TU Liver Diseases Predictor Manual

Anokhi Haria, Amogh Ananda Rao, Yun-Shuo Chou, Xinyue Lu

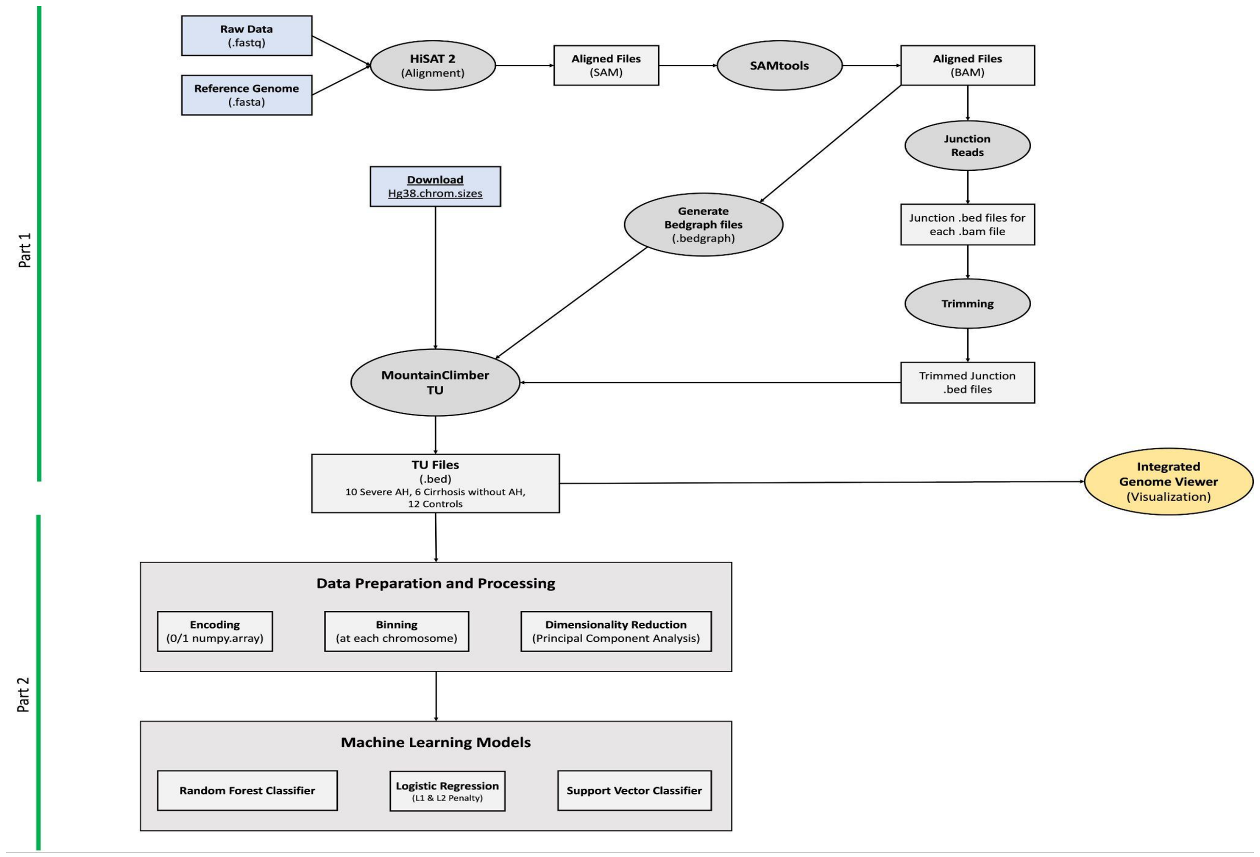
Table of Contents

Introduction

About this Pipeline

All necessary scripts and files are available on the project's Github repository. This pipeline was implemented on Bridges2 supercomputing systems. This pipeline has two parts: PART I downloads and processes data, while PART II uses machine learning models to make predictions. The first step of PART I is to download and convert SRA sequencing data to FASTQ format using a script. The data is then aligned to a reference genome and converted to BAM format using popular bioinformatics tools. MountainClimber is used to identify transcriptional units, generating .bed files that are used as input for PART II.
For PART II, after we obtained the .bed files recording the transcriptional units, we performed dimension reduction to extract the features and feed the features into machine learning classification models(Logistic regression, Logistic regression with L1 penalty, Random Forest, SVM) to further predict the disease state.

Workflow


Part 1
Part 2
Raw Data (.fastq)
Reference Genome (.fasta)
HiSAT 2 (Alignment)
Aligned Files (SAM)
SAMtools
Aligned Files (BAM)
Junction Reads
Junction .bed files for each .bam file
Trimming
Trimmed Junction .bed files
Download Hg38.chrom.sizes
Generate Bedgraph files (.bedgraph)
MountainClimber TU
TU Files (.bed) 10 Severe AH, 6 Cirrhosis without AH, 12 Controls
Integrated Genome Viewer (Visualization)
Data Preparation and Processing
Encoding (0/1 numpy.array)
Binning (at each chromosome)
Dimensionality Reduction (Principal Component Analysis)
Machine Learning Models
Random Forest Classifier
Logistic Regression (L1 & L2 Penalty)
Support Vector Classifier

Packages and Installation

Conda:

Conda is an open-source, cross-platform tool designed to manage packages, dependencies, and environments for various programming languages, including Python, R, Lua, Scala, and Java. It is particularly popular among data science teams that primarily use Python. While traditional Python users typically rely on pip for package management and venv for environment management, conda offers a comprehensive solution by efficiently handling both packages and working environments in one tool.

Installation:

To download and use the conda package, please refer to the website https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html. For the PSC server, you can follow the bash commands below:

```
Unset
wget https://repo.anaconda.com/miniconda/Miniconda3-py310_23.3.1-0-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
```

This will install the latest version of miniconda on your machine by following the prompt. You also need to rerun the bash script to set the environment variable (Assuming you are using the bash shell).

```
Unset
source ~/.bashrc
```

Then, you can check if the installation is successful by:

```
Unset
conda --version
```

This should display the installed conda version.

SRA toolkit:

The Sequence Read Archive (SRA) Toolkit is a collection of command-line tools and libraries for working with high-throughput sequencing data in the SRA format. It allows users to download SRA data, convert SRA files to other formats, and perform quality control and filtering of SRA data.

Installation:

The SRA toolkit can be downloaded from the NCBI website: https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit . From the website, you can download the appropriate package for your operating system. Once you have downloaded the package, you can follow the installation instructions to install it on your computer.
For example, the operating system on the PSC is centOS, you can download the non-sudo tar archive of SRA toolkit using the following bash commands:

```
Unset
wget https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/3.0.2/sratoolkit.3.0.2-centos_linux64.tar.gz
```

HiSAT2:

HiSAT2 is a fast and accurate alignment tool for mapping RNA sequencing reads to a reference genome. It uses a hierarchical indexing strategy to achieve fast alignment while maintaining high sensitivity and specificity. HiSAT2 can output the alignments in standard SAM/BAM format, which can be further processed by downstream analysis tools.

Installation:

HISAT2 can be downloaded from the HISAT2 website: http://daehwankimlab.github.io/hisat2/download/. You can download the appropriate package for your operating system from the website. Once you have downloaded the package, you can follow the installation instructions to install it on your computer. Since we use conda to manage the packages, run the following command to install HiSAT2 with conda.

```
Unset
conda install -c bioconda hisat2
```

Samtools:

Samtools is a suite of tools for working with SAM/BAM files, which are standard formats for storing aligned sequencing reads. Samtools can be used to manipulate and filter SAM/BAM files, as well as to perform various operations such as sorting, merging, and indexing.

Installation:

Samtools can be downloaded from the Samtools website: http://www.htslib.org/download/. You can download the appropriate package for your operating system from the website. Once you have downloaded the package, you can follow the installation instructions to install it on your computer. Similar to HiSAT2, we can install Samtools with conda.

```
Unset
conda install -c bioconda samtools
```

MountainClimber

MountainClimberTU is a component of the MountainClimber tool (https://github.com/gxiaolab/mountainClimber) that is designed to identify transcription units (TUs) from RNA-Seq data. It employs a de novo approach to call TUs independently for each sample. The tool generates a set of TUs for each sample, which can then be used for downstream analysis such as identifying alternative transcription start sites (ATS) and alternative polyadenylation sites (APA).

Installation:

To use mountainClimber, we can clone the package from the Github repository using the command:

```
Unset
git clone https://github.com/gxiaolab/mountainClimber
```

Note that MountainClimber has Dependencies include: python (v2.7.2), scipy (v0.15.1), numpy (v1.10.4), peakutils (v1.0.3), sklearn (v0.18.1), pysam (v0.9.0), pybedtools (v0.6.2), bisect, itertools. These packages have been preinstalled in the conda env_mtnclb.yml.

sklearn

Scikit-learn is a popular Python library for machine learning that provides various algorithms and tools for data preprocessing, feature engineering, model selection, and evaluation. It is built on top of NumPy, SciPy, and matplotlib and is designed to work seamlessly with these libraries.

Installation:

Since we use conda to manage the packages, we can directly install sklearn using conda.

```
Unset
conda install -c anaconda scikit-learn
```

Running the Pipeline

1. Create conda environment SRA from yml:
This step involves creating a conda environment named "SRA" using the configuration file provided in a YAML format. The YAML file contains a list of packages and their versions required for running the SRA pipeline.

```
Unset
conda env create -f env_sra.yml
```

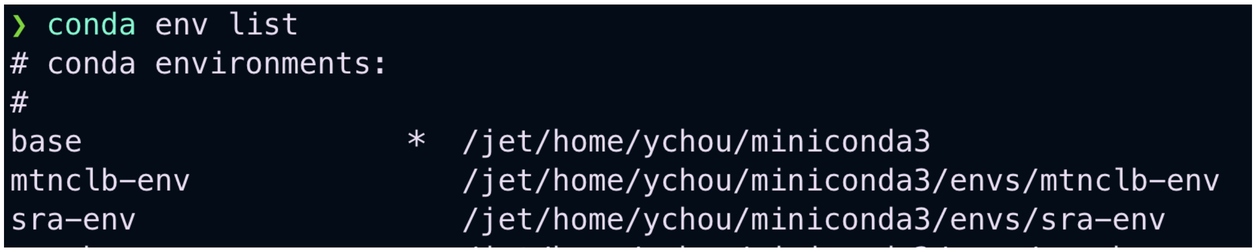
2. Create conda environment mtnclb from yml:
Similar to the first step, this step involves creating another conda environment named "mtnclb" using the configuration file provided in a YAML format. The YAML file contains a list of packages and their versions required for running the mtnclb pipeline.

```
Unset
conda env create -f env_mtnclb.yml
```

After the creation of conda environments. Run the following command to check if it is successful.

```
Unset
conda env list
```

This command would print out all the existing conda environments. You should get the result like this:


```
❯ conda env list
# conda environments:
#
base                  *  /jet/home/ychou/miniconda3
mtnclb-env               /jet/home/ychou/miniconda3/envs/mtnclb-env
sra-env                  /jet/home/ychou/miniconda3/envs/sra-env
```

3. Running an interactive session:

Before running the shell scripts, you need to launch an interactive session. For example,

```
Unset
interact -p RM-shared --ntasks-per-node=2 -t 08:00
```

would create a interactive session using the RM-shared partition, with 2 core and run for 8 hours.

4. Activate SRA: Once the SRA environment is created, it needs to be activated to use it. This step can be achieved by running the command:

```
Unset
conda activate sra-env
```

5. Run Shell Scripts:

SRA_download_fastq.sh

This script is used to download fastq files from the NCBI SRA database using the SRA toolkit. The script creates an output directory named "fastq" under the "data/SRA" directory. It then loops through a set of SRR IDs, generates file names for each ID, and checks if the fastq files already exist for that ID. If the fastq files do not exist, the script runs fastq-dump with specific arguments to download and compress the files. If the files already exist, the script skips the download step for that ID. The output files are written in the "fastq" directory, and the naming convention for the files is based on the SRR ID.

SRA_process.sh

This script performs the alignment of the SRA fastq files to the reference genome. Firstly, it downloads and unzips the reference genome if it does not already exist. It then builds the HISAT2 index for the reference genome. The script then proceeds to locate all the SRA fastq files in the specified directory, runs HISAT2 alignment on each SRA accession, and outputs the resulting SAM files to a new directory. The script then converts the SAM files to BAM format, sorts the BAM files, and outputs the resulting sorted BAM files to a new directory. If the BAM files already exist in the specified directory, the script skips this step. Finally, the script prints a message indicating that the processing is complete.

6. Conda deactivate: After running the shell script, it is recommended to deactivate the conda environment to avoid any conflicts with other environments or packages. This step can be achieved by running the command "conda deactivate" in the terminal or command prompt.

   Activate environment mtnclb: After deactivating the SRA environment, the mtnclb environment needs to be activated to use it. This step can be achieved by running the command:

```
Unset
conda activate mtnclb-env
```

The reason we need to create a new conda environment for the MountainClimber is because MountainClimber requires a usage of Python version=2.7, which is not compatible with the packages in the SRA environment

7. Run shell script **MC_process.sh**: This step involves running another shell script named "MC_process.sh". The script sorts the junction files and bedgraph files generated in the previous steps and then runs the MountainClimberTU.py script for each sample to generate transcriptional units.

8. Run python script **gen_embedding.py**: This script will generate embeddings for the transcriptional unit information to be used as input for the following ML classification model. In detail, this script reads in a set of bed files containing genomic intervals indicating the positions for the transcription units, aggregates them at a specified resolution, performs principal component analysis (PCA) on the resulting data for each

chromosome separately to reduce the feature dimension, and saves the resulting PCA components as a numpy array for future analysis. The script uses the numpy and pandas libraries for data manipulation, the argparse library for handling command line arguments, and the scikit-learn library for PCA.

9. Run python script **predict.py**: This script utilizes the feature extracted from the previous step to train ML classification to predict the disease states. In detail, it takes in command-line arguments (feature file, label file, model name, and number of splits for K-fold cross-validation), provides options for training different machine learning models (logistic regression with L1 penalty, logistic regression, SVM, or random forest) using the specified feature and label data, evaluates the model's accuracy and F1 score and create a confusion matrix to further analysis the performance.