TELL-Seq™ Data Analysis Software User Guide

for

Tell-Link

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

Document # 100025 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**

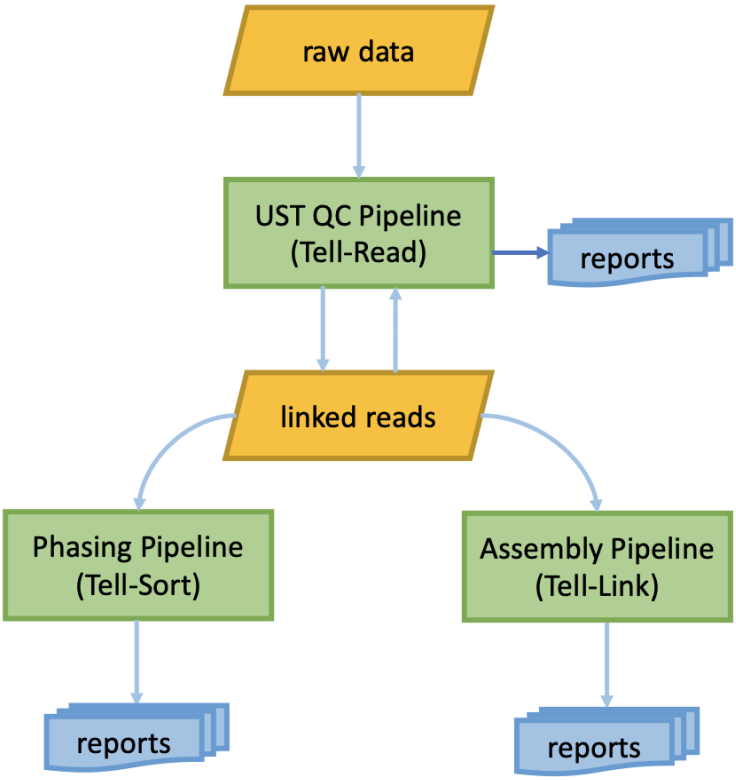
  a set of pipeline processes that takes as input the sequencing output from an NGS sequencing instrument and generates linked-read FASTQ data, as well as QC reports.

- **Tell-Sort**

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

- **Tell-Link**

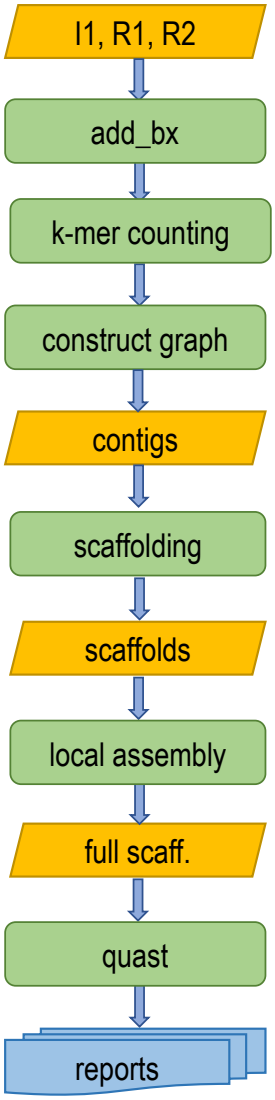
  de novo assembly pipeline processes that build barcode-aware assembly graph, assembles contigs and performs scaffolding.


raw data
UST QC Pipeline
(Tell-Read)
reports
linked reads
Phasing Pipeline
(Tell-Sort)
Assembly Pipeline
(Tell-Link)
reports
reports

`

2. Tell-Link Pipeline

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



The following is a brief description of major components 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.

K-mer Counting

We use kmer-counting function of KMC library to read fastq reads into memory and obtain the kmer sequence and its frequency.

Construct Graph

For each k+1 mer, we extract one kmer prefix and one kmer suffix to create two nodes in the de Bruijn graph. We then add this k+1 mer as the edge between these two nodes to the graph. We simplify the assembly graph iteratively by performing these procedures: tip removal, chimeric edges removal, bubble and loop removal and bulges removal. We stop the iteration when there is no more simplifying operation that can be executed.

Scaffolding

We use both read pair information, which is useful in connecting contigs disjoint by small gaps, and barcode sharing information, which is helpful in connecting contigs disjointed by large gaps, to construct scaffolds.

Local Assembly

For two contigs that are predicted to be consecutive on the genome, finding the path between them on the assembly graph is not straightforward. This region usually comprises similar k-mer compositions from many copies of a repetitive sequence in the genome. Rather than ignoring it and filling ambiguous characters between the two consecutive contigs, we attempt to de novo assemble the repeat region between them locally using linked-read information. We use a local k-mer to build the de Bruijn graphs in local (gap) regions.

Quast

After the final scaffolds are done, we use Quast to evaluate assembly performance.

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'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 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-Link docker image.

1. Download the Tell-Link docker image package `tellink.tar.gz`.

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

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

3. Load the docker image

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

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-tellinkt` to upgrade to a newer version

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

4. Run Tell-Link Pipeline

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

The Tell-Link pipeline is delivered as a docker image. The 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 is `run_tellink.sh`. The command line looks like following,

```
$ run_tellink.sh \
      -r1 /path/to/R1/data \
      -r2 /path/to/R2/data \
      -i1 /path/to/I1/data \
      -r <genome reference file in fasta> \
      <-d metagenomics> \
              -o <path/to/output> \
-c 1 \
-n <number> \

      -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.
-c	When this optional parameter is set, the pipeline will try to circularize contigs. Use –c 1.

-n	This optional parameter specifies max number of local assembly iterations.
-k	This optional parameter specifies k-mer length for constructing global assembly graph.
-lc	This optional parameter specifies local k-mer length for constructing local assembly graph. See section *Local Assembly* below for more details.
-p	This parameter specifies a prefix name for identifying a result set.
-j	This optional parameter specifies number threads to run.

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 \
  -j 30
```

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`. The final assembly result is in file `assembly.fasta`.

```
$ 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/
drwxr-xr-x 5 ubuntu ubuntu   18432 Jun 19 14:44 __skipping_57/
drwxr-xr-x 5 ubuntu ubuntu   14336 Jun 19 14:26 __skipping_77/
 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 4654220 Jun 19 15:18 assembly.fasta
 -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
```

’ Auto-selected k-mer sizes by default

The pipeline now automatically selects optimal global and local k-mer values. For general use, user no longer needs to set –k and –lc options.

For advanced users who still need to find best possible k-mer combinations, -k and –lc options are still available for use. Please refer to section “Optimal k-mer size recommendations” for discussions of k-mer size selection.

’ Customized 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 to arrive at the best assembly.

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

- For read length < 100bp, the global k-mer size range: 35-55, with local k-mer size: 23-31
- For read length between 100bp and 130bp, the global k-mer size range: 55-85, with local k-mer size 35-45
- For read length ~150bp, the global k-mer size range: 99-115, with local k-mer size: 69-79.

In the latest Tell-Link pipeline, Tell-Link executes local assembly process by applying an optimal k-mer for each individual gap instead of same k-mer size for all gaps. The selected local k-mer size by user now becomes the seed k-mer. Tell-Link pipeline searches and applies optimal local k-mer size around seeded k-mer size to try to fill the gaps in an iterative manner.
Tell-Link uses following formula to select a k-mer size for each iteration,

$$k_{n+1} = k_n + (-1)^n 4n, \qquad n = 0, \ldots, 6$$

For example, when user set , the pipeline will use following k-mer sizes, 45, 41, 49, 37, 53, 33, 57, for up to 7 iterations of local assemblies for yet unfilled gaps. The iteration can stop before all 7 rounds are complete when pipeline determines no further improvement can be achieved.

To make sure assembly can finish within a reasonable timeframe, the number of k-mer iterations is further limited depending on complexity of contig scaffolding.

To limit max number of local assembly iterations, use command line option –n <number>, by specifting a value <number> that is smaller than 6.

5. Run Tell-Link with Singularity

This chapter outlines steps to run Tell-Link 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) Runing Tell-Read with Singularity

The Tell-Link package includes a singularity image for Tell-Link as well as a wrapper script to run the pipeline in Singularity. The script is, `run_tellink_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
100025-USG	1.0.3	DCR-210082 Initial Release
100025-USG	1.1	DCR-220058 New version of SW supporting new TELL-Beads product
100025-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
100025-USG	1.1.2	DCR-240004 This version added following changes: Auto kmer selection by default; Option to check if a contig is circular; Option for max number local assembly iterations.