



TELL-Seq™ Data Analysis Software User Guide

**for
Tell-Link**

Table of Contents

1. Introduction	2
2. Installation	3
3. Run Tell-Link Pipeline	5
➤ Run Tell-Link on Linked Read FASTQ Data	5
➤ Optimal k-mer size recommendations	8

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 to be used 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 Tell-Link 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-Link docker image.

1. Download Tell-Link docker image package `docker-tellink.tgz`.
2. Unzip `docker-tellread.tar.gz`, this will create a file named `docker-tellread.tar`, and a Unix shell script `run_tellink.sh`.

```
$ tar xzvf tellink.tar.gz
```

3. Load the docker image

```
$ docker load -i docker-tellink
```

4. Check image `docker-tellink` is loaded

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-tellink	latest	607af6111097	About an hour ago	1.09GB

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

```
$ docker image rm -f 607af6111097
```

3. Run Tell-Link Pipeline

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

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

➤ Run Tell-Link on Linked Read FASTQ Data

The wrapper script to run Tellink pipeline is `run_tellink.sh`. The command line looks like following,

```
$ run_tellink.sh \  
  -r1 <R1 read>.fastq.gz \  
  -r2 <R2 read>.fastq.gz \  
  -il <I1 read>.fastq.gz \  
  -r <genome reference file in fasta> \  
  -d metagenomics \  
  -o <path/to/output> \  

```

```
-k <k-mer size selected to build assembly graph> \
-lc <local k-mer size selected to build assembly graph> \
-p <prefix name>
```

-r1	This parameter specifies read 1 fastq file in gz compressed format.
-r2	This parameter specifies read 2 fastq file in gz compressed format..
-i1	This parameter specifies index 1 fastq file in gz compressed format.
-r	This parameter specifies genome reference file in fasta format. This parameter is optional.
-d	metagenomics is the only value for this parameter. This parameter is optional and can be set for assembling a metagenomics sample.
-o	This parameter specifies the output directory.
-k	This parameter specifies k-mer length for constructing global assembly graph.
-lc	This parameter specifies local k-mer length for constructing local assembly graph. See section <i>Local Assembly</i> below for more details.
-p	This parameter specifies a prefix name for identifying a result set.

An example is shown below,

```
$ run_tellink.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/DH10b/ecoli_dh10b.fasta \
  -o runTestResult \
  -k 45 \
  -lc 31 \
  -p 501
```

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

```
$ ls -al runTestResult/
```

```
drwxrwxr-x 5 ubuntu ubuntu 6144 Jun 19 15:15 ./
drwxr-xr-x 83 ubuntu ubuntu 14336 Jun 19 15:15 ../
drwxrwxr-x 3 ubuntu ubuntu 6144 Jun 19 15:18 501/
drwxrwxr-x 14 ubuntu ubuntu 6144 Jun 19 15:15 .snakemake/
```

The directory view for the resulted assembly graph, contigs and scaffolds are stored in directory 501.

```
$ ls -al runTest/501/
drwxrwxr-x 3 ubuntu ubuntu 6144 Jun 19 15:18 ./
drwxrwxr-x 5 ubuntu ubuntu 6144 Jun 19 15:15 ../
drwxrwxr-x 7 ubuntu ubuntu 6144 Jun 19 15:18 501-eval/
drwxrwxr-x 5 ubuntu ubuntu 22528 Jun 19 15:18 __skipping/
-rw-rw-r-- 1 ubuntu ubuntu 36166 Jun 19 15:18 aligned.paf
-rw-rw-r-- 1 ubuntu ubuntu 6306 Jun 19 15:18 dotplot.eps
-rw-rw-r-- 1 ubuntu ubuntu 4693277 Jun 19 15:18 scaffolds.fasta
-rw-rw-r-- 1 ubuntu ubuntu 4654220 Jun 19 15:17 scaffolds.full.fasta
-rw-rw-r-- 1 ubuntu ubuntu 0 Jun 19 15:18 tengicungduoc
```

The QUAST reports for assembly performance is stored in the evaluation directory under the label 501-eval.

```
$ ls -al runTest/501/501-eval/
drwxrwxr-x 7 ubuntu ubuntu 6144 Jun 19 15:18 ./
drwxrwxr-x 3 ubuntu ubuntu 6144 Jun 19 15:18 ../
drwxrwxr-x 2 ubuntu ubuntu 6144 Jun 19 15:18 aligned_stats/
drwxrwxr-x 2 ubuntu ubuntu 6144 Jun 19 15:18 basic_stats/
drwxrwxr-x 3 ubuntu ubuntu 6144 Jun 19 15:18 contigs_reports/
drwxrwxr-x 2 ubuntu ubuntu 6144 Jun 19 15:18 genome_stats/
-rw-rw-r-- 1 ubuntu ubuntu 53735 Jun 19 15:18 icarus.html
drwxrwxr-x 2 ubuntu ubuntu 6144 Jun 19 15:18 icarus_viewers/
-rw-rw-r-- 1 ubuntu ubuntu 4934 Jun 19 15:18 quast.log
-rw-rw-r-- 1 ubuntu ubuntu 376398 Jun 19 15:18 report.html
-rw-rw-r-- 1 ubuntu ubuntu 39481 Jun 19 15:18 report.pdf
-rw-rw-r-- 1 ubuntu ubuntu 2111 Jun 19 15:18 report.tex
-rw-rw-r-- 1 ubuntu ubuntu 1071 Jun 19 15:18 report.tsv
-rw-rw-r-- 1 ubuntu ubuntu 2200 Jun 19 15:18 report.txt
-rw-rw-r-- 1 ubuntu ubuntu 1720 Jun 19 15:18 transposed_report.tex
-rw-rw-r-- 1 ubuntu ubuntu 1071 Jun 19 15:18 transposed_report.tsv
-rw-rw-r-- 1 ubuntu ubuntu 1918 Jun 19 15:18 transposed_report.txt
```


➤ Optimal k-mer size recommendations

Tell-Link uses barcode-aware reads to construct assembly graph, compute contigs and build scaffold. The choice of k-mer size is essential for an assembly graph and can affect assembly result.

Genome repeats can cause great difficulty for *de novo* assembly processes. They tangle the assembly graph and shorten contigs. Tell-Link first utilizes barcode information to resolve complex structures in the assembly graph at the global level. It then uses an additional local assembly process to simplify the assembly graph by collecting a set of reads that shared the same barcodes between two edges and reconstructing local assembly from that set. The local assembly graph is simpler than the global assembly graph and therefore, easier to be untangled using read-pair information.

Tell-Link pipeline gives users options to specify k-mer sizes for both global assembly graph and local assembly graph for achieving optimal assembly result. Usually for each sequencing dataset, one tests with different combinations of k-mer sizes in order to arrive the best assembly.

In general, we recommend following combinations of k-mer sizes for different library read lengths.

- For read length < 100bp, the global k-mer size range: 45-55, with local k-mer size: 27-31
- For read length ~150bp, the global k-mer size range: 99-115, with local k-mer size: 69-79.