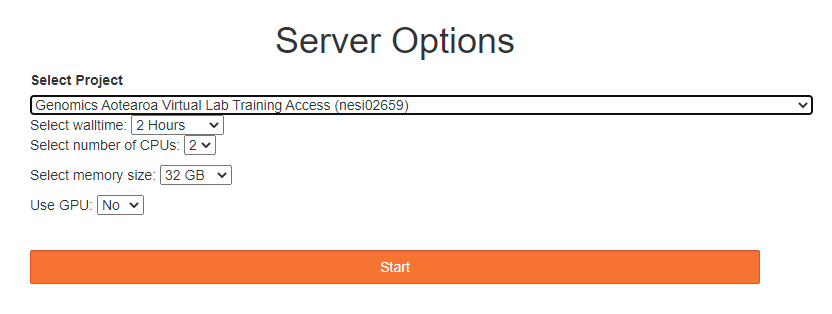
Hi-c scaffolding workflow

This document describes the workflows that we used for scaffolding rewarewa and stick insect genome assemblies from HiC data at Landcare Research (Auckland). We’ll go through the steps of 1) HiC data QC, 2) scaffolding using Salsa2 and ALLHIC, 3) visualization and correction using Juicebox tools.

# Logging into Juypter

Click the link  <https://jupyter.nesi.org.nz/hub/login> and choose the options shown below:



# Setting up

The programs/scripts and datasets we’ll be using in this session are installed in the directory under the nesi02659 project. Let’s export the directory to make a short cut for them:

export software=/nesi/project/nesi02659/Hi-C/SW/

export data=/scale\_wlg\_persistent/filesets/project/nesi02659/Hic\_practical/

Now, make your own directory, so that you wouldn’t change the source files.

cd path/to/your/own/working/directory/

# Sample data description and QC

For this workshop, we will use the published corn leaf aphid Hi-C datasets to scaffold the genome contigs generated from PacBio long reads (downloaded from: [https://academic.oup.com/gigascience/article/8/4/giz033/5429686)](https://academic.oup.com/gigascience/article/8/4/giz033/5429686)t). These datasets can be found from the exported data path. The statistics of contigs is shown below:

|  |  |
| --- | --- |
| Total genome size | 326 Mb |
| No. of contigs | 689 |
| N50 | 9 Mb |
| No. of chromosomes | 4 |

## 3.1 Hic\_qc (https://github.com/phasegenomics/hic\_qc)

Check data sequencing quality using fastqc:

ml load FastQC/0.11.9

fastqc SRR7989292\_1.fastq

fastqc SRR7989292\_2.fastq

FastQC report shows data is clean with high quality. However, some filtering/trimming tools such as BBMap and trimmomatic should be used to clean the Hi-C data.

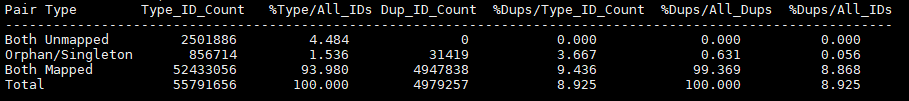
Now, we need to check if the raw hic reads are efficient for scaffolding the contigs. Firstly, we need to map the Hi-C reads to the contigs using BWA (https://phasegenomics.github.io/2019/09/19/hic-alignment-and-qc.html). We are going to mark the duplicate with samblaster and samtools to convert sam to bam files as well. SAM files shows us where the reads are mapped and their quality score on the reference assembly, and BAM is the compact version of SAM file. Load the modules and map the raw reads by typing:

ml load BWA/0.7.17-gimkl-2017a

ml load samblaster/0.1.26-GCC-9.2.0

ml load SAMtools/1.3.1-gimkl-2017a

bwa mem -5SP contig.fasta SRR7989292\_1.fastq SRR7989292\_2.fastq | samblaster | samtools view -S -h -b -F 2316 hic.bam

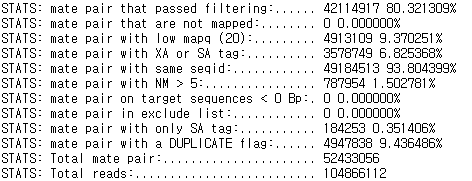


hic\_qc.py is written in Python 3. So, we need to load the appropriate module and run hic\_qc.py on the mapped reads. But before, we will filter out any reads that are not properly paired, have low mapping quality or unmapped reads using the bamfilt in matlock:

ml load Python/3.8.2-gimkl-2020a

ml load GSL/2.4-GCC-7.4.0

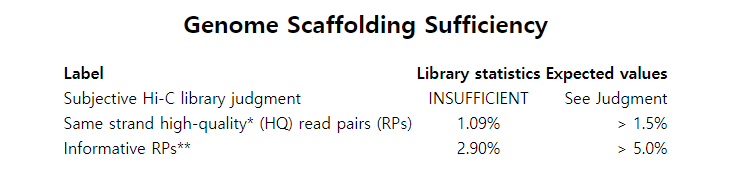
$software/matlock/bin/matlock bamfilt -i hic.bam -o filtered\_hic.bam



pip install --upgrade scipy

python $software/hic\_qc/hic\_qc.py -b filtered\_hic.bam -r -o hic\_qc -n 42114971

The result .html file tells you if your hic library is efficient for scaffolding:



At the bottom of the reports, it explains what the informative reads are and their meanings. For our cases, we have insufficient dataset. This means the HiC reads might not be sufficient for scaffolding. However, this doesn’t mean that it would not improve the assembly. Thus, let us continue with the scaffolding step.

# Hi-C scaffolding

## 4.1 SALSA (https://github.com/marbl/SALSA)

For scaffolding human genome using 40x coverage data, SALSA requires 11.43 CPU hours with 21.43 GB for peak memory usage. CPU hours and memory can vary depending on number of CPUs and the quality of the assembly.

Now we will use SALSA to scaffold the contigs based on the filtered mapping file (filtered\_hic.bam). SALSA requires bed file rather than BAM file, thus we need to convert the bam file to bed file. BED file only shows the region of reads that are mapped to without any quality information. To do so, we will use BEDtools to convert it and run SALSA. SALSA also requires the name of the restriction enzyme used for library preparation using the -e parameter. However, we can give the restriction enzyme sites, GATC, rather than a name. This is easier as it gives no restriction on the list of predefined restriction enzyme list and you can use any RE library preparation. If you used multiple RE, you could insert two ligation sites separated by a comma. As a default, SALSA goes through three iterations of run as a default. If you want more or less iterations, you can also specify using -i parameter. You can also correct mis-joint scaffolds after each iteration based on Hi-C data by using the option of “-m”. Keep in mind that this would take longer time to run. SALSA is written in Python 2, thus we need to load BEDtools and Python2 to run SALSA:

ml load Python/2.7.18-gimkl-2020a

ml load BEDTools/2.29.2-GCC-9.2.0

bamToBed -i filtered\_hic.bam hic.bed

python $software/SALSA/run\_pipeline.py -b hic.bed -a contig.fasta\_nt -l contig.fasta.fai -o test\_SALSA -e GATC

## 4.2 ALLHIC (https://github.com/tangerzhang/ALLHiC)

If you know the karyotype, it would be good to try ALLHIC. We can give a cluster number (chromosome number) to ALLHIC to tell it how many ‘super-scaffolds’ are expected. ALLHIC can also scaffold auto polyploids (<https://www.nature.com/articles/s41588-018-0237-2>). However, it needs allelic information/annotation. To scaffold diploid species, following the commands below:

$software/ALLHiC/bin/ALLHiC\_partition -b filtered\_hic.bam -r contig.fasta -e GATC -k 4

$software/ALLHiC/bin/allhic extract $/data/visualise/filtered\_hic.bam $data/SALSA/scaffolds\_FINAL.fasta --RE GATC

$software/ALLHiC/bin/allhic optimize filtered\_hic.counts\_GATC.4g1.txt filtered\_hic.clm

$software/ALLHiC/bin/allhic optimize filtered\_hic.counts\_GATC.4g2.txt filtered\_hic.clm

$software/ALLHiC/bin/allhic optimize filtered\_hic.counts\_GATC.4g3.txt filtered\_hic.clm

$software/ALLHiC/bin/allhic optimize filtered\_hic.counts\_GATC.4g4.txt filtered\_hic.clm

$software/ALLHiC/bin/ALLHiC\_build contig.fasta

# Visualization and mis-joint scaffold correction

## JuiceBox (https://github.com/aidenlab/Juicebox)

To visualize the contact map, we need to map the raw hic reads to the new assembly. Follow through the mapping step from step 3.1. We will use the new resultant filtered\_bam file to create “.assembly” and “.hic” file which are the input files for juicebox.

python $software/juicebox\_scripts/juicebox\_scripts/makeAgpFromFasta.py $data/SALSA/scaffolds\_FINAL.fasta out.agp

python $software/juicebox\_scripts/juicebox\_scripts/agp2assembly.py out.agp out.assembly

$software/matlock/bin/matlock" bam2 juicer $/data/visualise/filtered\_hic.bam out.links.txt

sort -k2,2 -k6,6 out.links.txt out.sorted.links.txt

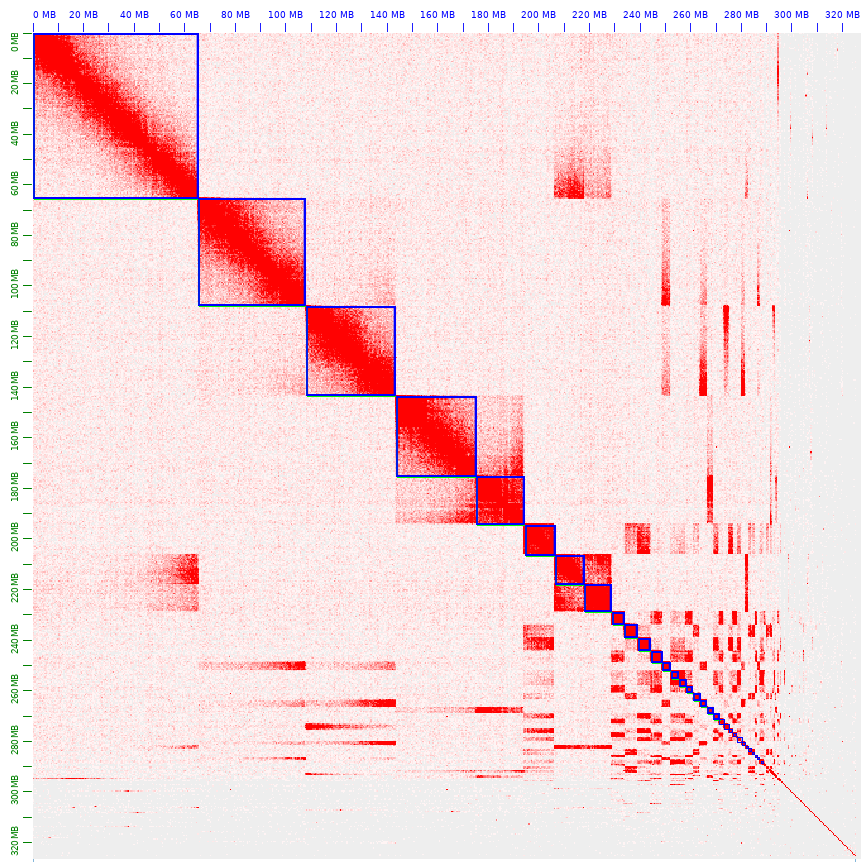
$software/3d-dna/visualize/run-assembly-visualizer.sh -p false out.assembly out.sorted.links.txt

Now we will use the out.assembly and out.hic file to visualize the contact map using the Juicebox.

java -jar $software/juicer/Juicebox\_1.11.08.exe

It is a GUI tools, so you will need to select the .hic and .assembly file manually by clicking on FileOpen for .hic file and AssemblyImport Map assembly for .assembly file. If you are running on Jupyer, juicebox does not work. Please download the software through this link: <https://github.com/aidenlab/Juicebox/wiki/Download>

When you open all the files, this is the contact map it will visualize:



This is the video I watched when I tried to curate my scaffold in juicebox manually:

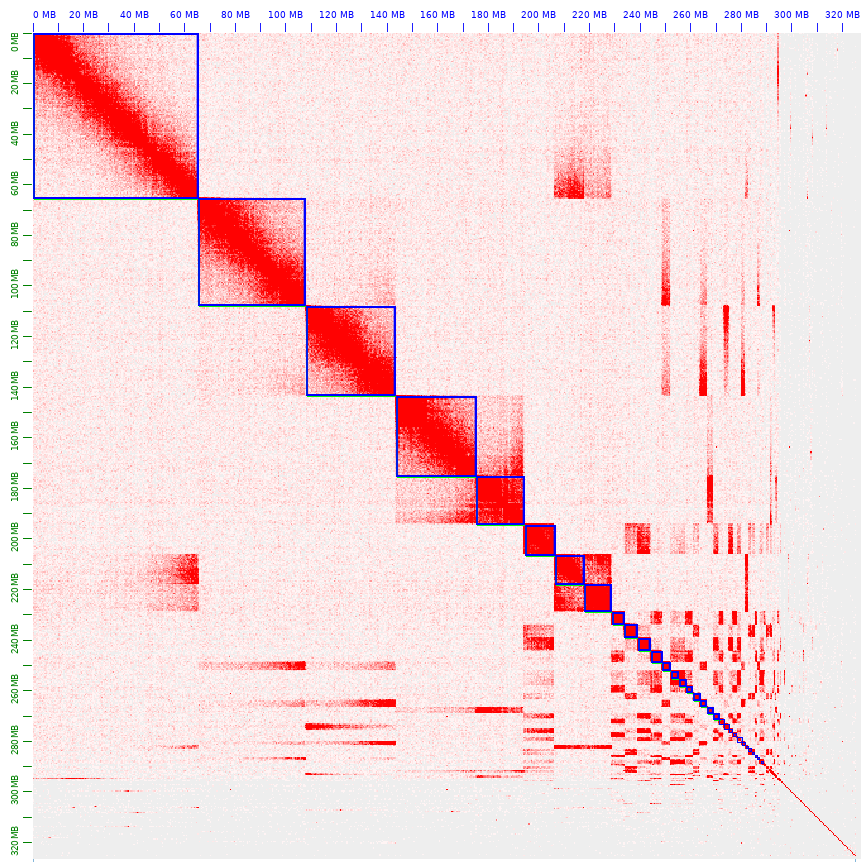
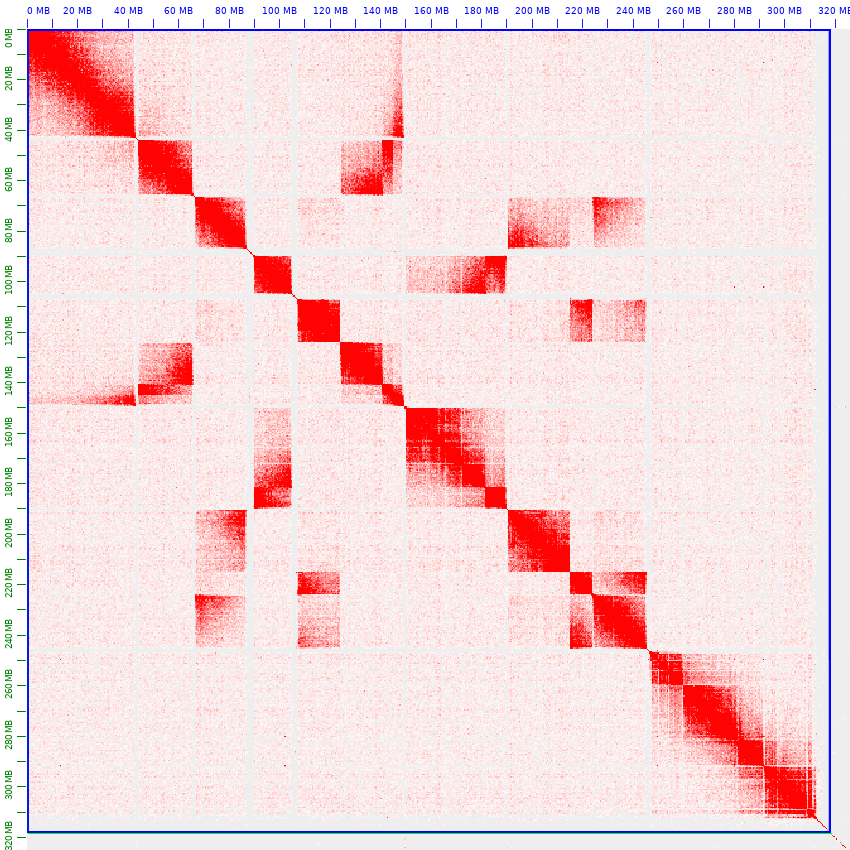
<https://www.youtube.com/watch?v=Nj7RhQZHM18>

# Comparing assemblies

Statistics

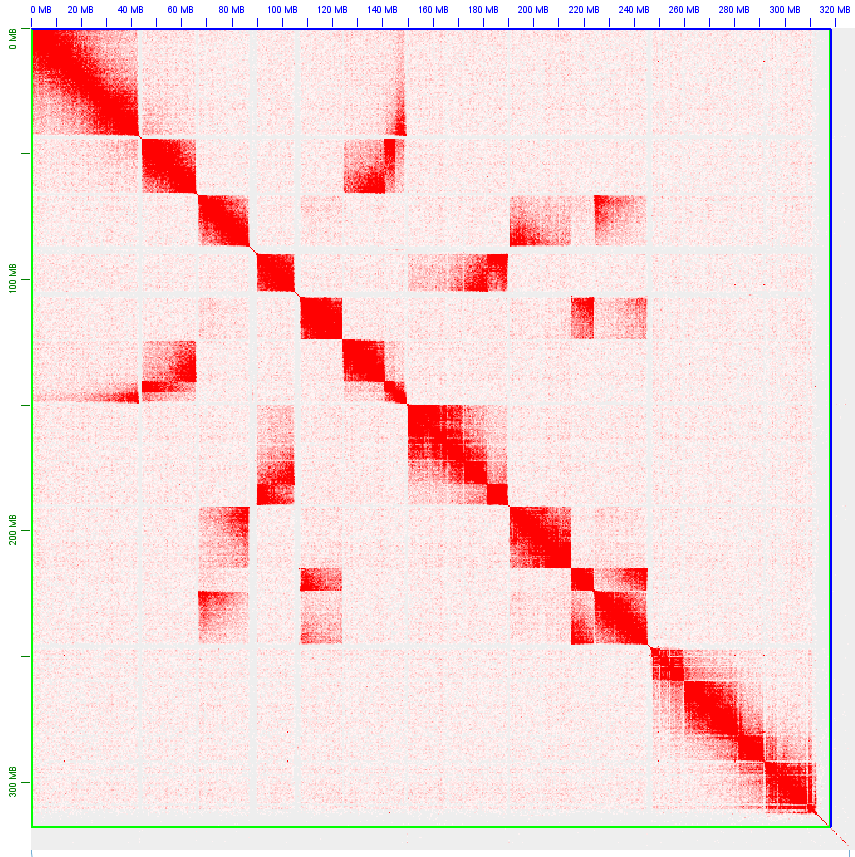
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | contigs | ALLHIC | SALSA | SALSA + ALLHIC |
| # sequences | 689 | 234 | 681 | 347 |
| Total length | 3.26E+08 | 3.26E+08 | 3.26E+08 | 3.26E+08 |
| N50 | 9046396 | 3.18E+08 | 32073405 | 3.14E+08 |
| # N's per 100 kbp | 0 | 13.96 | 4.29 | 14.54 |

HI-C contact map



ALLHIC

SALSA



SALSA + ALLHIC