TELL-Seq™ Data Analysis Software User Guide

for

Tell-Read

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

Document # 100023 Version 1.1

August 2022

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, metagenomic studies, *de novo* sequencing assembly, etc.

Tellysis software comes in the form of three main pipelines:

- **Tell-Read**

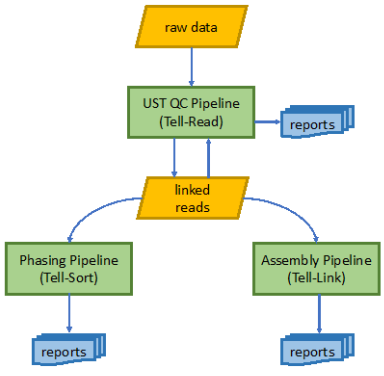
  a set of pipeline processes that takes as input the sequencing output from an NGS sequencing instrument and generates linked-read FASTQ data, as well as QC reports.

- **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**

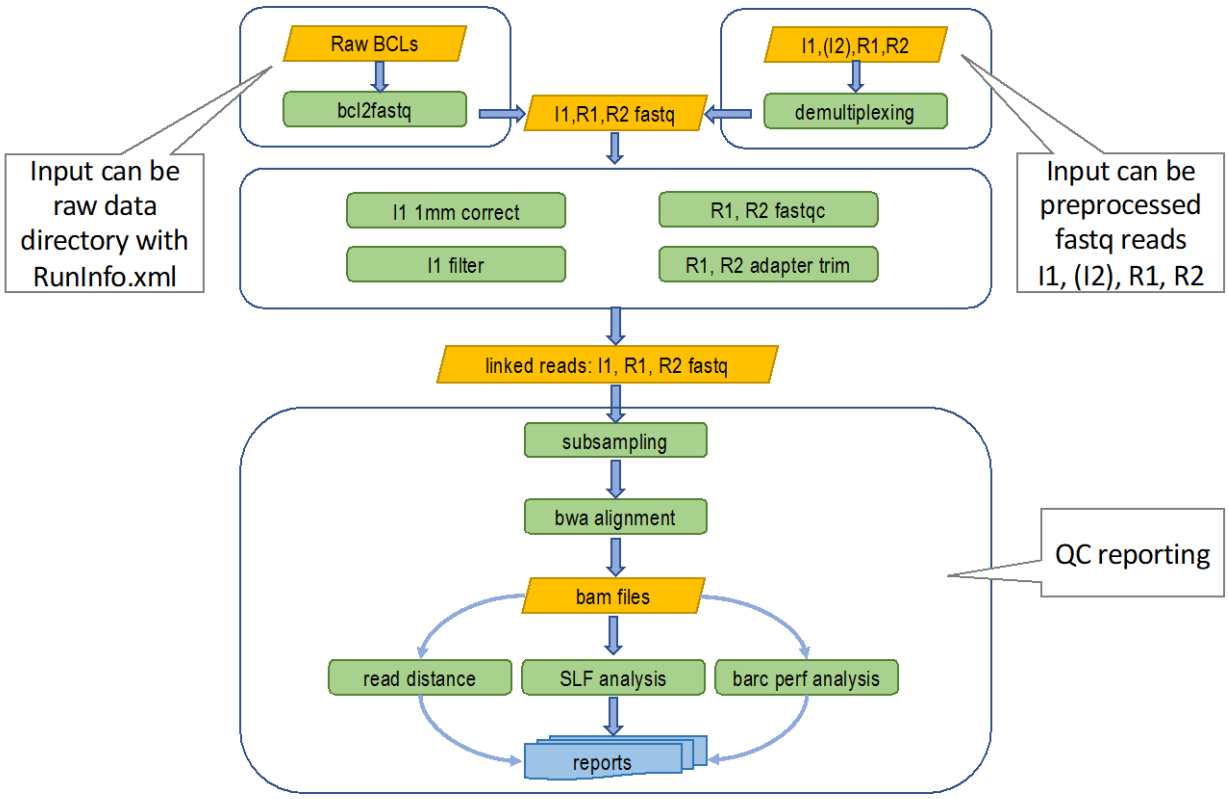
  de novo assembly pipeline processes that build barcode-aware assembly graph, assembles contigs and performs scaffolding.



Tell-Seq Data Analysis Pipelines

2. Tell-Read Pipeline

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



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

BCL to FASTQ Transformation and Sample Demultiplexing

The pipeline can take either raw BCL run data or already-converted FASTQ files as input. When raw BCL run data is the input, the pipeline uses `bcl2fastq` tool to convert and demultiplex BCL data into sample-separated FASTQ files, I1, R1 and R2. When the input is in FASTQ format, the pipeline runs `demultiplex` program to generate per-sample read files, I1, R1 and R2. I1 reads are the TELL_Seq barcode sequences. For each sequencing library construction, a set of unique barcode sequences was randomly chosen from a 2.4 billion-barcode pool. These sample-demultiplexed FASTQ files are saved as the raw data output files.

Reads Clean Up

The next step of the pipeline is the QC processing of I1, R1 and R2 files. Read sequencing quality is processed by `fastqc`. Adapter sequences in R1 and R2 are trimmed using the `cutadapt` utility. The adapter-trimmed reads are then further processed.

Unique barcodes associated with only one read are most likely caused by sequencing errors in the barcode. These barcodes are first identified if they are 1-base mismatched with one of the barcodes associated with multiple reads, and then error-corrected. Barcodes with errors after this step are filtered out. The erroneous barcodes along with their associated reads are removed and excluded from the rest of analyses.

The remaining R1 reads and R2 reads, along with their associated I1 reads (barcodes) are the TELL_Seq linked reads. They are the input for downstream analyses, such as phasing, variant calling, SV detections and de novo assembly.

QC Reporting

Subsampling for Performance Analysis

The rest of the Tell-Read pipeline uses a randomly selected subset (12,000) of unique I1 reads along with their R1 and R2 reads to evaluate the library and linked read performance. The subsampled reads are mapped to the reference genome using `bwa`. Various barcode and read statistics can then be assessed, such as, total mapped reads, duplicate rate, raw barcode statistics, barcode processing statistics, distribution of barcode and barcode associated reads.

Read Distance

One important property to pay attention to is the distribution of distances between the nearest alignments of the same barcode for all mapped reads. The bimodal distribution can be used to gauge the quality of the linked reads. A good library should have a high linked read peak (1st peak) and smaller (ideally less than half of the 1st peak by height) distal peak (2nd peak). This is usually achieved by the proper DNA to TELL bead ratio and sufficient sequencing depth.

SLF Analysis

Super Long Fragment (SLF): identified by sequencing as the original fragments which generate linked barcoded reads. It can be used as a representation of the gDNA fragments (DNA input). This sheds light on the input DNA quality and linked read performance. Since multiple SLFs can be tagged by the same barcode, a maximum distance threshold of 50kb is used allocate reads separated longer than this threshold value to different SLFs.

Understand Output

Following main artifacts of the pipeline can be found in the output directory.

- Raw FASTQ files can be found in <output>/1_demult/Raw directory.
- Final error-corrected FASTQ files are in <output>/Full directory.
- The QC summary report QC_Analysis_<run>.html. A detailed description of this report is given in Chapter 6.

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 set up 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
```

Step 6: 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-Read docker image.

1) Download the Tell-Read docker image package `tellread.tar.gz`.

2) Unzip `tellread.tar.gz`, and this will create a directory `tellread-release` which contains the docker image of the pipeline called `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
```

4. Run Tell-Read Pipeline

The Tell-Read pipeline can take as input either one of the two types of raw data: 1) `bcl` files and 2) `fastq` data converted from `bcl` data by `bcl2fastq`.

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

➢ Run Tell-Read on BCL Raw Data

A wrapper script `run_tellread.sh` is provided to simplify the command line invocation. 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 */data/MiniSeq/raw/180718_MN00867_0016_A000H2JVYN.*
-o	This specifies the output directory to store the results. For example, */data/Run_Analysis/run180718.*
-f	This specifies the directory that contain all genome reference files. For example, */data/genome.* See *Prepare Genome Reference Directory* section below on setting up the genome reference directory for the Tell-Read pipeline.
-s	This is a comma-delimited sample index list. For example, *T501,T502,T503.* Note: No spaces between sample names. If the run has only one sample, this parameter is not necessary and the result is identified by the default sample name *T500*.
-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 and are specified during the preparation of the genome reference directories. Detailed steps on how to prepare these reference files are discussed in the *Prepare Genome Reference Directory* section below. ***Note:*** *No spaces between genome names.*

Example 1: multiple samples

```
$ 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, and all will use Arab as the reference.

Example 2: single sample

```
$ run_tellread.sh \
      -i /data/MiniSeq/raw/180718_MN00867_0016_A000H2JVYN \
      -o /data/run180718 \
      -f /data/genome \
      -g Arab
```

In this specific case, the raw data contains single sample. By default, the sample name is T500.

Example 3: de novo samples

```
$ run_tellread.sh \
      -i /data/MiniSeq/raw/180718_MN00867_0016_A000H2JVYN \
      -o /data/run180718 \
      -s T501,T502,T504,T508 \
      -g NONE,NONE,NONE,NONE
```

In this specific case, the raw data contains 4 samples and the genome references are not available. When running in this mode, genome reference directory option -f can be omitted, and the genome name should be specified as "NONE".

➢ Run Tell-Read on FASTQ Raw Data

The wrapper script to run Tell-Read pipeline on raw FASTQ data is `run_tellread_fq.sh`. The command line looks like following,

```
$ run_tellread_fq.sh \
      -i1 </path/to/I1_read.fastq.gz> \
      -i2 </path/to/I2_read.fastq.gz> \
      -r1 </path/to/R1_read.fastq.gz> \
      -r2 </path/to/R2_read.fastq.gz> \
      -o <path/to/output> \
      -f <path/to/reference> \
      -s <comma separated sample list> \
      -g <comma separated genome list>
```

Example 1: multiple samples

```
$ run_tellread_fq.sh \
      -i1 ~/runTraining190704/Test_I1_001.fastq.gz \
      -i2 ~/runTraining190704/Test_I2_001.fastq.gz \
      -r1 ~/runTraining190704/Test_R1_001.fastq.gz \
      -r2 ~/runTraining190704/Test_R2_001.fastq.gz \
      -o /data/runTraining190704_test \
      -f /data/genome \
      -s T501,T506,T516 \
      -g hg38,hg38,hg38
```

In this example, the input fastq file includes multiple samples. The -s option is needed to demultiplex samples.

Example 2: single sample

```
$ run_tellread_fq.sh \
      -i1 ~/runTraining/Test_I1_T501_raw.fastq.gz \
      -i2 ~/runTraining/Test_I2_T501_raw.fastq.gz \
      -r1 ~/runTraining/Test_R1_T501_raw.fastq.gz \
      -r2 ~/runTraining/Test_R2_T501_raw.fastq.gz \
      -o /data/runTraining_test \
      -f /data/genome \
      [-s T501 \]
      -g hg38
```

In this example, the input fastq file is already sample-demultiplexed, the -s option is not needed. However, if -s option is omitted, the result will be given default sample name T500. If -s T501 is specified, the result will show sample name T501.

Example 3: de novo samples

```
$ run_tellread_fq.sh \
      -i1 ~/runTraining190704/Test_I1_001.fastq.gz \
      -i2 ~/runTraining190704/Test_I2_001.fastq.gz \
      -r1 ~/runTraining190704/Test_R1_001.fastq.gz \
      -r2 ~/runTraining190704/Test_R2_001.fastq.gz \
      -o /data/runTraining190704_test \
      [-f /data/genome \]
      -s T501,T506,T516 \
      -g NONE,NONE,NONE
```

In this example, the pipeline is running in de novo mode with genome names specified as "NONE". Genome reference directory is not needed. So -f option can be omitted.

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.
-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 *Sample Index Names* 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 *T500*. ***Note:*** *No spaces between sample names.*
-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 *Prepare Genome Reference Directory* section below. ***Note:*** *No spaces between genome names.*

' Prepare Genome Reference Directory

Genome Reference Directory is the root reference directory that contains individual genome subdirectories.

If the genome reference is known for the sequenced samples, detailed performance report on Tell-Seq library can be generated. To this end, Tell-Read pipeline randomly selects a subset of sequencing reads and barcodes to align with the reference genome. To prepare the genome reference for this purpose, genome indexes and bed files need to be created. For each genome reference, a subdirectory within the root 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. This script takes 3 inputs: the genome reference in FASTA format, the full path to the root reference directory, and the genome reference name. Note, the genome reference name is specified by user. A subdirectory of this name will be created in the root reference directory. This name will be used later to reference the genome when running the pipeline. The created indexes will be in the same directory as 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:

```
/path/to/generateGenomeIndexBed.sh MyGenome.fasta ReferenceDir GenomeRefName
```

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).

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

```
$ cd /data/genomes   # root reference directory contains multiple individual
genome reference subdirectories
$ /path/to/generateGenomeIndexBed.sh ecoli_dh10b.fasta /data/genomes DH10b
[bwa_index] Pack FASTA... 0.03 sec
[bwa_index] Construct BWT for the packed sequence...
……
$ ls -al DH10b
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

Under the root genome reference directory, there is a file named genomes.json. Each individual 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 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 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 Tell-Read pipeline to locate it, the full path needs to be supplied to the `-f` option of the `run_tellread.sh` script.

➢ Run Tell-Read pipeline without a reference

If the user runs the analysis with the de novo samples, the genome reference name used for -g option is "`NONE`".

Currently, the Tell-Read 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. The other option is to run separate analysis for de novo samples.

➢ 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 several index 2 primers for pooling samples together. The following table lists primer names and their corresponding sequences.

T500	NNNNNNNN
T501	TGAACCTT
T502	TGCTAAGT

T503	TGTTCTCT
T504	TAAGACAC
T505	CTAATCGA
T506	CTAGAACA
T507	TAAGTTCC
T508	TAGACCTA
T509	CATCCGAA
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

5. Run Tell-Read with Singularity

This chapter outlines steps to run Tell-Read 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-Read package includes a singularity image for Tell-Read as well as wraper scripts to run the pipeline in Singularity. The scripts are, run_tellread_sing.sh and run_tellread_fq_sing.sh. They take exactly the same command line options as their docker counterparts. For detailed descriptions of how to run pipeline with different types of input dataset, please refer to Chapter 4.

6. Tell-Seq Run Analysis Report

This chapter gives an explanation on major sections of the QC report.

FastQC

This is standard FastQC tool. We extract some of FastQC analysis output in our report.

Index 1	Standard fastqc report for sample demultiplexed barcode reads (I1)
Read 1	Standard fastqc report for sample demultiplexed reads (R1)
Read 2	Standard fastqc report for sample demultiplexed reads (R2)

Overrepresented Sequences

We use this to monitor adapter dimer level. For each sample, we add these percentage value together, if it is <3%, the library is considered as clean.

Read Distance

Mapped read distance (All)

The plot of distribution of distances between the nearest alignments of same barcode for all mapped reads.

The bimodal distribution can be used to gauge the quality of the linked reads. A good library should have a high linked read peak (1st peak) and smaller (ideally less than half of the 1st peak by height) distal peak (2nd peak). This is usually achieved by the proper DNA to TELL bead ratio and sufficient sequencing depth.

In the plot label, used_reads (a,b), a is the percentage of reads over unique reads, b is the percentage of reads over total reads.

PE Insert Length

The insert length is taken from alignment file.

Raw Barcode Statistics

total_reads	The total number of reads for the sample specified
reads_with_barcode_all_Gs	Number of reads with the barcode whose sequence is all Gs
reads_with_correct_barcode	Number of reads with the barcode that passed raw filters
reads_with_error_barcode	Number of reads with the barcode that didn't passe raw filters
%reads_with_correct_barcode	reads_with_correct_barcode / total_reads x 100%
%reads_with_error_barcode	reads_with_error_barcode / total_reads x 100%
unique_barcode	The total number of unique barcodes for the specified sample
unique_correct_raw_barcode	The total number of unique barcodes that passed raw filters for the specified sample
mean_#reads/correct_barcode	reads_with_correct_barcode / unique_correct_raw_barcode

Barcode Processing Statistics

barcode_with_single_read	The total number of barcodes with only one read
barcode_with_more_than_3_reads	The total number of barcodes with more than 3 reads. This value gives us an estimate on the number of barcodes used in the reaction. It is probably still larger than the actual number of barcodes, but not too far off.
reads_related_to_barcode_with_more_than_3_reads	The total number of reads associated with the barcodes in the previous row
1mismatch_barcode_corrected	Number of 1 Hamming distance barcodes corrected. Many single count barcodes are generated due to sequencing errors. We are able to correct some of them.
error_barcode_number	Number of barcodes that don't pass filters. The reads associated with these barcodes will be excluded from the downstream analyses.
final_correct_barcode_number	Number of barcodes after removal of erroneous barcodes
final_reads_number	The total number of reads associated with correct barcodes and will be used for downstream analyses.

Subsampling Analysis Using Reads Associated with 12,000 Unique Barcodes

For the rest of report, we used subset of data from 12,000 unique barcodes to evaluate library and sequencing performance.

Read Alignment Statistics

read_type	Single-ended (SE) or pair-ended (PE) reads
read_length	Number of bps in read sequences
cluster_number	Total number of read pairs
reads_mapped (R1 + R2)	Number of mapped R1 reads + number of R2 reads
reads_mapped (R1 + R2) percentage	Reads_mapped / (cluster_number x 2) x 100%
read1_reads_mapped	Number of mapped R1 reads
read1_reads_mapped_percentage	read1_reads_mapped / cluster_number
read2_reads_mapped	Number of mapped R2 reads
read2_reads_mapped_percentage	Read2_reads_mapped / cluster_number
duplicates	Total number of duplicate reads
duplication rate	Duplicates / cluster_number. TELL_Seq runs normally see this value around 25% - 35%.
read1_reads_total	Number of records in R1 fastq file
read1_duplicates	Number of duplicate R1 reads
read1_duplicate_rate	read1_duplicates / read1_reads_total x 100%
read2_reads_total	Number of records in R2 fastq file
read2_duplicates	Number of duplicate R2 reads
read2_duplicate_rate	Read2_duplicates / read2_reads_total x 100%

Read/Barcode Statistics

This table displays some statistics on the distribution of barcode and barcode associated reads

total_reads	The total number of aligned read pairs with correct barcode
unique_reads	The total number of unique mapped read pairs
total_uniq_barcodes_genome	The total number of unique barcodes
barcode_with_single_read_count	Number of unique barcodes with single read
barcode_with_2-3_read_count	Number of unique barcodes with 2 or 3 reads
barcode_with_4ormore_read_count	Number of unique barcodes with at least 4 reads
single-read-barc_reads_unique	Number of unique reads associated with single-read barcodes

2-3-read-barc_reads_unique	Number of unique reads associated with 2- or 3-read barcodes
4ormore-read-barc_reads_unique	Number of unique reads associated with multiple-reads (>=4) barcodes
single-read-barc_reads_all	Number of all reads associated with single-read barcodes
2-3-read-barc_reads_all	Number of all reads associated with 2- or 3-read barcodes
4ormore-read-barc_reads_all	Number of all reads associated with multiple-reads (>=4) barcodes

SLFs Analysis

Super Long Fragment (SLF): identified by sequencing as the original fragments which generate linked barcoded reads. It can be used as a representation of the gDNA fragments (DNA input). This table sheds a light on the input DNA quality and linked read performance.

Mean Value

Reads_number_in_SLFs(>=1 read)	Mean number of reads per SLF
Reads_number_in_SLFs(>=2 reads)	Mean number of reads per SLF when single-read SLFs are excluded
Size_of_SLFs (including 1 read SLF)	Mean SLF size when single-read SLFs are included. If this metrics is close to 10,000bp, it will indicate there were very little low molecular weight DNA in the input.
Size_of_SLFs (only >4kbp)	Mean SLF size when SLFs larger than 4kbp are included in calculation
Number_of_SLFs_for_each_barcode	Mean number of SLFs for each barcode. High number of SLFs for barcode is a major contributor to the high level of distal reads in the Read Distance plot. For human size genome, we target 10-12.
Number_of_Chromosomes_for_each_barcode	Mean number of chromosomes for each barcode

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
100023-USG	1.0.3	DCR-210082 Initial Release
100023-USG	1.1	DCR-220058 New version of SW supporting new TELL-Beads product