TELL-Seq™ Data Analysis Software User Guide

for

Tell-Sort

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

Document # 100024 Version 1.1.2

March 2024

Table of Contents

1. Introduction

This document describes instructions on how to use TELL-Seq Data Analysis software "Tellysis" accompanied with the TELL-Seq WGS Library Prep Kit.

The TELL-Seq WGS library prep kit uses an innovative Transposase Enzyme Linked Long-read Sequencing (TELL-Seq™) technology to prepare a paired-end library to generate barcoded linked reads from an Illumina sequencing system. Linked reads can then be processed and analyzed by Tellysis for genome wide variant calling, haplotype phasing, structural variation detection, metagenomic studies, *de novo* sequencing assembly, etc.

Tellysis software comes in the form of three main pipelines:

- **Tell-Read**

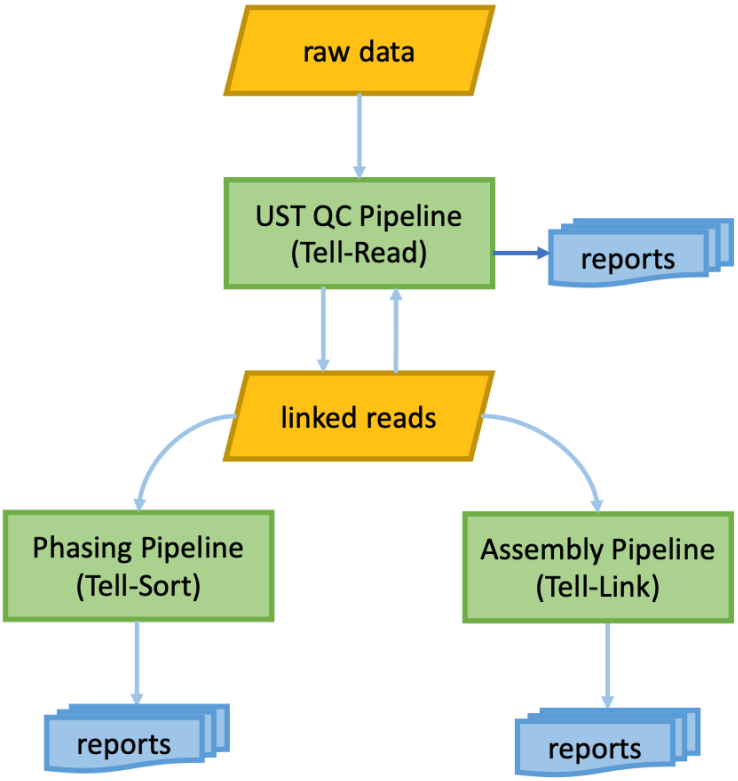
  a set of pipeline processes that takes as input the sequencing output from an NGS sequencing instrument and generates linked-read FASTQ data, as well as QC reports.

- **Tell-Sort**

  a set of pipeline processes that takes as input the linked-read data from Tell-Read result and performs variant calling, phasing.

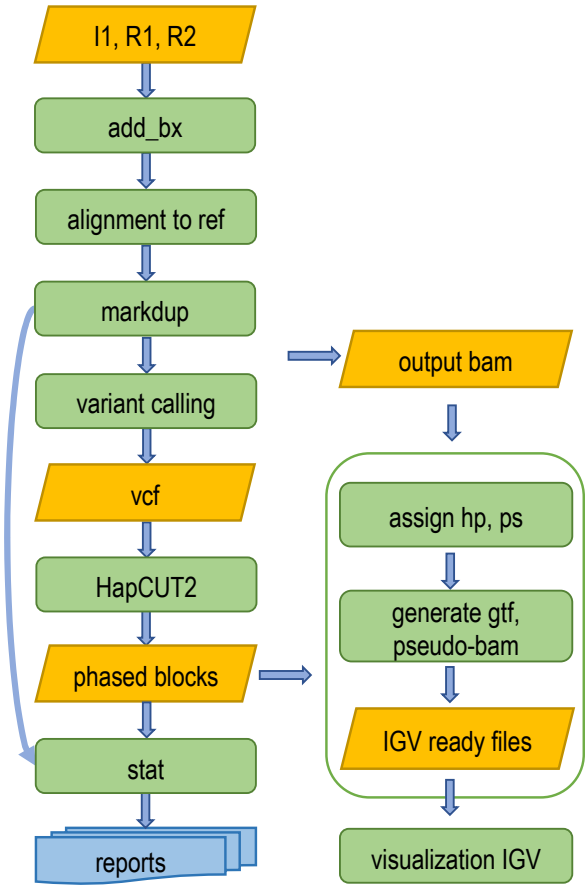
- **Tell-Link**

  de novo assembly pipeline processes that build barcode-aware assembly graph, assembles contigs and performs scaffolding.



2. Tell-Sort Pipeline

Tell-Sort pipeline processing steps can be summarized in the following diagram.



The following is a brief description of major processes in the pipeline.

Add Barcode

Barcode sequences are first added to the comment section of the read name of the FASTQ files for their associated read pairs.

Alignment to Reference

The read FASTQ files are mapped to the reference genome using `BWA-MEM`, by default, with barcode tagged as BX:Z field. The mapped BAM file is sorted by chromosome coordinates. In release V1.1.1, user can optionally use EMA alignment with command line option –a (see section Run Tell-Sort Pipeline for details).

Mark Duplicates

Duplicate reads are marked and removed with Picard and read group information are added to the BAM file.

Variant Calling

Germline variants including single nucleotide polymorphisms (SNPs) and small indels between the sample and the reference genome are called using `HaplotypeCaller` in the GATK tool package.

Phasing

Linked-reads are phased with HapCUT2 (https://github.com/ vibansal/HapCUT2) using heterozygous SNVs that involved two alleles of the same length. The BAM file of diploid variants with duplicates removed is used as input to "`extractHAIRS`" in the HapCUT2 tool to create the compact fragment file containing only haplotype-relevant information. Linked fragments are generated using the "`LinkFragments.py`" program in the package. The linked fragments and variants in VCF format are then used as input to "`HapCUT2`" for phasing.

Haplotype and Phasing Block Assignment

After phasing call, each phased read in the BAM is assigned to the proper haplotype using "HP" tag. Each phased block is assigned phased set tag "PS".

3. Installation

The Tellysis pipelines are delivered as Docker images for consistent installations and executions to minimize any potential issues arising from user environment. As such, a Docker running environment is required. For Docker engine installation instructions, user is referred to the Docker web site https://docs.docker.com/install/.

If a Docker running environment is not already available on the system, it will need to be installed. Docker is available in two editions: Community Edition (CE) and Enterprise Edition (EE). The following is an example for getting and installing Docker CE for Ubuntu/Debian systems. If a Docker running environment is already available on the system, these steps can be skipped and only the Tell-Read docker image would need to be installed.

Step 1: Update Software Repositories
As usual, it is a good idea to update the local database of software to make sure you've got access to the latest revisions.

Therefore, open a terminal window and type:

```
sudo apt-get update
```

Allow the operation to complete.

Step 2: Uninstall Old Versions of Docker
Next, it's recommended to uninstall any old Docker software before proceeding.

Use the command:

```
sudo apt-get remove docker docker-engine docker.io
```

Step 3: Install Docker
To install Docker on Ubuntu, in the terminal window enter the command:

```
sudo apt install docker.io
```

Step 4: Start and Automate Docker
The Docker service needs to be setup to run at startup. To do this, type in each command followed by enter:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Step 5: Running Docker as a non-root user
If you don't want to preface the `docker` command with `sudo`, create a Unix group called docker and add users to it:

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

Step6: Log out and log back in
After logging back in, run `Docker` as a non-root user.

After the installation of Docker or if you already have a Docker environment, follow the steps below to install the Tell-Sort docker image.

1. Download the Tell-Sort docker image package `tellsort.tar.gz`.

2. Unzip `tellsort.tar.gz`, and this will create a directory `tellsort-release` which contains the docker image of the pipeline called `docker-tellsort` and a Unix shell script called `run_tellsort.sh`.

```
$ tar xzvf tellsort.tar.gz
```

3. Load the docker image

```
$ cd tellsort-release
$ docker load -i docker-tellsort
```

4. Check image `docker-tellsort` is loaded

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-tellsort	latest	bad2a17180f9	About an hour ago	3.15GB

5. (Optional) To remove the image `docker-tellsort` to upgrade to a newer version

```
$ docker image rm -f bad2a17180f
```

4. Run Tell-Sort Pipeline

The Tell-Sort pipeline takes as input from processed fastq data resulted from Tell-Read pipeline (See *TELL-Seq Data Analysis Software User Guide for Tell-Read*).

The Tell-Sort pipeline is delivered as a docker image. The Tell-Sort package provides wrapper scripts to run the pipeline so users can avoid the docker details.

➢ Run Tell-Sort on Linked Read FASTQ Data

The wrapper script to run Tell-Sort pipeline is `run_tellsort.sh`. The command line looks like following,

```
$ run_tellsort.sh \
      -r1 /path/to/R1/data \
      -r2 /path/to/R2/data \
      -i1 /path/to/I1/data \
      -r <genome reference file in fasta> \
       -d 100000    \
       -a ema       \
      -o <path/to/output> \
      -p <prefix name> \
      -b <genome_variants>.bed \
      -v <genome_variants>.vcf.gz \
      -t number of threads
```

-r1	This required parameter specifies read 1 fastq file in gz compressed format.
-r2	This optional parameter specifies read 2 fastq file in gz compressed format.
-i1	This required parameter specifies index 1 fastq file in gz compressed format.
-r	This required parameter specifies genome reference file in fasta format.
-o	This required parameter specifies the output directory.
-p	This required parameter specifies a prefix name for identifying a result set.
-a	This optional parameter specifies an alignment algorithm to use. Currently there're two values for this option: "bwa", "ema". The default is using bwa. To select ema, use –a ema.

-d	This optional parameter specifies the maximal distance between intra-molecule reads. The default value is 50000bp.
-b	This optional parameter specifies reference genome variants bed file. This is a standard reference result that the pipeline will compare against with.
-v	This optional parameter specifies reference genome variants VCF file. This is a standard reference result that the pipeline will compare against with.
-t	This optional parameter specifies the number of threads. The default is 30.

Example 1: user supplies reference VCF via -b, -v

```
$ run_tellsort.sh \
  -r1 /runTest/Full/runTest_R1_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -r2 /runTest/Full/runTest_R2_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -i1 /runTest/Full/runTest_I1_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -r /data/genome/hg38/hg38.fasta \
  -o runTestResult \
  -p 501 \
  -b /home/ubuntu/grch38.protein_coding_genes_variants.bed \
  -v /home/ubuntu/GRCh38_GIAB_highconf.NA12878.vcf.gz
```

In this example, user uses GIAB high confidence VCF calls as reference VCF files. These reference VCF files will be used as the standard to assess the phasing performance of the pipeline run. The performance metrics will be summarized in the html report phasing_final_reports.html.

The GIAB latest NA12878 reference VCF files, similar to, *grch38.protein_coding_genes_variants.bed, GRCh38_GIAB_highconf.NA12878.vcf.gz* can be downloaded from GIAB site, https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/NA12878_HG001/latest/GRCh38.

The output directory runTestResult will be created and phasing results will be stored in the directory.

```
$ ls -al runTestResult/
drwxrwxr-x 4 ubuntu ubuntu       6144 Jul 21 01:34 ./
drwxrwxr-x 3 ubuntu ubuntu       6144 Jul 19 13:38 ../
-rw-rw-r-- 1 ubuntu ubuntu  637401457 Jul 20 23:34 501.diploid.vcf
-rw-rw-r-- 1 ubuntu ubuntu 1156487793 Jul 20 23:34 501.vcf
-rw-rw-r-- 1 ubuntu ubuntu       5612 Jul 21 01:33 501.log
drwxrwxr-x 2 ubuntu ubuntu       6144 Jul 21 01:33 501_stats/
drwxrwxr-x 2 ubuntu ubuntu      71680 Jul 21 01:33 501_temp/
```

The summary report `phasing_final_reports.html` is stored in directory 500_stats.

```
$ ls -al runTestResult/500_stats/
drwxrwxr-x 2 ubuntu ubuntu   6144 Jul 21 01:33 ./
drwxrwxr-x 4 ubuntu ubuntu   6144 Jul 21 01:34 ../
-rw-rw-r-- 1 ubuntu ubuntu   4179 Jul 20 15:33 data.js
-rw-rw-r-- 1 ubuntu ubuntu 649137 Jul 20 19:49 phasing_final_reports.html
-rw-rw-r-- 1 ubuntu ubuntu   3617 Jul 20 19:49 phasing_final_reports.txt
-rw-rw-r-- 1 ubuntu ubuntu   8203 Jul 20 14:04 phasing_results.txt
-rw-rw-r-- 1 ubuntu ubuntu   3653 Jul 20 15:31 plot.html
-rw-rw-r-- 1 ubuntu ubuntu    883 Jul 20 15:33 summary.inf
```

Example 2: run without -b, -v

```
$ run_tellsort.sh \
  -r1 runTest/Full/runTest_R1_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -r2 runTest/Full/runTest_R2_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -i1 runTest/Full/runTest_I1_A501.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz \
  -r /data/genome/hg38/hg38.fasta \
  -o runTestResult \
  -p 501 \
```

In this example, user does not provide the reference VCF files. Tell-Sort pipeline will produce phasing results without benchmarking metrics in the report phasing_final_reports.html.

➢ Prepare Genome VCF Directory (**-g** parameter)

Tell-Sort pipeline produces the phasing statistics at the chromosome level for the specie. If user provides reference VCF calls as the standard, the pipeline will measure the results against it and generate performance metrics in the final report. To this end, a VCF reference directory needs to be created to establish the reference VCF at the chromosome level, in addition to the genome level references already specified by **-b** and **-v** parameters.

The genome VCF directory contains two things: 1) a list of chromosome level reference VCF files; 2) a text file **chromosomes_{genome name}.txt** simply lists all the chromosome names for which variations will be sought and phased. The chromosome level reference VCF files are automatically generated if user provides the reference VCF files.

The genome VCF directory can be created in one of the three ways:

1. Pipeline automatically creates it. When user supplies the reference VCF files, both chromosome level reference VCF files and the text file **chromosomes_{genome name}.txt** will be generated. The chromosomes listed in the text file are taken from reference VCF files.
2. Pipeline automatically creates it. But user does not provide the reference VCF files. In this case, chromosome names will be looked up from genome fasta file.
3. User creates it. User has a chance to dictate which chromosomes the pipeline should process the VCFs for. The user prepared reference directory is provided via **-g** option.

To manually prepare VCF directory, following these steps:

1. Prepare a folder that contains all the reference chromosome VCF files and name it after the genome name. (e.g., GRCh38).
2. Create a text file and name it **chromosomes_{genome name}.txt** and include all the chromosome names in that text file. Put the text file inside of the same folder that contains the chromosome VCF files (e.g., /path/to/GRCh38/chromosomes_GRCh38.txt).

```
$ ls -al /home/ubuntu/GRCH38
total 1624188
drwxrwxr-x  2 ubuntu ubuntu      4096 Oct  3 16:05 ./
drwxrwxr-x 27 ubuntu ubuntu      4096 Sep 24 13:22 ../
-rw-rw-r--  1 ubuntu ubuntu       128 Sep 11 06:28 chromosomes_GRCh38.txt
-rw-rw-r--  1 ubuntu ubuntu  86879423 Sep 11 06:31 GRCh38_chr10.vcf
-rw-rw-r--  1 ubuntu ubuntu  87526542 Sep 11 06:31 GRCh38_chr11.vcf
-rw-rw-r--  1 ubuntu ubuntu  77999828 Sep 11 06:31 GRCh38_chr12.vcf
-rw-rw-r--  1 ubuntu ubuntu  68692074 Sep 11 06:31 GRCh38_chr13.vcf
-rw-rw-r--  1 ubuntu ubuntu  56595194 Sep 11 06:31 GRCh38_chr14.vcf
-rw-rw-r--  1 ubuntu ubuntu  48835798 Sep 11 06:31 GRCh38_chr15.vcf
-rw-rw-r--  1 ubuntu ubuntu  30458753 Sep 11 06:31 GRCh38_chr16.vcf
-rw-rw-r--  1 ubuntu ubuntu  42534812 Sep 11 06:31 GRCh38_chr17.vcf
-rw-rw-r--  1 ubuntu ubuntu  41635063 Sep 11 06:31 GRCh38_chr18.vcf
-rw-rw-r--  1 ubuntu ubuntu  32849295 Sep 11 06:31 GRCh38_chr19.vcf
-rw-rw-r--  1 ubuntu ubuntu 135561328 Sep 11 06:31 GRCh38_chr1.vcf
-rw-rw-r--  1 ubuntu ubuntu  37348542 Sep 11 06:31 GRCh38_chr20.vcf
-rw-rw-r--  1 ubuntu ubuntu  24049493 Sep 11 06:31 GRCh38_chr21.vcf
-rw-rw-r--  1  ubuntu ubuntu 19136769 Sep 11 06:31 GRCh38_chr22.vcf
-rw-rw-r--  1 ubuntu ubuntu 134410383 Sep 11 06:31 GRCh38_chr2.vcf
-rw-rw-r--  1 ubuntu ubuntu 121443463 Sep 11 06:31 GRCh38_chr3.vcf
-rw-rw-r--  1 ubuntu ubuntu 107020665 Sep 11 06:31 GRCh38_chr4.vcf
-rw-rw-r--  1 ubuntu ubuntu 102128061 Sep 11 06:31 GRCh38_chr5.vcf
-rw-rw-r--  1 ubuntu ubuntu 112250082 Sep 11 06:31 GRCh38_chr6.vcf
-rw-rw-r--  1 ubuntu ubuntu  94251504 Sep 11 06:31 GRCh38_chr7.vcf
-rw-rw-r--  1 ubuntu ubuntu  82237554 Sep 11 06:31 GRCh38_chr8.vcf
-rw-rw-r--  1 ubuntu ubuntu  74125821 Sep 11 06:31 GRCh38_chr9.vcf
-rw-rw-r--  1 ubuntu ubuntu  45124223 Sep 11 06:31 GRCh38_chrX.vcf

$ cat /home/ubuntu/GRCH38/chromosomes_GRCh38.txt
chr1
chr2
chr3
chr4
chr5
```

```
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chr20
chr21
chr22
chrX
```

Note:

Users need to make sure the chromosome names referenced in genome reference FASTA file and VCF reference file are consistent. Preprocessing may be needed to rename them for consistency. All variants in the VCF reference file must pass all filtering criteria and indicate so with a “PASS in Column 7 for each listed variant.

➢ More output files

A few output files of particular interest to users are highlighted here.

1. phasing_final_reports.html

This is the summary report for the Tell-Sort phasing pipeline. The report is located in `<prefix>_stats` subdirectory in the output directory.

2. phased.bam

This file contains Tell-Sort aligned reads with their phasing information. Compared to the unphased bam file, the phased bam file contains two customized tags: HP tag, which is for the haplotype assignment, with two potential values of haplotype “1” or “2”; and PS tag, which is for the phased

set, the PS tag value will be the phased block id. This file is in sub-directory `<prefix>_IGV_Files`.

3. **phased.vcf**

This file contains Tell-Sort phased variants and their genomic location in the reference genome. This file includes all phased and unphased heterozygous variants with the "GT" (genotype) attribute assigned with "|" or "/" depending on whether the variant is phased or remains unphased, respectively. Each variant will also have the "PS" attribute for the phase set. For the phased variants, this value will be the phased block id; for the unphased variants, the value will not be available and a "." will be assigned. This file is in sub-directory `<prefix>_IGV_Files`.

4. *.gtf

This file contains the imputed phase blocks aligned to the reference genome. The starting and ending coordinates of the block are obtained from the phase set in the phased vcf file. Phase blocks are shown as a horizontal bar on IGV spanning across the region with phased variants.

5. *pseudo_bam.bam

This file displays variants as part of haplotypes (phase blocks). This file translates the phased variants from the phased vcf file into pseudo reads (*contigs*) stored in bam format so that the user can upload them into the alignment track. The pseudo-bam file facilitates the visualization of the phased vcf file. In the pseudo-bam file, variants and genotypes are shown as vertical bars connected by a thick line and a number. The line connects variants that belong to the same haplotype (phase block). The number indicates the distance between adjacent phased positions. By default, reference alleles are shown in grey and alternate alleles are color-coded (set through Preferences, Alignments tab, and Show mismatched bases).

5. Data Visualization with IGV

This section describes how to visualize TELL-Seq data on the Integrative Genomics Viewer (IGV). IGV (https://software.broadinstitute.org/software/igv/) is an open-source interactive tool developed by the Broad Institute for the visual exploration of genomic data.

IGV version 2.4 or higher is required to view Tell-Sort data, as only these versions support to view the linked reads by parsing the bam tags of BX (barcode), MI (molecular identifier), and HP (haplotype).

IGV data files are in a sub-directory <prefix>_IGV_Files. An example is 501_IGV_Files. Under this directory, you can see following files:

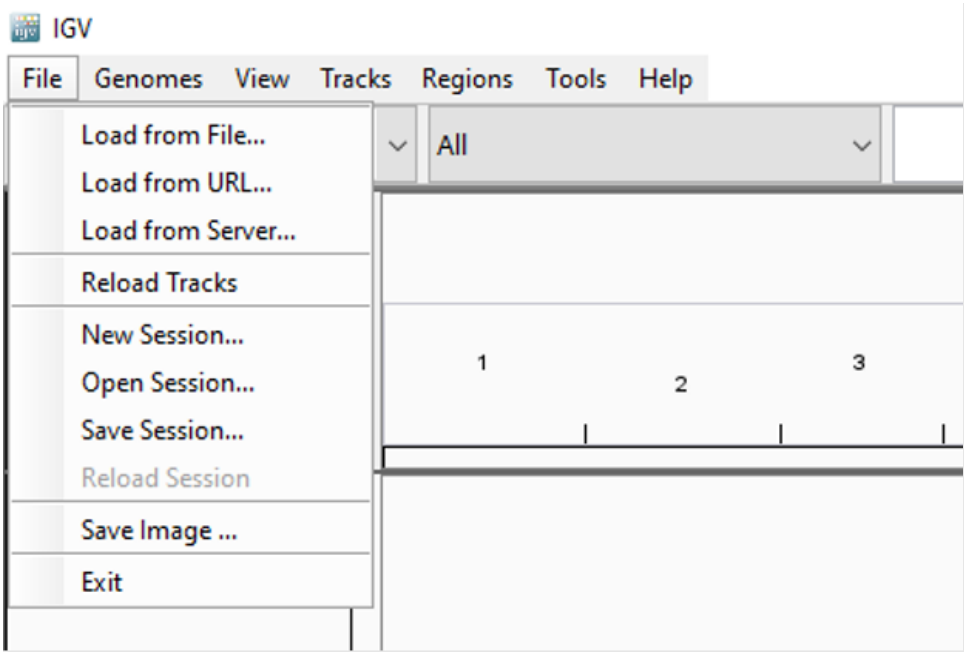
i. 501.gtf
ii. 501.phased.bam
iii. 501.phased.bam.bai
iv. 501.phased.vcf.gz
v. 501.phased.vcf.gz.tbi
vi. 501.pseudo_bam.bam
vii. 501.pseudo_bam.bam.bai

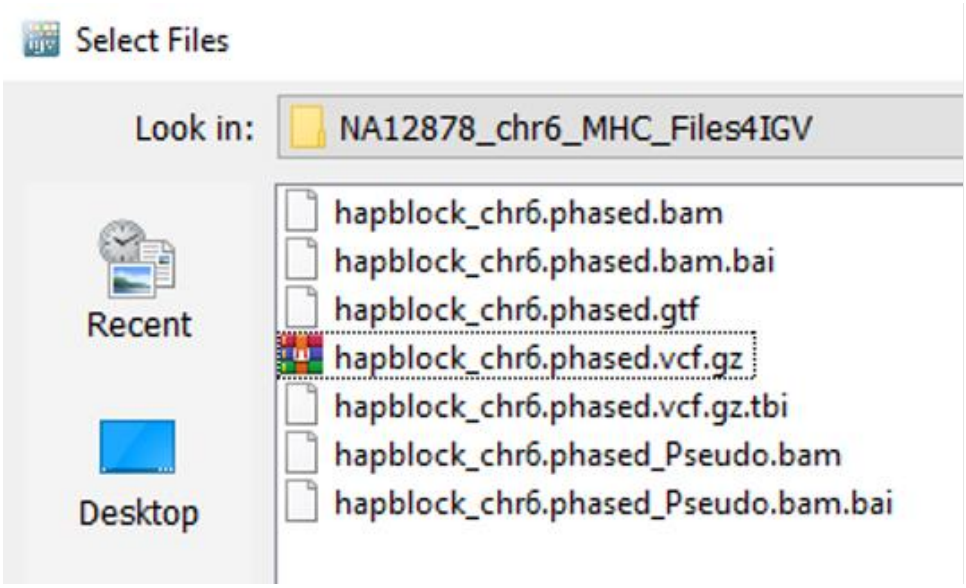
501.gtf: the gtf file for the phased block range on the reference genome.

➢ File Loading

We recommend loading the four IGV tracks described here for a comprehensive visualization of Tell-Seq data: phased vcf, phased bam, gtf, and pseudo bam (genome assembly version should be properly selected, in general, hg38). When working with multiple samples, however, visualizing all tracks at the same time might be a challenge due to the limited screen size in most settings.

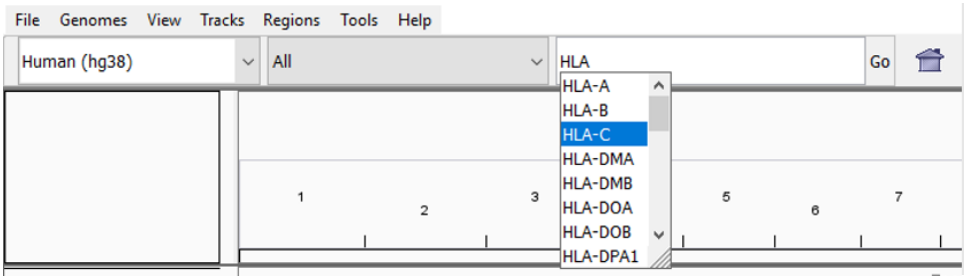
The following image represents an example of four tracks loaded from the same sample on IGV:

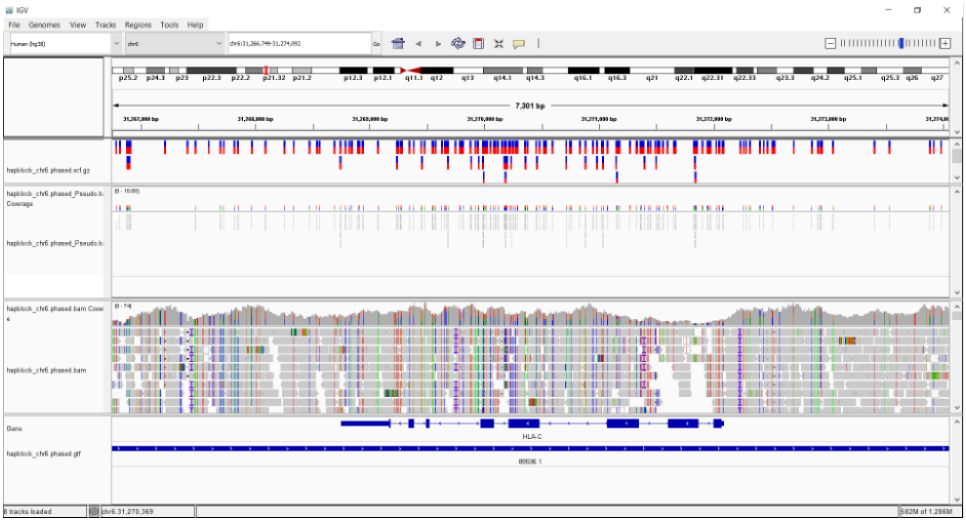




The loading order and layout are depended on the user preferences. We suggest the following layout (from top to bottom): vcf, pseudo bam, phased bam, and gtf.

After uploading the files, the users may select the genomic region or the gene symbol of interest:

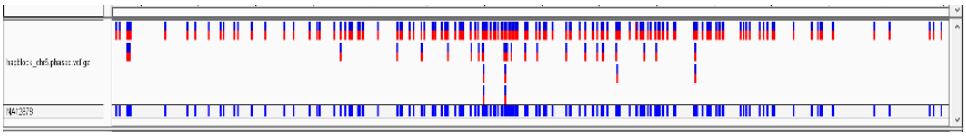




➢ Understanding each track and adjusting the settings for optimum visualization

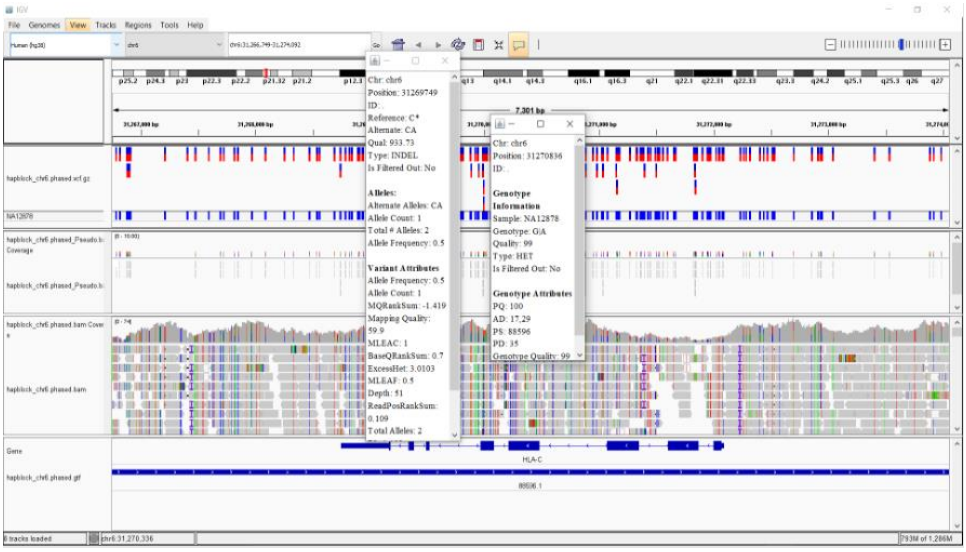
Vcf track

Tell-Sort provides a vcf file that can be uploaded as an IGV track. This track contains two track views: variants and genotypes. The users may need to re-size the track panel or scroll up and down to see all the views in this track:



The view for the variants shows vertical bars with color-coded calls relative to the reference genome at a given genomic position. Blue means the same allele as the reference. Red means a different allele from the reference. In the current Tell-Sort setting, we have only included the heterozygous variants in this vcf file—i.e., user will see all vertical bars with both blue and red. Occasionally, users may see red-only vertical bars. In those positions, the two alleles will be different from the reference. Depending on variant density, there may be multiple rows in this view to allow the users to easily find and interact the variant calls with their mouse.

The view for the genotype is below the view for the variants. It is shown in one row with little vertical blue bars. In the vcf track, both variant and genotype views will not show the detail information unless the users interact the sites with their mouse. Below are the examples when the users click the bar they are interested in:

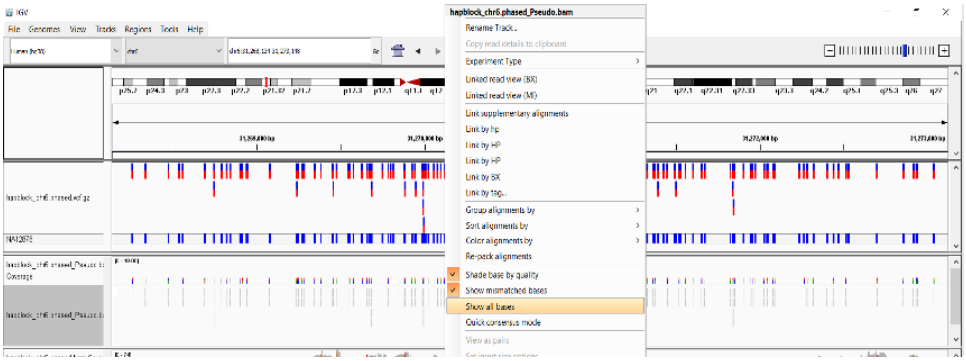


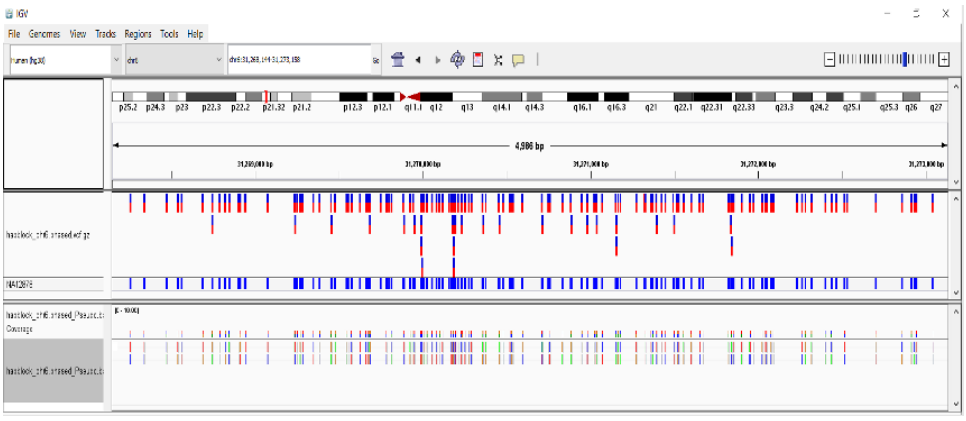
Tip: IGV allows the users to modify popup text behavior in data panels, see the red arrow points.

Pseudo bam track

Tell-Sort provides a pseudo bam file that can be uploaded as an IGV track. As described above, this track conversts variant genotype information into pseudo reads that have been mapped to the reference. The pseudo bam file includes two tags: “hp” for haplotype, and “PS” for the phase set. These two tags are used to display haplotypes in phase blocks.

After the pseudo bam file has been uploaded into IGV, the view settings will need to be adjusted. To adjust the view settings, right click mouse on the track and select “Show all bases;” right click mouse, and select “Link by hp.” For the IGV copy that doesn’t include the selection of “Link by hp”, use instead “Link by tag…,” and type “hp,” and click “OK.”





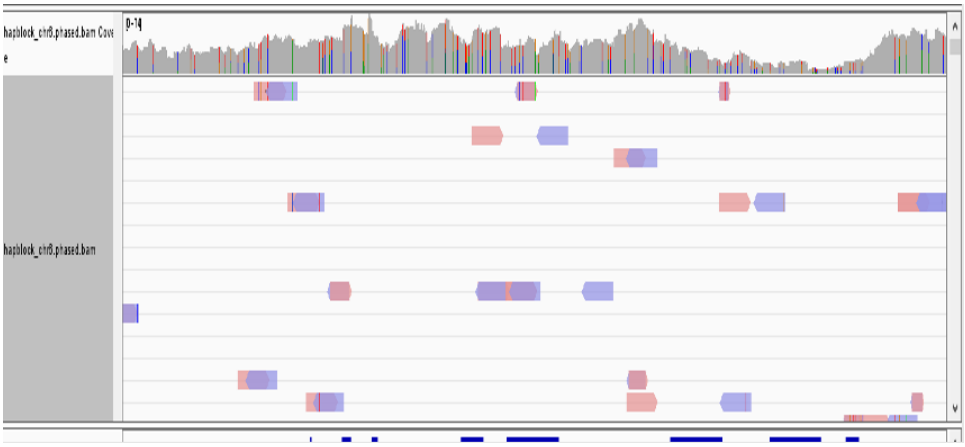
In this alignment track, since those pseudo reads are from the genotypes in the vcf file, each read represents an allele from the genotype in the giving genomic position, the view for the coverage on the top will not be meaningful and should not be read by users.We borrow this track to provide users with the convenient way to read the phased genotypes. Below the coverage view, the phased genotypes are displayed. Each haplotype has been shown with each phased allele from the heterozygous variant connected with a grey line. Since the genotypes are displayed with the nucleotide characters or the default nucleotide colors, the users will catch the phased allele sequences without click or mouse hover the displayed bars.

Read bam track

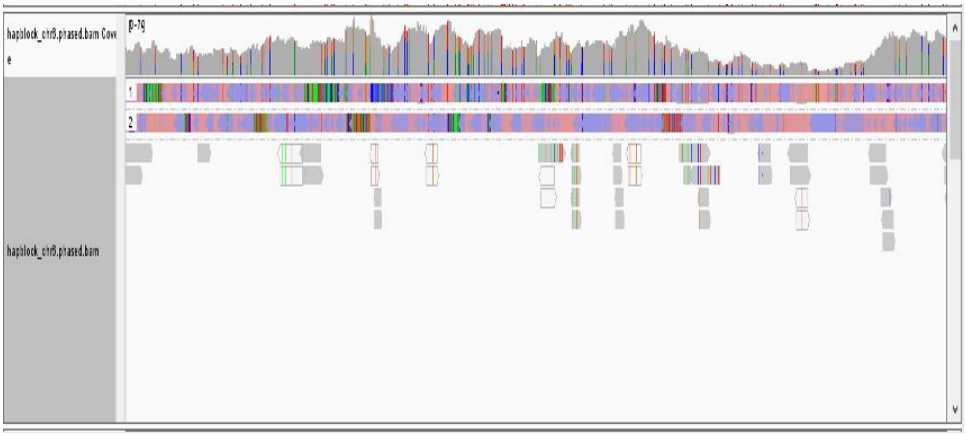
Tell-Sort provides a read bam file that can be uploaded as an IGV track to show how reads have been mapped to the reference. The read bam file includes three tags for linked reads: the BX tag for the barcode; the HP tag for the haplotype; and the PS tag for the phase set. This track shows how linked reads have been mapped; which reads have been assigned to haplotypes 1 and 2, or remain unphased; and which reads belong to a phase block.

The track has two views: coverage (on top), which is no different to shot-gun sequences; and read alignments (at the bottom). Depending on the purpose, the view settings may need to be adjusted as it follows:

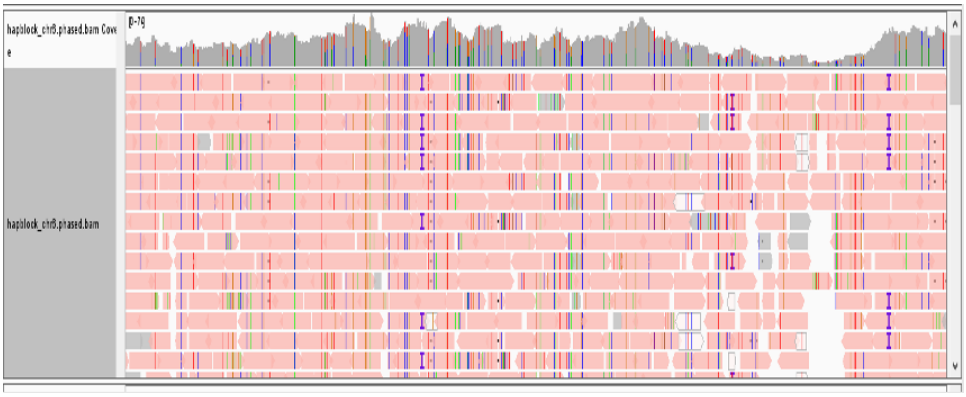
- **View the reads linked by barcodes**: right click in the track, select “Link by BX” or “Link by tag…”, and type “BX” in the text box; then click “OK”.



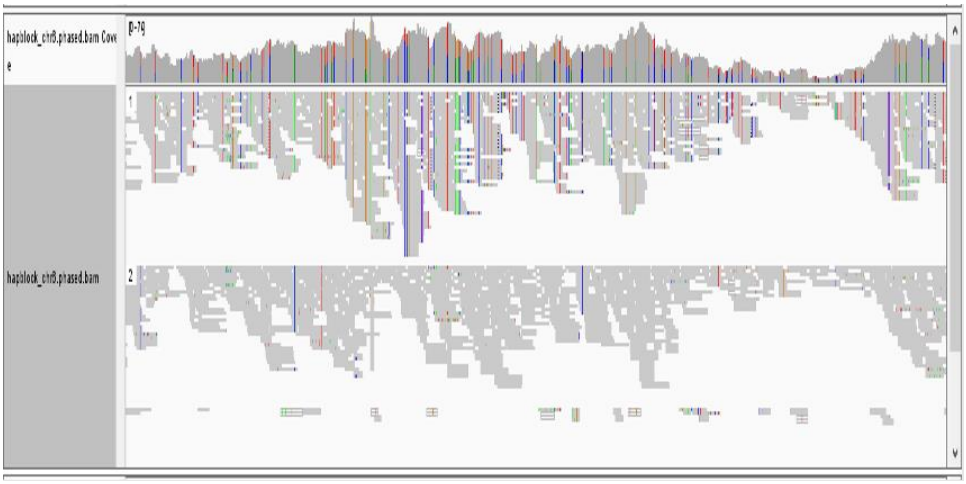
- **View the reads linked by haplotype**: right click in the track, select “Link by HP” or “Link by tag…”, and type “HP” in the text box; then click “OK”. Right click, select “Group alignments by”, and select “phase.”



- **View reads belonging to the same phased block by colors**: right click in the track, select “Color alignments by,” then select “tag,” and type “PS” in the text box; click “OK.”



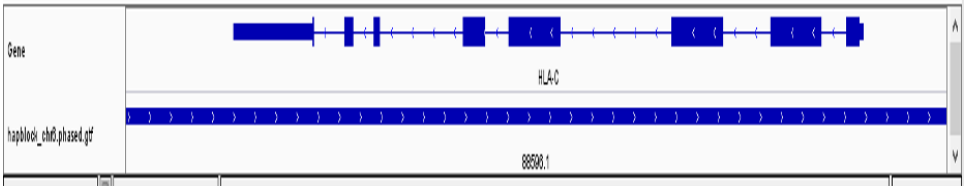
- **Linked read view with the reads in each haplotype as well as in the unphased**: right click in the track, and select “Linked read view (MI).” Right click, select “Group alignments by,” and select “phase” (or select “tag”, and in the text box, type “HP”, and click “OK”).



- There are additional view setting configurations in IGV. Users may need to customize the settings to fit their own specific purposes.

Gtf track

Tell-Sort provides a gtf file that can be uploaded as an IGV track that shows where phase blocks span along the reference. It is important to note that haplotypes that belong to different phase blocks may not derive from the same chromosomal copy (maternal or paternal). Thus, the annotation of haplotype 1 and 2 in one block may not represent the same chromosomal copy than the annotation of haplotype 1 and 2 in another or the adjacent phase block. Phase blocks are displayed as a horizontal bar underneath the gene annotation:



6. Run Tell-Sort with Singularity

This chapter outlines steps to run Tell-Sort pipeline using Singularity. If you need to learn more about Singularity container, please check out resources, such as, Singularity Tutorial on GitHub, Singularity at the NIH HPC.

1) Download and install Singularity

Follow the installation steps in the GitHub tutorial to install Singularity.

2) Running Tell-Read with Singularity

The Tell-Sort package includes a singularity image for Tell-Sort as well as a wraper script to run the pipeline in Singularity. The script is, `run_tellsort_sing.sh`. It takes exactly the same command line options as its docker counterpart. For detailed descriptions of how to run pipeline with different types of input dataset, please refer to Chapter 4.

This document is proprietary to Universal Sequencing Technology Corporation and is intended solely for the use of its customers in connection with the use of the products described herein and for no other purposes.

The instructions in this document must be followed precisely by properly trained personnel to ensure the proper and safe use of the TELL-Seq kit.

UNIVERSAL SEQUENCING TECHNOLOGY CORPORATION DOES NOT ASSUME ANY LIABILITY OCCURING AFTER INCORRECT USE OF THE TELL-SEQ KIT.

Revision History

Document #	Version	DCR Reference and comment
100024-USG	1.0.3	DCR-210082 Initial Release
100024-USG	1.1	DCR-220058 New version of SW supporting new TELL-Beads product
100024-USG	1.1.1	DCR-220085 This version will support the following products: 100035 KIT, TELL-Seq Library Reagent Box 1 V1 RUO 100036 KIT, TELL-Seq Library Reagent Box 2 V1 RUO (TELL Bead Plex option) 100043 TELL-Seq™ Library Multiplex Primer C-series (1-96) Plate
100024-USG	1.1.2	DCR-240004 This version added following changes: Option to choose alignment algorithm between “BWA” and “EMA”.