditions for AaegL4 and CpipJ3 genome assemblies from (Dudchenko et al. 2017).
- data – mapping data tables used for validation of the *Aedes aegypti* and *Culex quinquefasciatus* genome assemblies from (Dudchenko et al. 2017).

Output files

The pipeline generates a number of files as it progresses. The files, designed to be loadable in Juicebox Assembly Tools (and in many cases compatible with vanilla Juicebox

functionality) serve to provide the user with some information on the performance and give the opportunity to tune the parameters of the pipeline if necessary. Several most important file types are listed below. The main output is the fasta annotated as "FINAL" which contains the output candidate chromosome-length scaffolds.

- *.fasta* files
    - The *.fasta* is a standard text format for representing nuclear sequences. The output files in the *.fasta* format include chromosome-length sequences, labeled "FINAL". Several additional fasta files are generated by the pipeline, including all individual fragments of the draft sequences generated as part of misjoin detection and merging (if in diploid mode).
- *.hic* files
    - The .hic file is a highly compressed binary file that stores contact matrices from multiple resolutions in a clever way, allowing random access. Read more about the *.hic* format here: https://github.com/theaidenlab/juicer/wiki/Data#hic-files.
    - The output files in the *.hic* format include "final" – contact map, corresponding to the final chromosome-length output (when running in haploid mode). This is a most natural candidate for review in Juicebox Assembly Tools (to be used with the corresponding final *.assembly* file, see below).
    - "sealed", "polished", "resolved", [0123…] etc. represent contact maps corresponding to assembly pipeline stages. These can also be loaded and reviewed in Juicebox Assembly Tools, together with their corresponding *.assembly* files.
- *.assembly* (and legacy *.cprops* and *.asm* files)
    - Space-delimited text that encodes, in a concise manner, a set of instructions to be performed on draft sequences including splitting, changing their order, orientation and anchoring into chromosomes. (Previously the information was split between two other file formats, *.cprops* and *.asm*, now deprecated.) Files are generated for all stages of the pipeline.
- *.scaffold_track.txt* & *.superscaf_track.txt*
    - Stand-alone scaffold and superscaffold (chromosome) boundary files in Juicebox 2D annotation format. Files are generated for all stages of the pipeline. These can be used when limited to vanilla Juicebox functionality but not necessary when using Juicebox Assembly Tools (boundary annotations will be automatically generated based on the *.assembly* file).

- *.bed* & *.wig* files

    - 1D-track files illustrating signals used by the misjoin detector, collapsed repeat detector, polisher and chromosome slitter. Invaluable when troubleshooting these pipeline stages. The files can be loaded into Juicebox to be reviewed together with the corresponding *.hic* file. Note that the files are not yet compatible with Juicebox Assembly Tools, i.e. can be viewed only with respect to vanilla 3D-DNA output, prior to interactive revisions. See FIGURE 5 for an example of 'normal-looking' 1D tracks.

- more files:

    - *edits.for.step.*.txt*; *mismatches.at.step.*.txt*; *suspect_2D.at.step.*.txt* – list of problematic regions (Juicebox 2D annotation format).

    - *alignments.txt* (for diploid mode only) – pairwise alignment data for alternative haplotype candidates, as generated by LASTZ.

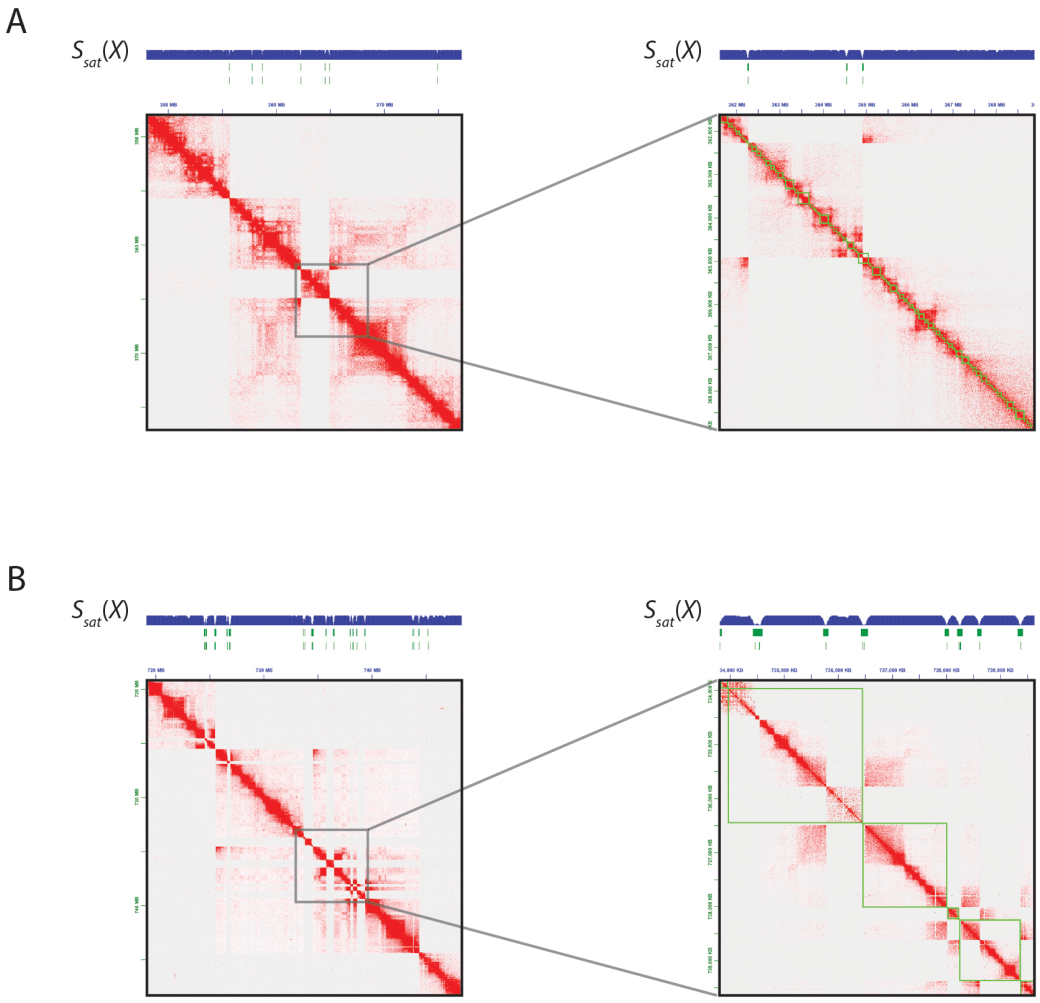

FIGURE 5. Exploring 1D tracks to analyze the output of the 3D-DNA pipeline. In this example *.wig* files illustrating the saturation signal used by the 3D-DNA pipeline to detect misjoins is loaded in blue, and the calls made by the pipeline (stored in the *.bed* file format) based on this signal are loaded in green. Both 'coarse' and 'fine' calls for misjoins are shown. Similar tracks are available for coverage anomalies, and when polishing and splitting into chromosomes. From (Dudchenko et al. 2017).

Miscellaneous

- When running a haploid version of the pipeline a good sanity check is to evaluate the total number of sequenced bases in the FINAL fasta: it should be the same as in the draft fasta. (In diploid mode this invariant does not hold.) The following script can be helpful for this: ./supp/fasta-count-sequenced-bases.sh.

- Note that at present our 3D *de novo* assembly pipeline does not properly handle ambiguous bases in the draft genome sequence. If the draft fasta you are working with contains ambiguous bases, we recommend that you convert them to Ns before running 3D-DNA (or before finalizing the fasta).

- Suboptimal Hi-C libraries can often lead to large coverage biases. A highly uneven coverage signal can mislead the default collapsed repeat detector, leading to large proportion of sequence being annotated as debris: check the *.wig* and *.bed* files associated with repeat annotation to check if suspecting this scenario and increase the default threshold repeat coverage if necessary. Similarly, suboptimal of insufficiently sequenced Hi-C libraries can fail to saturate near the diagonal, again leading to too much sequence being annotated as problematic: check the *.wig* and *.bed* files if suspecting this to be the case. Tweak the editor parameters to improve performance on your data. Note that coverage biases and insufficient saturation are sometimes observed when using long-read draft sequences (due to alignment biases) or scaffolds with very large gaps (see Intermediate Linking Data section), even when the Hi-C library appears good in QC.

References

If you use 3D-DNA in your work, please cite the following papers:

Olga Dudchenko, Sanjit S. Batra*, Arina D. Omer*, Sarah K. Nyquist, Marie Hoeger, Neva C. Durand, Muhammad S. Shamim, et al. 2017. **"De Novo Assembly of the Aedes Aegypti Genome Using Hi-C Yields Chromosome-Length Scaffolds."** Science 356 (6333): 92–95. https://doi.org/10.1126/science.aal3327. (*These authors contributed equally to this work);

and

Neva C. Durand*, James T. Robinson*, Muhammad S. Shamim, Ido Machol, Jill P. Mesirov, Eric S. Lander, and Erez Lieberman Aiden. 2016. **"Juicebox Provides a Visualization System for Hi-C Contact Maps with Unlimited Zoom."** Cell Systems 3 (1): 99–101. https://doi.org/10.1016/j.cels.2015.07.012. (*These authors contributed equally to this work).

Chapter 5

Assembly Review with Juicebox Assembly Tools

Errors in genome assemblies are, as a rule, visually obvious in Hi-C maps. When the reference is correctly assembled, the loci that are adjacent in the 1D assembly are also in close physical proximity in the Hi-C experiment, leading to the appearance of a bright band of elevated contact frequencies along the diagonal of the Hi-C heatmap. Conversely, when there are errors in the reference assembly, anomalous patterns appear in the heatmap, indicating disagreement between the 1D and the 3D signals.

3D-DNA generates assembly heatmaps as part of its workflow (see Chapter 4, Individual Pipeline Modules: visualize). The heatmaps are created by partitioning the genome assembly into loci of fixed size; and each heatmap entry indicates the frequency of contact between a pair of loci. The heatmaps provide user with information about the progress of the pipeline as well as instruct on how to tune assembly parameters if the defaults do not fit a particular task at hand.

In practice, the default parameters often work adequately for a wide range of problems, generating a chromosome-length assembly with very few remaining errors without any tuning. Thus, rather than sweeping across a wide array of parameters to identify a reliable options setting for a particular genome, it is more efficient to manually identify and remove the few remaining assembly errors. Manual review and refinement also helps to push the limits of genome assembly, for example generating reliable genomes with less input data than recommended for automatic workflow.

Assembly Tools is a new module in the Juicebox desktop application that extends the Juicebox interface for Hi-C data visualization to allow for interactive assembly refinement. When assembly errors are found, users can correct them, using a simple point-and-click interface, in a matter of seconds. Both the heatmap and the reference genome are updated in real-time to reflect these changes. Using Juicebox Assembly Tools (JBAT), users can improve genomes and reduce the cost of genome assembly. JBAT can also be used to assemble genomes manually.

For a quick start, check out this tutorial video:

https://www.youtube.com/watch?v=Nj7RhQZHM18.

Environment

The layout of the Juicebox Assembly Tools is show in FIGURE 6. The user interface consists of: the Contact map window, Menu bar, View controls, Right side and Annotation panels. The contact map window includes the name of the contact map and the Juicebox distribution. Juicebox Assembly Tools are available starting with distribution 1.8.7. Latest distribution (1.8.8) is recommended.

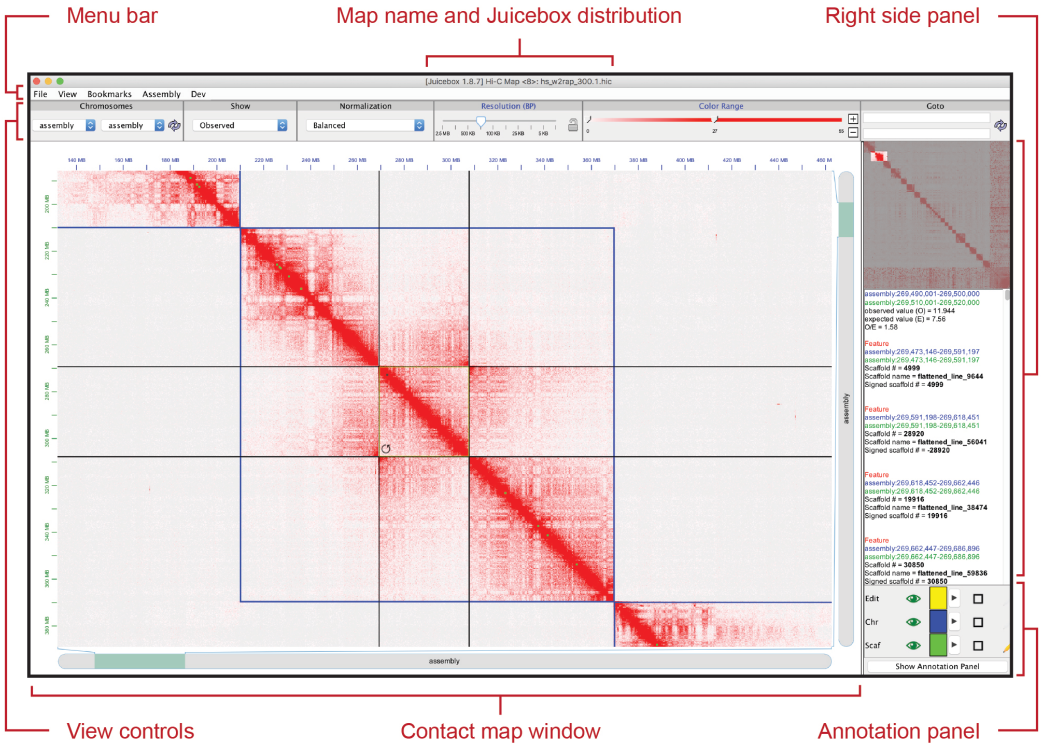

FIGURE 6. Juicebox Assembly Tools environment illustrated with hs-1k genome assembly data, see (Dudchenko et al. 2018). KR-normalized data (Normalization: Balanced) binned at 250kb resolution (Resolution slider: 250kb) is being analyzed. Only a fragment of the map is visible in the contact map window: the position of the fragment with respect to the rest of the map is highlighted on the mini map in the right side panel. Three layers of annotations associated with the assembly are visible in the Annotation panel: green (draft scaffolds), blue (output superscaffolds/chromosomes), and yellow (for holding temporary annotations). The annotations appear as squares superimposed on the contact map along the diagonal. Annotation layers are automatically populated upon loading the 'assembly' *.hic* file (Chromosome: assembly) via the *File* menu and importing the associated *.assembly* file via *Assembly* menu. In this case, the *.hic* and *.assembly* files were generated by 3D-DNA. Only a few green (scaffold) annotations are visible along the diagonal of the map: most of the individual scaffolds in hs-1k draft are too small to be visible at this zoom level. The center of the map is dominated by one blue (chromosome) annotation representing a single chromosome-scale superscaffold. A set of consecutive scaffolds is selected (encompassed by a yellow temporary annotation square with black highlight), and the detailed information associated with the selected scaffolds is displayed in the information panel under the mini map. A mouse cursor prompt for inverting the selected fragment of the assembly is shown in the lower left corner of the selection annotation square. If clicked, the action will result in changes to the contact map as well as to the underlying assembly. Changes to the assembly can be exported as a modified *.assembly* file via *Assembly* menu. The modified *.assembly* can be converted into corresponding changes in the reference *.fasta* sequence (in this case substitution of the selected sequence in the assembly for its reverse complement) using a simple command-line script (distributed as part of the 3D-DNA package). From (Dudchenko et al. 2018), with modifications.

The main JBAT components are described below. Note that the description will focus on new UI items relevant to genome assembly. Those available in older Juicebox distributions are documented here: https://github.com/theaidenlab/Juicebox/wiki.

The *File* and *Assembly* menus are most important when working in the context of Juicebox Assembly Tools.

The *File* menu is the main entry point and is used to load *.hic* files into the contact map window. Note that although we are working on making JBAT compatible will all *.hic* files at present interactive assembly is possible only with 'assembly' *.hic* files, generated while representing the reference as an abstract 'assembly' chromosome. Any contact map can be represented this way (see notes in the Quick start guide on visualizing assemblies), and 3D-DNA generates *.hic* files in compatible format by default.

The *Assembly* menu is used to import *.assembly* files (menu item "Import Map Assembly"). These files are a concise representation of the reference *.fasta* file and serve to inform JBAT of how individual sequences in the underlying reference relate to loci in the contact map. It is thus very important that the *.assembly* and *.hic* files are generated using the same reference sequence. Any reference *.fasta* can be represented as a *.assembly* file (see notes in the Quick start guide on visualizing assemblies). 3D-DNA generates *.assembly* files for each *.hic* contact map automatically.

Assembly menu is also used to save changes to the assembly (in a form of a modified *.assembly* file; menu item "Export Assembly"). By default, the exported file will have a '.review' suffix added to the original *.assembly* filename. Note that *.hic* files remain unchanged throughout the review process.

Assembly menu can also be used to load saved changes (menu item "Import Modified Assembly"). Note that when loading changes after exiting the application it is paramount to first load the original (unmodified) *.assembly* file via the "Import Map Assembly" menu item, and only then load the modified *.assembly* file. This is because, at least at present, the *.hic* file by itself does not contain any information about how the contact matrix positions relate to underlying sequences. The information needs to be added before the changes encoded in *,review.assembly* file can be meaningfully interpreted.

The *View* menu contains items for customizing the appearance of the contact map window. It is also the component to load 1D and 2D feature files (including *.bed* and *.wig*, *.bedpe* and *.txt* annotation files, coverage tracks etc. produced by 3D-DNA) to view them alongside the Hi-C map.

Additional menus are available via *Bookmarks*, where one can save the details of one's location to load reconstruct the contact map window appearance again later, and via *Dev,* which contains several menu items under development.

VIEW CONTROLS

The View controls allows one to select chromosomes (in JBAT context there is only one "assembly" pseudochromosome), switch between Observed, Observed/Expected,

Control (currently not available in JBAT), and other views, adjust normalization, change the resolution, fine-tune the color range, or go to a specific location in the contact map. When working in Juicebox Assembly Tools "Goto" control can also be used to search for map area corresponding to specific reference sequences by sequence names.

CONTACT MAP WINDOW

This is where the heat maps load. One can pan by grabbing the map with the mouse and moving or by scrolling: scrolling on top of the contact map shifts the display area along the diagonal while scrolling on top of the chromosome icons (visible at the bottom and on the right of the contact map area) shifts the display area vertically or horizontally. Double-clicking zooms in. You can also zoom in by holding down ALT key and drawing a box around a region of interest.

ANNOTATION PANEL

This is a quick access panel for 2D annotations (also accessible via *View* menu). 2D annotations play an important role in JBAT as they are used to establish correspondence between the reference sequences and the loci on the contact map.

Assembly annotations appear as squares superimposed on the contact map along the diagonal, at positions defined by the linear coordinate of the individual sequence in the assembly, and with the side equal to sequence length. (If the assembly consists of more than one chromosome 'global' assembly coordinates are calculated i.e. 1D coordinates along a particular chromosome are shifted by the total length of preceding chromosomes.)

Thus, each annotation square flanks the region on the contact map that displays the frequency of contacts of loci within the given sequence. The areas to the top center and bottom center, as well as to the left center and right center from the square shows the frequency of contacts of the loci from the given sequence with loci on other reference sequences.

When loading a *.assembly* file the annotation panel is automatically populated with three layers: Scaf (for displaying scaffold boundaries, shown in green color by default), Chr (for displaying chromosome boundaries, in blue), and Edit (for temporary annotations, in yellow). Scaffolds represent individual entries in the draft *.fasta* file used as a reference to build a Hi-C map. (Note that draft sequences very well may be contiguous, the term 'scaffolds' is used here as more generic.) Scaffolds can be grouped into chromosomes (superscaffolds). Grouping two scaffolds within the same chromosome/superscaffold (having two green squares encompassed by a single blue square on a contact map) is a signal that the scaffold sequences should be joined into a single sequence in the output *.fasta* file, when finalizing the output from the reviewed *.assembly* file. (A gap of fixed size will be added between the draft scaffolds.)

Note that to increase performance only annotations larger than one pixel size are displayed at a given zoom/resolution level. Zoom in in order to see smaller annotations.

Scaffolds are the main focus of user interaction in JBAT. Individual scaffolds can be selected using shift clicking, allowing for the following actions to be performed: cut (cut the scaffold into smaller fragments, necessary for misjoin correction), move (move the scaffold to a new position in the assembly, necessary for correcting translocations) and flip (substitute the underlying sequence for its reverse complement in the assembly, necessary for correcting inversions), see the detailed descriptions on how to perform these actions below. Translocations and inversions can be performed on selections of successive scaffolds as well as on individual scaffolds (use shift drag to select consecutive scaffolds). The selection is represented as a temporary yellow annotation square (with black highlight), encompassing the selected scaffolds (see FIGURE 6).

In addition to the three actions described above chromosome boundaries can be added and removed between any given pair of scaffolds (see below), reflected by the changes in the appearance of the annotations in the chromosome annotation layer.

Several buttons are available on the quick access annotation panel to hide/display individual layers or customize the appearance of the annotations. More customization options are available in the extended annotation layer panel (available via "Show Annotation Panel" button) as well as through several shortcut keys: use F2 to toggle the visibility of the layers, F3 to enlarge annotations (use this to make all the annotations visible, even those smaller than one pixel at a given zoom level), F4 to toggle transparency, and F6 to customize the plotting style.

RIGHT SIDE PANEL

The right side panel contains the mini map and the information panel.

The mini map serves to help the user orient oneself with respect to the genome assembly (or chromosome when using vanilla Juicebox). One can move the transparent square in the mini map to quickly move to the corresponding position in the main contact map window.

When the mouse moves over the heat map, the text in the pane is updated with information about the specific Hi-C pixel. When features such as 2D annotations are superimposed on top of the map, detailed information about them appears here when the mouse hovers over the feature. In the context of assembly tools this information includes scaffold name and id (number assigned to the scaffold in the *.assembly* format), scaffold orientation (in the form of the 'signed' id as represented by the *.assembly* file, with positive id reflecting that the sequence should be incorporated into the final assembly in the same orientation as it appears in the draft *.fasta* file, while the negative id calls for a reverse complement), as well as current 1D coordinates of the sequence corresponding to the annotation. Chromosome (superscaffold) ids and coordinates are also shown. When there

is an active selection the information panel is frozen displaying the data relevant to the selected scaffolds (see FIGURE 6).

Modifying Assembly in JBAT

Juicebox Assembly Tools allow for visual identification and interactive correction of errors in genome assemblies. The corrections are stored in a form of a *.assembly* file, a concise custom text format (used also by the 3D-DNA pipeline) that, when applied to the original reference *.fasta* by means of a simple command-line script, produces a modified reference *.fasta* reflecting the changes introduced by the user. In this section we describe four common types of misassemblies and the corresponding actions available in JBAT to address them. The user interface for this relies on a system of custom situational pointers.

The misassemblies in this section are illustrated using an example GM12878 Hi-C data set (100M reads from HIC001 library shared in (Rao, Huntley et al. 2014)). This human Hi-C data is aligned to a simulated genome assembly created by deliberately introducing errors into the sequence of two chromosome-length scaffolds from hg19 (chromosomes 2 and 4). The position of the loci according to hg19 is shown using chromograms (rainbow tracks on top in FIGURE 8, FIGURE 9, FIGURE 10, and FIGURE 11). (For the purpose of illustration, gaps have been removed from hg19 sequence.) The original pseudo assembly consists of 4 scaffolds grouped into 3 chromosomes.

MISJOINS

A 'misjoin' error in JBAT is defined as a misjoin in the draft input scaffold or contig, i.e. when regions that are not in close 1D proximity along the chromosome (or even belong to different chromosomes), are spliced together as part of the same input sequence.

Misjoins typically manifest as breaks in the bright band of elevated contact frequency along the diagonal inside a green square annotation (see FIGURE 8, left), reflecting the lack of physical proximity between sequences implied to be nearby by the draft.

The errors are addressed in a manner similar to that employed by 3D-DNA (see FIGURE 7): by excising a small region around the breakpoint, effectively cutting the original scaffold into three pieces. The procedure results in two internally consistent fragments of the original draft scaffold. The excised fragment is labeled as 'debris' and moved to the very end of the assembly. Debris fragments cannot be subject to more cutting.

In order to fix the misjoin, select the problematic scaffold by shift clicking on it (a black-and-yellow highlight will appear around the selected scaffold). Then, move the mouse towards the diagonal of the heatmap: a scissor-shaped situational pointer will appear as an invitation to 'cut' the scaffold (see FIGURE 8, middle). An accompanying dotted yellow edit annotation will show the region that will be excised out of the sequence once the scissor cursor is clicked. Use scroll to increase or decrease the size of the region flanking the tentative breakpoint. Note that the 'cut' cursor prompt appears only when the selection constitutes a single scaffold.

When happy with the tentative cut click the mouse: this will result in the appearance of two new scaffolds in place of the original one (and correspondingly two new green scaffold annotations along the diagonal in place of one), see FIGURE 8, right. The small 'debris' scaffold is moved to the very end of the assembly and kept there for future reference.

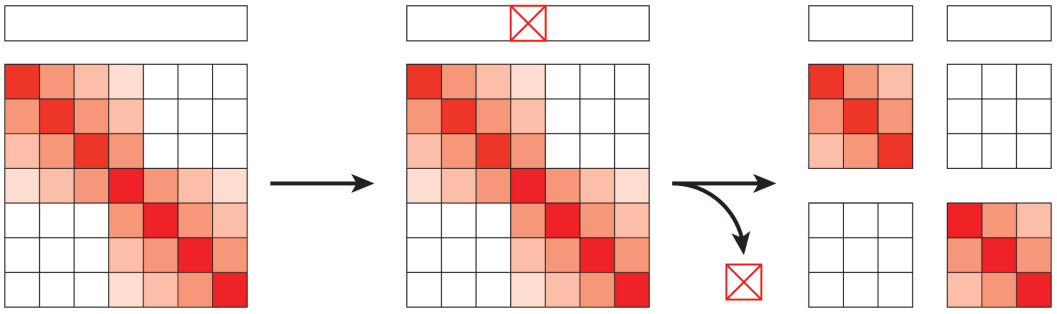

FIGURE 7. Misjoin correction, conceptual scheme. A problematic region that lies inside an input scaffold (a bin marked with an X), gets excised resulting in two internally consistent fragments of the original input scaffold (labeled as 'fragments'). The third fragment that spans a misassembled region is labeled as 'debris'. The debirs fragments are moved to the very end of the genome assembly where they are kept for future reference. From (Dudchenko et al. 2017).

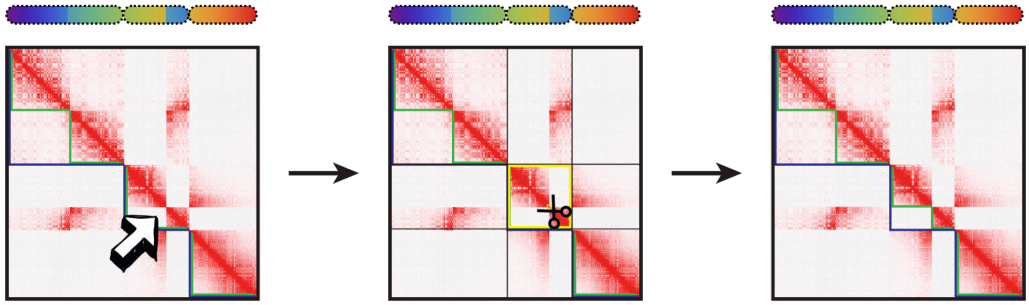

FIGURE 8. Misjoin correction example. Left panel shows a typical anomalous Hi-C signal associated with a misjoin: a draft scaffold containing a point along the diagonal such that the signal to the lower-left and the upper -right from the point is extremely depleted, reflecting the lack of physical proximity between sequences upstream and downstream of the point. In order to correct for the error select the problematic scaffold, move the mouse towards the diagonal and position the appearing situational pointer in such a way that the dotted 'debris' annotation is encompassing the breakpoint, see middle panel. (One can use scroll to adjust the size of the region flanking the breakpoint.) Once the mouse is clicked, the small debris sequence is excised and moved to the very end of the assembly, leaving behind two internally consistent scaffolds in place of the original one, available for downstream user interaction (right panel). The assembly at this point consists of 5 scaffolds joined into 3 chromosomes and a very small 6th debris scaffold. From (Dudchenko et al. 2018), with modifications.

TRANSLOCATIONS

A common type of error in genome assembly is a translocation. In the context of JBAT to fix a translocation is to change the order of draft scaffolds in the reference genome assembly. Translocation error typically manifests as a vertical and a symmetric horizontal bowtie-ish enrichment motif away from the diagonal, indicating that two sequences not in close 1D proximity along the reference genome assembly exhibit unexpectedly high levels

of Hi-C contact. The points of highest enrichment within the bowtie motif point to sequences that need to be 'brought together' (see FIGURE 9, left).

To reorder the scaffolds select the sequence that one wants to translocate, and move the mouse towards the new desired position in between two consecutive draft scaffolds. This will result in appearance of an arrow situational pointer (see FIGURE 9, middle). (Note that the prompt will not appear if the two scaffolds are not consecutive in the assembly and hence the position of the suggested insertion cannot be unambiguously interpreted from the mouse prompt. This can lead to seemingly unresponsive behavior when trying to insert between two scaffolds that have small elements in between them. In such cases zoom in in order to unambiguously indicate the new position for the selected scaffold.)

Once the pointer is clicked, the scaffold is moved into a new position (indicated by the arrow) in the reference assembly and the contact map is recalculated accordingly (see FIGURE 9, right). Note that translocation can be performed on a selection consisting of multiple scaffolds.

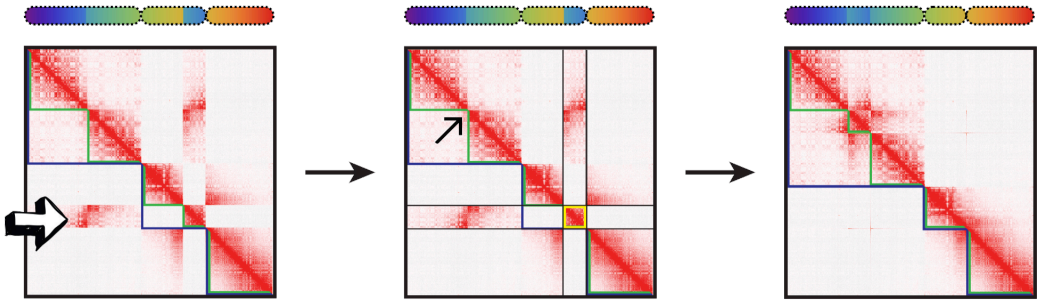

FIGURE 9. Translocation correction example. The left panel shows a typical anomalous Hi-C signal associated with a translocation: a bowtie-ish enrichment motif away from the diagonal indicating that loci not in close 1D proximity according to the reference assembly display unexpectedly high frequency of contact. The middle panel shows a situational pointer indicating a possible correction move: once clicked, the move will result in a relocation of the selected sequence (indicated by the yellow and black highlight) in between two scaffolds as indicated by the arrow. The right panel shows the result of the move, with both the reference and the heatmap updated to reflect the change. The anomalous off-diagonal motif is now gone. From (Dudchenko et al. 2018), with modifications.

INVERSIONS

Another typical error in genome assemblies is an inversion error. In JBAT, to fix an inversion is to change the orientation of the sequence represented by a single scaffold or a group of consecutive scaffolds. This error typically manifests as a bowtie enrichment motif, in parallel to the diagonal. The midpoint of the motif defines the region of the genome assembly that needs to be inverted (see FIGURE 10, left).

In order to fix the inversion, select the scaffold (or scaffolds) encompassing the region in question and move the mouse to either the bottom-left or the top-right corner of the selection. A situational pointer for flipping the sequence will appear as in FIGURE 10, middle.

Once the mouse is clicked, the sequence of the selected scaffold in the reference genome assembly will be substituted for its reverse complement. The contact map is updated in real time to reflect the change (see FIGURE 10, right).

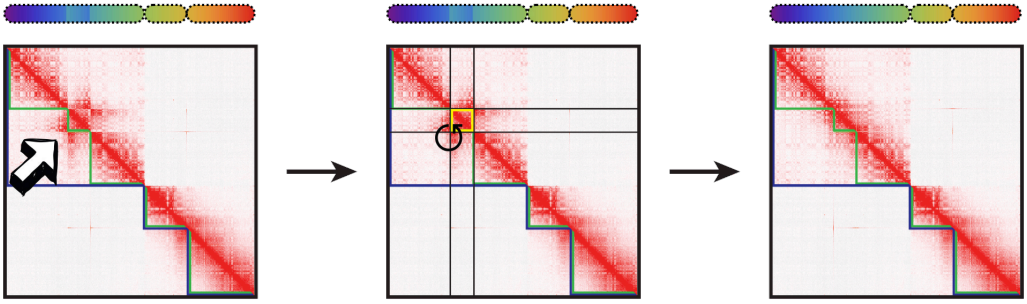

FIGURE 10. Inversion correction example. The arrow in the left panel highlights the anomalous contact motif characteristic of the inversion: a bowtie enrichment motif, parallel to the diagonal, indicating that the sequence at the beginning of the region flanked by the motif is in close proximity with the scaffold immediately downstream from the region and, vice versa, the sequence at the very end of the flanked region is in close proximity with the scaffold immediately upstream. In order to correct for the inversion, select the scaffold or scaffolds flanked by the motif and move the mouse towards either the bottom-left or the top-right corner. A prompt will appear as in the middle panel for inverting the selected sequence. Once clicked, instructions for reverse-complementing the relevant sequence in the reference genome assembly are cached, and the map is updated to reflect the change to the reference (right panel). The anomalous bowtie motif is no more. From (Dudchenko et al. 2018), with modifications.

CHROMOSOME BOUNDARIES

Adjusting chromosome boundaries is another type of assembly polishing that can be addressed via Juicebox Assembly Tools. In JBAT, chromosome boundaries can be added or removed between scaffolds, indicating whether the latter should appear as part of the same output sequence or not.

Consider the example from FIGURE 11. In this example, though the contact map appears to consist of two large enriched squares representing two chromosome territories, the assembly is split into three chromosomes (three blue annotations). In order to remove the extra boundary (indicated by the arrow in the left panel), move the mouse cursor in between the scaffolds that flank the chromosome breakpoint (no scaffolds should be selected for this): a situational pointer for toggling a chromosome boundary will appear as in FIGURE 11, middle panel. Upon click, scaffolds in chromosome groups immediately upstream and downstream from the boundary will be joined into a single group, reflected by the changes in blue chromosome layer annotations (see FIGURE 11, right panel). The new assembly now consists of two chromosome-length scaffolds. Note that chromosome boundaries can be added and removed between any two scaffolds, and the situational prompt for both actions is the same.

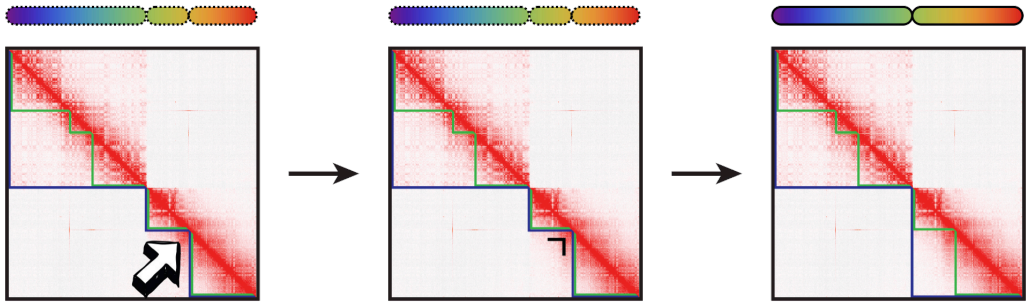

FIGURE 11. Adjusting chromosome boundaries with JBAT. In this case, the tentative assembly (left panel) consists of three chromosomes while the Hi-C signal suggests two chromosomes. The anomalous signal is in some sense opposite to the one observed for misjoin: sequences that belong to two different chromosomes according to the assembly and are hence not in close 1D proximity nonetheless exhibit strong interaction according to the Hi-C signal, indicated by a strong, unbroken diagonal (see arrow in the left panel and compare to the signal associated with the 'true' chromosome boundary upstream from the pointer). The extra chromosome boundary should as such be removed. In order to perform the action move the mouse pointer towards the diagonal, in between the two scaffolds flanking the breakpoint: a situational prompt will appear as shown in the middle panel. On click the chromosome boundary is removed, reflected by the changes in the blue chromosome layer annotations as in the right panel.

Right-click Menu, Undo and Redo

The context menu in JBAT mirrors that in vanilla Juicebox except for three additional menu items: "Move to debris", "Undo" and "Redo."

The "Move to debris" menu item serves to move a selection of scaffolds to the very end of the assembly. This may be useful when dealing with sparse or ambiguous signal, and when the requirements for accuracy overweigh the loss of information associated with the removal of a sequence from a plausible location in the chromosome-length portion of the assembly.

"Undo" menu item work to reverse the last assembly modification, and the "Redo" reverses the last Undo. The actions are available also as hotkeys: Command-U (Ctrl-U on Windows) and Command-R (Ctrl-R on Windows) for Undo and Redo, respectively.

Generating Fasta after JBAT Review

Use *Assembly* menu item "Export Assembly" to save the changes introduced during JBAT review. The changes are saved in the form of a modified *.assembly* file. By default, incorporating a 'review' suffix to the exported file name will be suggested in order to avoid overwriting the original *.assembly* file. (Remember that you may need the original *.assembly* in order to resume work on the assembly or to quickly visualize changes to the Hi-C map introduced by the review without rebuilding the *.hic* file from scratch, see notes on this in the "main menu" section in the "Environment" paragraph.)

The easiest way to convert the reviewed *.assembly* file into a fasta is to use the command-line tools distributed as part of the 3D-DNA package. To finalize the results run the following script:

Example run with default parameters

```
./run-asm-pipeline-post-review.sh –r draft.review.assembly draft.fa \
                merged_nodups.txt
```

The script will not only generate the final fasta (labeled with a "FINAL" suffix), but also rebuild the final *.hic* map for quality control. Check help for available options to customize output.

Miscellaneous

- *Assembly* menu item "Reset Assembly": use this to reverse all changes to the assembly

- *Assembly* menu item "Set Scale": a legacy option that allows one to rescale the Hi-C maps produced by 3D-DNA. (Due to data type limitations, the maps are scaled for large genomes.) After loading the assembly file the scale is set automatically.

- Move scaffold to the beginning of the assembly: we are working to add this convenience function to JBAT. For now this can only be achieved in two steps.

- Consider removing small individual scaffolds from your draft or preliminary assembly or 'bundling' small input scaffolds when reviewing very large *.assembly* files (containing hundreds of thousands of scaffolds) if having trouble with responsiveness. The idea behind the procedure is to either remove or group the scaffolds that are too small to reliably interact with into a temporary group to avoid the need of keeping track of them individually. The command-line scripts for bundling and unbundling are currently distributed as part of the 3D-DNA pipeline (see Utilities section). At some point we will make bundling/unbundling interactive and part of JBAT.

- Legacy *.cprops* and *.asm* formats are supported, though not encouraged. Select both files instead of a single *.assembly* file when loading assembly through the "Import Map Assembly" in the *Assembly* menu. Converting *.cprops* and *.asm* into the *.assembly* file format using 3D-DNA command-line tools (see Utilities section) and importing the result leads is equivalent to loading two legacy files.

- Windows machines may introduce carriage return characters that can cause problems when finalizing fasta from *.assembly* file. We will address this in the next JBAT release. In the meantime, if working on the Windows machine make sure to get rid of carriage returns in favor of a line feed character before running run-asm-pipeline-post-review.sh.

References

If you use Juicebox Assembly Tools in your work, please cite the following papers:

Neva C. Durand*, James T. Robinson*, Muhammad S. Shamim, Ido Machol, Jill P. Mesirov, Eric S. Lander, and Erez Lieberman Aiden. 2016. **"Juicebox Provides a Visualization System for Hi-C Contact Maps with Unlimited Zoom."** Cell Systems 3 (1): 99–101. https://doi.org/10.1016/j.cels.2015.07.012. (*These authors contributed equally to this work.)

and

Olga Dudchenko, Muhammad S. Shamim*, Sanjit Batra*, Neva C. Durand, Nathaniel T. Musial, Ragib Mostofa, Melanie Pham, et al. 2018. **"The Juicebox Assembly Tools Module Facilitates de Novo Assembly of Mammalian Genomes with Chromosome-Length Scaffolds for under $1000."** bioRxiv, January, 254797. https://doi.org/10.1101/254797. (*These authors contributed equally to this work.)

Chapter 6

Frequently Asked Questions

Coming soon. For now please check our online forum at http://www.aidenlab.org/forum.html. It is possible that we have already answered your question there!