



TELL-Seq™ Data Analysis Software User Guide

**for
Tell-Read**

Table of Contents

1. Introduction	2
2. Installation	3
3. Run Tellread Pipeline	5
➤ Run Tellread on BCL Raw Data	5
➤ Run Tellread on FASTQ Raw Data	6
➤ Prepare Genome Reference Directory	7
➤ The genomes.json file	8
➤ Run Tellread pipeline without a reference	9
➤ Typical layout of a result directory	10
➤ Sample index names	11

1. Introduction

This document describes instructions on how to use 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 barcode 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 and *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 Tellread result and performs variant calling, phasing and SV.

- **Tell-Link**

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

2. Installation

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

If you don't already have a Docker running environment, you need to install the Docker engine. Docker is available in two editions: Community Edition (CE) and Enterprise Edition (EE). Following is an example for getting and installing Docker CE for Ubuntu/Debian systems. If you already have a Docker environment, you can skip these steps and go to the next paragraph for installation of Tellread docker image.

Step 1: Update Software Repositories

As usual, it's 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 so, 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
```

Step 6: Log out and log back in

After you log back in, you can 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 Tell-Read docker image.

1. Download Tell-Read docker image package `tellread.tar.gz`.
2. Unzip `tellread.tar.gz`, this will create a directory `tellread-release` that contains the docker image of the pipeline named `docker-tellread`, and three Unix shell scripts: `generateGenomeIndexBed.sh`, `run_tellread.sh`, and `run_tellread_fq.sh`.

```
$ tar xzvf tellread.tar.gz
```

3. Load the docker image

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

4. Check image `docker-tellread` is loaded

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-tellread	latest	9996bd6089c9	8 seconds ago	3.05GB

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

```
$ docker image rm -f 9996bd6089c9
```

3. Run Tellread Pipeline

The Tellread pipeline takes two types of raw data: 1) `bcl` files and 2) `fastq` data converted from `bcl` data by `bcl2fastq`.

Tellread pipeline is delivered as a docker image. Tellread package provides wrapper scripts so users can avoid the docker details.

➤ Run Tellread on BCL Raw Data

A wrapper script `run_tellread.sh` is provided to simplify the command line invocation if you are not into Docker details. Bash shell script `run_tellread.sh` takes the following format.

```
$ run_tellread.sh \
  -i <path/to/raw/data> \
  -o <path/to/output> \
  -f <path/to/reference> \
  -s <comma separated sample list> \
  -g <comma separated genome list>
```

The command line options are explained in the table below.

-i	This specifies the path to the raw data directory. For example, for a MiniSeq sequencing run, this would be <code>/data/MiniSeq/raw/180718_MN00867_0016_A000H2JVYN</code> .
-o	This specifies output directory that the results will be. For example, <code>/data/Run_Analysis/run180718</code> . NOTE: Currently, the last part of output directory name cannot have “_” character. For example, <code>run_180718</code> is NOT allowed.

-f	This specifies the directory that genome reference files are located. For example, <i>/data/genome</i> . See <i>Prepare Genome Reference Directory</i> section below on setting up the genome reference directory for Tellread pipeline.
-s	This is a comma-delimited sample index list. For example, <i>T501, T502, T503</i> . Note: No spaces in between sample names. If the run has only one sample, this parameter is not necessary. In the output, result is identified by the default sample name <i>T500</i> .
-g	This is a comma-delimited genome reference list. For example, <i>DH10B, Arab, Fly</i> . These reference names are used to retrieve specific genome FASTA files. Detailed steps on to make these reference files will be discussed in the <i>Prepare Genome Reference Directory</i> section below. Note: No spaces in between genome names.

An example is shown below,

```
$ run_tellread.sh \
-i /data/MiniSeq/raw/180718_MN00867_0016_A000H2JVYN \
-o /data/run180718 \
-f /data/genome \
-s T501,T502,T504,T508 \
-g arab,arab,arab,arab
```

In this specific case, the raw data contains 4 samples, T501, T502, T504, T508.

➤ Run Tellread on FASTQ Raw Data

The wrapper script to run Tellread pipeline on raw fastq data is `run_tellread_fq.sh`. The command line looks like following,

```
$ run_tellread_fq.sh \
-i1 <I1 read>.fastq.gz \
-i2 <I2 read>.fastq.gz \
-r1 <R1 read>.fastq.gz \
-r2 <R2 read>.fastq.gz \
-m <sequencer type> \
-o <path/to/output> \
-f <path/to/reference> \
-s <comma separated sample list> \
```

```
-g <comma separated genome list>
```

The command line options are explained in the table below.

-i1	This is a required parameter. It specifies the I1 read file in fastq.gz format.
-i2	This is an optional parameter. It specifies the sample index I2 read file in fastq.gz format. If there is only one sample in the run dataset, this parameter is not needed.
-r1	This is a required parameter. It specifies the R1 read file in fastq.gz format.
-r2	This is an optional parameter. It specifies the R2 read file in fastq.gz format in a pair-end run.
-m	This parameter specifies machine type for the sequencer. The supported values are: “MiniSeq”, “MiSeq”, “HiSeq”, “NextSeq”, and “NovaSeq”. This parameter is needed if the dataset is not sample-demultiplexed. If there is only one sample the dataset, this parameter is not needed.
-o	This is a required parameter. It specifies the output directory for analysis results.
-f	This is a required parameter. It specifies the directory that genome reference files are located.
-s	This parameter specifies a comma-delimited sample index list. See <i>Sample Index Names</i> below. If there is only one sample in the dataset, this parameter is not needed. In the output, result is identified by the default sample name <i>T500</i> .
-g	This is a comma-delimited genome reference list. For example, DH10B, Arab, Fly. These reference names are used to retrieve specific genome FASTA files. Detailed steps on to make these reference files will be discussed in the <i>Prepare Genome Reference Directory</i> section below. Note: No spaces in between genome names.

➤ Prepare Genome Reference Directory

In order to run the Tellread pipeline, genome indexes and bed files need to be created. These files will be stored in the Reference Directory. For each genome reference, a subdirectory within the Reference Directory is created that will hold the genome’s indexes and bed files. A script called “generateGenomeIndexesBed.sh” in the package helps users generate the reference subdirectory and those files. This script takes the reference genome in fasta format as the input. The created indexes will be located in the same directory of the input fasta file, and the bed files will be in the sub-directory called “bed”.

To run this script, type the following in the command line, from the same directory that reference FASTA file is located: `/path/to/generateGenomeIndexBed.sh MyGenome.fasta`

Depend on the genome size, this script running time varies, from a couple of seconds (e.g. some bacteria genomes), to a couple of hours (e.g. human genome).

Following example goes through a process of generating a reference for E.coli strain DH10b under the reference directory `/data/genomes`. It starts with the user creating a sub-directory named DH10b and copy a FASTA file `ecoli_dh10b.fasta` into it.

```
$ cd /data/genomes
$ mkdir DH10b
$ cp ecoli_dh10b.fasta DH10b
$ cd DH10b
$ ls -al
total 4652
drwxrwxr-x 2 ubuntu ubuntu    4096 Jun 10 13:47 .
drwxrwxr-x 4 ubuntu ubuntu    4096 Jun 10 13:46 ..
-rwxrwxr-x 1 ubuntu ubuntu 4753176 Jun 10 13:47 ecoli_dh10b.fasta
$ ~/generateGenomeIndexBed.sh ecoli_dh10b.fasta
[bwa_index] Pack FASTA... 0.03 sec
[bwa_index] Construct BWT for the packed sequence...
.....
$ ls -al
total 17348
drwxrwxr-x 4 ubuntu ubuntu    4096 Jun 10 13:49 .
drwxrwxr-x 4 ubuntu ubuntu    4096 Jun 10 13:46 ..
drwxrwxr-x 2 ubuntu ubuntu    4096 Jun 10 13:49 bed
-rw-rw-r-- 1 ubuntu ubuntu     37 Jun 10 13:49 ChromNameLinks.txt
-rw-rw-r-- 1 ubuntu ubuntu    104 Jun 10 13:49 ecoli_dh10b.dict
-rw-rw-r-- 1 ubuntu ubuntu 4753176 Jun 10 13:49 ecoli_dh10b.fasta
-rw-rw-r-- 1 ubuntu ubuntu     33 Jun 10 13:49 ecoli_dh10b.fasta.amb
-rw-rw-r-- 1 ubuntu ubuntu    119 Jun 10 13:49 ecoli_dh10b.fasta.ann
-rw-rw-r-- 1 ubuntu ubuntu 4686216 Jun 10 13:49 ecoli_dh10b.fasta.bwt
-rw-rw-r-- 1 ubuntu ubuntu     22 Jun 10 13:49 ecoli_dh10b.fasta.fai
-rwxrwxr-x 1 ubuntu ubuntu 4753176 Jun 10 13:47 ecoli_dh10b.fasta.original
-rw-rw-r-- 1 ubuntu ubuntu 1171536 Jun 10 13:49 ecoli_dh10b.fasta.pac
-rw-rw-r-- 1 ubuntu ubuntu 2343120 Jun 10 13:49 ecoli_dh10b.fasta.sa
drwxrwxr-x 2 ubuntu ubuntu    4096 Jun 10 13:49 ecoli_dh10b_LAM
```

➤ The genomes.json file

In order for pipeline to locate indexed genome reference files, users need to create/update the json configuration file `genomes.json`. Each genome reference is represented by an entry in the file. When a new genome reference is created, a corresponding entry is added to this file automatically by

script `generateGenomeIndexBed.sh`. When this script is run the first time, the `genomes.json` will be created. In the subsequent runs, it will be updated with new genome entry being added.

The `genomes.json` looks like,

```
[
  {
    "ref_name":      "DH10b",
    "ref_fa":        "DH10b/ecoli_dh10b.fasta",
    "ref_bed":       "DH10b/bed/ecoli_dh10b_45kby5k.bed",
    "num_chrom":     "1"
  },
  .....
]
```

In each entry, user specifies following items,

"ref_name"	A user defined reference name to be used in the pipeline command. By convention, it is the same name as the subdirectory name
"ref_fa"	Path to genome fasta file starting from the top level of the genome reference directory
"ref_bed"	Path to bed file starting from the top level of the genome reference directory
"num_chrom"	Number of chromosomes of the genome

After genome indexes subdirectories are generated for all genomes, the genome reference directory should look something like this,

```
$ls -al /data/genomes
drwxrwxr-x  4 ubuntu ubuntu 4096 Jun 10 14:39 ./
drwxr-xr-x 17 ubuntu ubuntu 4096 Jun 10 15:05 ../
drwxrwxr-x  4 ubuntu ubuntu 4096 Jun  9 22:50 Arab/
drwxrwxr-x  4 ubuntu ubuntu 4096 Jun 10 13:49 DH10b/
-rw-rw-r--  1 ubuntu ubuntu 3480 Jun 10 14:39 genomes.json
```

The name and location of the genome reference directory is set by the user. However, in order for the Tellread pipeline to locate it, the full path need to be supplied to the `-f` option of `run_tellread.sh` script.

➤ Run Tellread pipeline without a reference

If the user runs analysis with the de novo samples, the -g option is set with "NONE" as the reference name.

Currently, the Tellread analysis pipeline supports two modes for -g option:

- 1.) All samples with a known genome reference. In this case -g is followed by a list of reference names, one for each sample; The reports will have mapping statistics for the samples.
- 2.) All samples with unknown genome reference. In this case -g is followed by a list of string "NONE", one for each sample.

As of this release, the pipeline does not support the mixed mode where some of the samples' reference is known, and the other samples' reference is unknown. For users with the mixed samples, currently they can use a fake reference such as DH10b in the corresponding place of -g reference list. This will generate irrelevant mapping results in the reports that users can ignore.

➤ Typical layout of a result directory

A typical result directory looks like following.

```
$ls -al run190530
drwxrwxr-x 17 ubuntu ubuntu 4096 Jun 15 18:17 ./
drwxrwxrwx 69 root root 4096 Jun 15 13:48 ../
drwxrwxr-x 4 ubuntu ubuntu 4096 May 31 19:55 0_fastq/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:53 10_genomecov/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:53 12_long_fragment/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:53 14_SLFs/
drwxrwxr-x 4 ubuntu ubuntu 4096 May 31 19:55 1_demult/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:49 2_barcode_indiv/
drwxrwxr-x 3 ubuntu ubuntu 4096 May 31 19:53 3_bwa/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:54 4_gc_bias/
drwxrwxr-x 3 ubuntu ubuntu 4096 May 31 19:53 5_read_dist/
drwxrwxr-x 20 ubuntu ubuntu 4096 May 31 19:54 benchmarks/
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:54 download/
-rw-rw-r-- 1 ubuntu ubuntu 299 May 31 19:54 emptyplot.png
drwxrwxr-x 3 ubuntu ubuntu 4096 May 31 22:02 Full/
-rw-rw-r-- 1 ubuntu ubuntu 23840 May 31 19:54 QC_Analysis_2.md
-rw-rw-r-- 1 ubuntu ubuntu 13817 May 31 19:54 QC_Analysis_2.Rmd
-rw-rw-r-- 1 ubuntu ubuntu 2781127 May 31 19:54 QC_Analysis_run190530.html
-rw-rw-r-- 1 ubuntu ubuntu 404 May 31 19:51 run190530_correction_filter_report.txt
-rw-rw-r-- 1 ubuntu ubuntu 3279 May 31 19:54 run190530_report.txt
-rw-rw-r-- 1 ubuntu ubuntu 28 May 31 19:45 sample_index_list_run190530.txt
```

In addition to some intermediate result directories marked by step numbers, the QC report for the run is summarized in the html file `QC_Analysis_run190530.html`. The barcode I1, read R1 and read R2 fastq files are saved in `Full` directory, as,
`run190530_I1_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz`,
`run190530_R1_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz`,
and
`run190530_R2_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz`.
These linked reads data in fastq format will be the input for downstream phasing and/or *de novo* assembly pipeline processes.

```
$ ls -al Full
drwxrwxr-x 3 ubuntu ubuntu 4096 May 31 22:02 ./
drwxrwxr-x 17 ubuntu ubuntu 4096 Jun 15 18:17 ../
drwxrwxr-x 2 ubuntu ubuntu 4096 May 31 19:49 fastqc/
-rw-rw-r-- 1 ubuntu ubuntu 168216924 May 31 19:51
run190530_I1_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 210 May 31 19:51
run190530_I1_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.log
-rw-rw-r-- 1 ubuntu ubuntu 162 May 31 19:48
run190530_I1_T503.fastq.gz.corrected.log
-rw-rw-r-- 1 ubuntu ubuntu 654829938 May 31 19:51
run190530_R1_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 694160936 May 31 19:51
run190530_R2_T503.fastq.gz.corrected.fastq.err_barcode_removed.fastq.gz
```

➤ Sample index names

TELL-Seq library kit provides a number of index 2 primers for pooling samples together. The following table lists primer names and their corresponding sequences.

"T500"	"NNNNNNNNN"
" T501"	"TGAACCTT"
" T502"	"TGCTAAGT"
"T503"	"TGTTCTCT"
"T504"	"TAAGACAC"
"T505"	"CTAATCGA"
"T506"	"CTAGAACA"
"T507"	"TAAGTTCC"

"T508"	"TAGACCTA"
"T509"	"CATCCGAA"
"T510"	"TTATGAGT"
"T510"	"TTATGAGT"
"T511"	"AGAGGCGC"
"T512"	"TAGCCGCG"
"T513"	"ACGAATAA"
"T514"	"TTCGTAGG"
"T515"	"GATCTGCT"
"T516"	"CGCTCCGC"
"T517"	"AGGCTATA"
"T518"	"GCCTCTAT"
"T519"	"AGGATAGG"
"T520"	"TCAGAGCC"
"T521"	"CTTCGCCT"
"T522"	"TAAGATTA"
"T523"	"AGTAAGTA"
"T524"	"GACTTCCT"