

User Manual

NanoTrans: an integrated computational framework for comprehensive transcriptome analysis with Nanopore direct-RNA sequencing

Release v1.0.0

(2024-04-22)

Correspondence Contact:

Jia-Xing Yue (岳家兴)

Email: yuejiaxing[AT]gmail[DOT]com

GitHub: yjx1217

Twitter: iamphioxus, evomicslab

Website: <http://evomicslab.org/>

Table of Contents

[Introduction 1](#_Toc155875770)

[Citation 1](#_Toc155875771)

[License 2](#_Toc155875772)

[Release history 2](#_Toc155875773)

[Software Installation and configuration 3](#_Toc155875774)

[Installation 3](#_Toc155875775)

[What’s Inside 5](#_Toc155875776)

[Expected input data 5](#_Toc155875777)

[Pipeline Design 6](#_Toc155875778)

[The Testing Example Walking Through 7](#_Toc155875779)

[The NanoTrans Installation 7](#_Toc155875780)

[Running analysis with NanoTrans 8](#_Toc155875781)

# **Introduction**

Nanopore direct-RNA sequencing (DRS) provides the direct access to native RNA strands with full-length information, shedding light on rich qualitative and quantitative properties of gene expression profiles. Here with NanoTrans, we present an integrated computational framework that comprehensively covers all major DRS-based application scopes, including isoform clustering and quantification, poly(A) tail length estimation, RNA modification profiling, and gene fusion detection. In addition to its merit in providing such a streamlined one-stop solution, NanoTrans also shines in its workflow-orientated modular design, batch processing capability, rich tabular and graphic report outputs, as well as automatic installation and configuration support. Given the rising adoption of Nanopore DRS technology in the field, we believe NanoTrans will become a highly useful tool to help researchers to fully explore the power of this exciting technology with rich biological insights obtained.

Under the hood, a series of task-specific modules are provided to carry out the full workflow of NanoTrans:

* **00. Reference\_Genome**

donwloading and preprocessing the reference genome and annotation

* **00. Long\_Reads**

performing basecalling and length/quality summarization of raw Nanopore DRS fast5 reads

* **01. Reference\_Genome\_based\_Read\_Mapping**

mapping the nanopore DRS reads against the reference genome

* **02. Isoform\_Clustering\_and\_Quantification**

clustering and polishing isoforms and quantifying their expression levels

* **03. Isoform\_Expression\_and\_Splicing\_Comparison**

comparing isoform usages and splicing preferences among different sample groups or samples

* **04. Isoform\_RNA\_Modification\_Identification**

identifying RNA modification profile of each isoform

* **05. Isoform\_PolyA\_Tail\_Length\_Profiling**

profiling poly(A) tail length of each isoform

* **06.Gene\_Fusion\_Detection**

identifying gene fusion based on the chimeric isoform evidence

* **07.Report**

summarizing the major results into one HTML

# **Citation**

Ludong Yang, Xinxin Zhang, Fan Wang, Li Zhang, Jing Li, Jia-Xing Yue. (2024) NanoTrans: An integrated computational framework for comprehensive transcriptome analyses with Nanopore direct-RNA sequencing. ***BioRxiv***, (doi: <https://doi.org/10.1101/2022.11.29.518309>)

# **License**

NanoTrans itself is distributed under the MIT license but some of its dependencies might have more strict license for commercial use. Please check the licensing details of those dependencies.

# **Release history**

* v0.0.1 Released on 2022/11/30
* v0.0.2 Released on 2024/01/10
* v1.0.0 Released on 2024/04/22
  + Module 00: [new] Add a new bash script into Module 00 to support basecalling with Dorado; Module 05: [improvement] Cancel to specify basecalling sequencing summary for nanopolish index when using Dorado as basecalling tool.
  + Module 02, 04, 05: [improvement] Support csi format index for BAM to adapt large chromosomes.
  + Module 03: [improvement] Restore the “master\_sample\_table” argument to make comparisons easier; optimize plots.
  + Module 04: [improvement] Compress large files to reduce disk usage; [fix] Skip query isoforms that do not exist in the reference.
  + Module 07: [fix] Specify the R PATH used in installation step to fix the conflict of different versions of R.
  + Installation: [fix] Add Java to the ENV PATH to fix the java version bug when identifying fused genes; [improvement] Install tools from local by replacing the URL with a local path.

**Software Prerequisites**

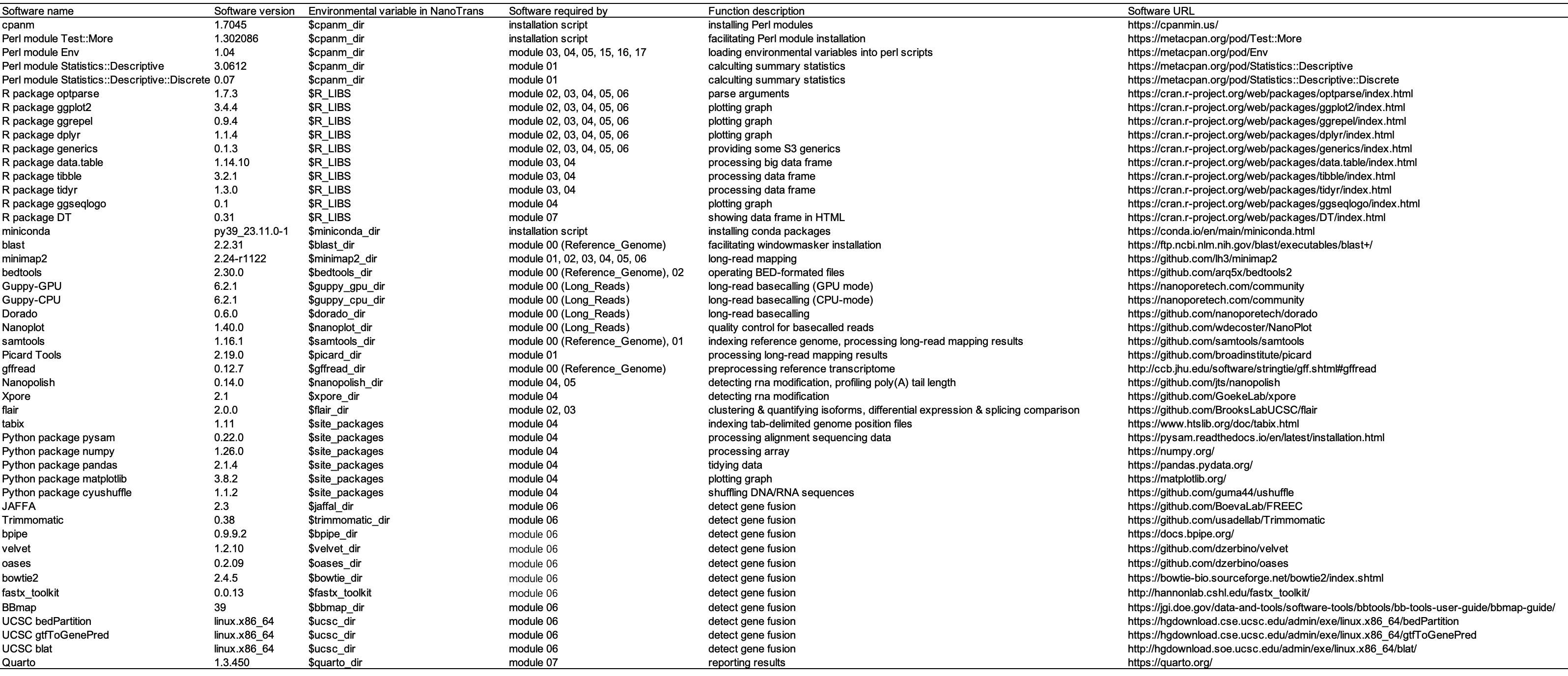
NanoTrans is designed for a desktop or computing server running an x86-64-bit Linux operating system. Multithreaded processors are preferred to speed up the process since some time-consuming steps can be configured to use multiple threads in parallel. A stable internet connection is required for its installation. Due to the large file size of Nanopore raw fast5 reads (~200-300 GB per MinION flowcell run), a minimal of 4TB hard disk storage size is strongly recommended. A number of standard Linux software compilation prerequisites are listed as below.

* bash (https://www.gnu.org/software/bash/)
* bzip2 and libbz2-dev (http://www.bzip.org/)
* gcc and g++ (https://gcc.gnu.org/)
* git (https://git-scm.com/)
* GNU make (https://www.gnu.org/software/make/)
* gzip (https://www.gnu.org/software/gzip/)
* libopenssl-devel
* libcurl-devel
* java runtime environment (JRE) v1.8.0 (https://www.java.com)
* perl v5.12 or newer (https://www.perl.org/)
* tar (https://www.gnu.org/software/tar/)
* unzip (http://infozip.sourceforge.net/UnZip.html)
* wget (https://www.gnu.org/software/wget/)
* zlib and zlib-devel (https://zlib.net/)
* xz and xz-devel (https://tukaani.org/xz/)
* perl-devel (https://pkgs.org/download/perl-devel)
* R 3.6 or newer (https://www.r-project.org/)

# **Software Installation and configuration**

In addition to the system-level prerequisites described above, NanoTrans relies on a number of third-party bioinformatics tools for data analysis, all of which can be automatically installed and configured by NanoTrans. A bash script (“install\_dependencies.sh”) is pre-shipped with NanoTrans to perform such installation and configuration. A detailed list of these third-party tools and their underlying functions in NanoTrans are listed below (Table 1).

***Table 1. Description of third-party software packages that will be downloaded and installed during NanoTrans’ automatic installation.***



# **Installation**

NanoTrans is implemented in Bash, Perl, Python, and R. It is designed for a desktop or computing server running an x86-64-bit Linux operating system. Multithreaded processors are preferred to speed up the process since many steps can be configured to use multiple threads in parallel. A stable internet connection is required for its installation. Note that due to the large file size of Nanopore raw fast5 reads (~200-300 GB per MinION flowcell run), a minimal of 2TB hard disk storage size is strongly recommended.

git clone https://github.com/yjx1217/NanoTrans.git  
cd NanoTrans  
bash ./install\_dependencies.sh

If the installation succeeds, you should see the following message:

“NanoTrans message: This bash script has been successfully processed! :)”

This signifies the success of the installation process.

Upon the success of the installation, a subdirectory named build and a file named env.sh will be generated. The build subdirectory holds all the installed dependencies, while the env.sh file contains the execution paths of these dependencies. This file will be automatically loaded to set up the working environment for NanoTrans’ various modules. The base directory of NanoTrans is defined as $NANOTRANS\_HOME in this file.

If unexpected error occurs during installation, normally you can just re-do “bash ./install\_dependencies.sh” step and the installation should be able to automatically resume from the previous interruption point.

# **What’s Inside**

Inside the downloaded NanoTrans directory, you should see the following file structure (Figure1).

***Graphical user interface

Description automatically generated***

***Figure 1. Overview of the NanoTrans directory system.*** *All top-level directories (boxes, solid lines) and individual files of NanoTrans are listed and briefly described. Additional directories and files will be generated during the installation and execution of NanoTrans (boxes, dashed lines).*

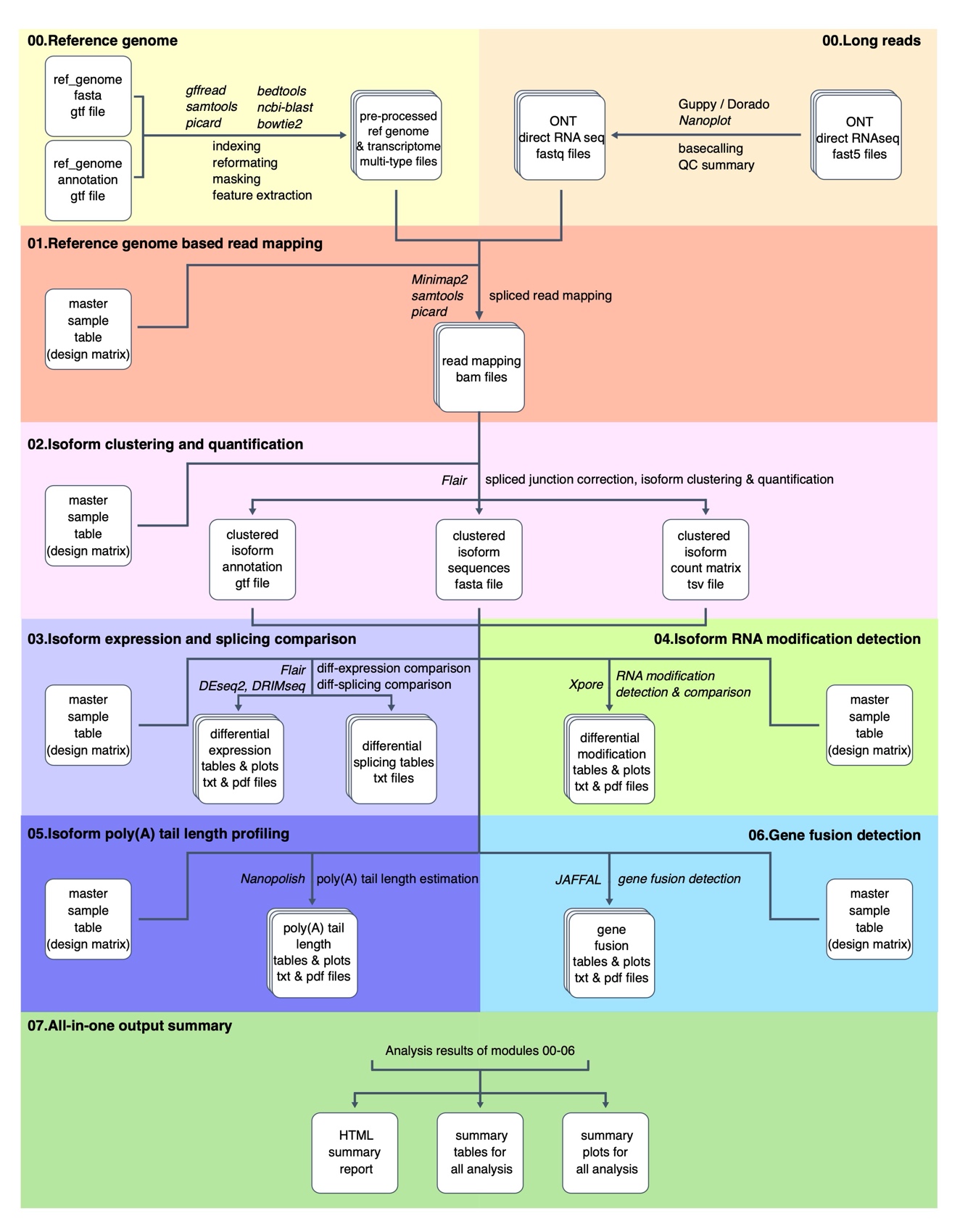
# **Expected input data**

NanoTrans expects the following inputs data:

1. one or more subdirectories containing raw Nanopore fast5 reads (each subdirectory corresponds to one sample/replicate) as the input data.
2. A master sample table in which the sample id, comparison group, replicate id, as well as the relative path to its associated nanopore reads.

# **Pipeline Design**

The design of NanoTrans is workflow-orientated, with a series of task-specific modules numbered according to their processing order (Figure 2). Briefly, NanoTrans first performs Nanopore reads basecalling and reference genome preprocessing with its two starting modules numbered with “00”. The basecalling step here can be processed either in GPU or CPU mode. Regarding the reference genome setup, NanoTrans supports all organisms with reference genome and annotation retrievable via Ensembl [(https://www.ensembl.org](file:///Users/yjx/Projects/2022_NanoTrans/(https:/www.ensembl.org)) or its sister sites (e.g., Ensembl Fungi, Ensembl Plants, Ensembl Protists, and Ensembl Metazoa). The basecalled fastq reads are subsequently mapped to the preprocessed genome in a splicing-aware manner (module “01”), after which isoform clustering and quantification are further performed accordingly (module “02”). Based on the clustered and quantified isoforms, NanoTrans can perform different application-specific analyses such as isoform expression and splicing comparison (module “03”), isoform RNA modification identification (module “04”), isoform poly(A) tail length profiling (module “05”). In addition, NanoTrans can also apply reference-based gene fusion detection (module “06”). Finally, an HTML containing major results can be generated by module “07”. For all these applications, a user-defined master sample table is used to specify sample information and experimental design, based on which automatic batch processing and between group comparison are natively supported, which comes handy for large-scale analysis with many samples.



***Figure 2. An overview of the NanoTrans framework.*** *Starting with raw fast5 reads, NanoTrans automates the full workflow of DRS data analysis and covers a wide range of applications settings including isoform clustering and quantification, differential expression and splicing examination, RNA modification identification, poly(A) tail length profiling, and gene fusion detection. The names of third-party tools employed in each step are denoted in italics.*

# **The Testing Example Walking Through**

**The NanoTrans Installation**

1. **Downloading and installing NanoTrans**

Run this step by typing:

git clone <https://github.com/yjx1217/NanoTrans.git>  
cd NanoTrans  
bash ./install\_dependencies.sh

**[Important Note]**

Please note that it will take a while for the installation to finish. Therefore, it is recommended to run the bash script above with nohup, which prevents the unintended interruption of the running script. The same trick applies to all the other module-specific bash script as well.

nohup bash ./install\_dependencies.sh > run.log.txt 2>&1 &

Please note if the installation script prompts for the following message at the end of the installation process:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Your java version is not the version required by NanoTrans (java v1.8)!

Please manually set the directory path to java 1.8 executable on the last line of the env.sh file generated by this installation script!"

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

If this message is prompted, please manually modify the last line of the env.sh file to provide the path to the java 1.8 executable accordingly after the installation process successfully finishes.

If the installation succeeds, you should see the following message:

“NanoTrans message: This bash script has been successfully processed! :)”

This signifies the success of the installation process. The same is true for all module-specific bash scripts (named as “NanoTrans.\*.sh”) of NanoTrans.

Upon the success of the installation, a subdirectory named build and a file named env.sh will be generated. The build subdirectory holds all the installed dependencies, while the env.sh file contains the execution paths of these dependencies. This file will be automatically loaded to set up the working environment for NanoTrans’ various modules. The base directory of NanoTrans is defined as $NANOTRANS\_HOME in this file.

In case of installation failure (most likely due to internet connection problem that might occur temporarily), the users only need to re-run the installation script install\_dependencies.sh. NanoTrans will automatically detect the previous interruption point and resume the installation process.

**Major outputs when running this step:**

build

# The subdirectory holding all the installed dependencies.

env.sh

# The file containing the execution paths of these dependencies.

**[Important Note]**

Although we made the effort to cover as much as technical details in our pre-shipped the installation script. Unexpected installation errors might still be encountered by some users due to server-specific problems. In that case, it is possible to skip the installation of certain problematic tool with NanoTrans while using a separately installed copy instead. To do this, just mask out (by putting “#” at the beginning of the corresponding lines) the installation commands in the install\_dependencies.sh script. In this case, by running the install\_dependencies.sh script, NanoTrans will still install other third-party dependencies and generated the env.sh file as usual. The only difference is that in the resulting env.sh file, the environmental variable to the skipped tool will have blank value. Users just need to manually add the accessible path (the full directory to the corresponding executable) to a separately installed copy of the skipped tool into the env.sh file, so that NanoTrans can correctly locate and summon the separately installed tool by loading env.sh file. This should work in most cases. If users still have installation problems, please do not hesitate to report the issue via GitHub’s issue ticket system (<https://github.com/yjx1217/NanoTrans/issues>), so that we can provide one-to-one help.

**Running analysis with NanoTrans**

1. **Creating a NanoTrans project directory**

Copying the Project\_Template directory to create your own NanoTrans project directory. Here we will name it as Project\_Example for this testing example. Once created, enter into this directory.

Run this step by typing:

cp -r Project\_Template Project\_Example  
cd Project\_Example

1. **Setting up the reference genome.**

For the testing example, we are going to use the *Arabidopsis* reference genome (version: TAIR10). At this step, the *Arabidopsis* reference genome assembly and annotation will be automatically downloaded from Ensembl ([https://www.ensembl.org](https://www.ensembl.org/)) and be properly set up.

Run this step by typing:

cd 00.Reference\_Genome

bash NanoTrans.00.Reference\_Genome\_Preprocessing.sh

NanoTrans supports all organisms with reference genome and annotation retrievable via Ensembl or its sister sites (e.g., Ensembl Fungi, Ensembl Plants, Ensembl Protists, and Ensembl Metazoa). This provided bash script is a general template of downloading and setting up the reference genome for any given organisms. You can adapt it for your own project by specifying the assembly and annotation downloading URLs of the organisms that want.

In NanoTrans, there is a dedicated section for customized parameter setting at the beginning of each module-specific bash script. In general, you only need to modify this part to adapt the script for your own project. For example, Figure 3 shows how you can adapt the provided NanoTrans.00.Reference\_Genome\_Preprocessing.sh script template for human-based analysis.

Text

Description automatically generated with medium confidence

*Figure 3. An example of NanoTrans’ customizable parameter setting section in the module-specific bash script. All such module-specific bash script has such a section for users to specify input files and customizable parameters.*

**Major outputs when running this step for the testing example:**

ref.genome.fa

# The preprocessed reference genome file in FASTA format.

ref.genome.gtf

# The preprocessed annotation file in GTF format.

ref.transcriptome.fa

# The preprocessed reference transcriptome file in FASTA format.

ref.transcript2gene\_map.txt

# The preprocessed ID mapping file among transcript IDs, gene IDs, and gene names.

1. **Setting up the raw Nanopore fast5 reads.**

At this step, we are going to set up the raw Nanopore DRS fast5 reads for basecalling. For this testing example, we will download the reads generated in BioProject PRJEB32782. The preferred sample IDs and the corresponding reads download URLs are provided in Table 2.

Table 2. Sample and reads information for the testing example.

|  |  |
| --- | --- |
| Sample ID | Download URL for the raw DRS fast5 reads |
| vir1.rep1 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764352/vir1_nanopore_drs_1.tar.gz> |
| vir1.rep2 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764353/vir1_nanopore_drs_2.tar.gz> |
| vir1.rep3 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764354/vir1_nanopore_drs_3.tar.gz> |
| vir1.rep4 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764355/vir1_nanopore_drs_4.tar.gz> |
| VIRc.rep1 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764356/VIRc_nanopore_drs_1.tar.gz> |
| VIRc.rep2 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764357/VIRc_nanopore_drs_2.tar.gz> |
| VIRc.rep3 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764358/VIRc_nanopore_drs_3.tar.gz> |
| VIRc.rep4 | <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764359/VIRc_nanopore_drs_4.tar.gz> |

For each sample, place the downloaded tar.gz file to the  
Project\_Example/00.Long\_Reads/raw\_fast5 subdirectory and apply file decompression and renaming. For instance, for the sample vir1.rep1, this can be done by:

cd Project\_Example/00.Long\_Reads/raw\_fast5

wget -c <ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR376/ERR3764352/vir1_nanopore_drs_1.tar.gz>

**[Important Note]**

1. Please avoid the “-“ or “\_” character in sample ID. Only the alphabet character, number, and “.” is permitted.
2. Renaming the original read folder name (e.g., “vir1\_nanopore\_dirs\_1”) to be the same as the sample ID (e.g., “vir1.rep1”) is important for downstream analysis.

Once this is set up, edit the NanoTrans.00.Nanopore\_Reads\_Preprocessing.sh script to specify the basecalling parameters. There are many parameters in this file but the ones that you need to adjust are only the following:

1. gpu\_run\_mode # The default value “cpu” will work fine for all projects. But if your computing server has GPU and CUDA (<https://developer.nvidia.com/cuda-zone>) support. It is recommended to set this to “gpu” to speed up the basecalling process. And in this case, you will also need to adjust the default values for

gpu\_bin\_path, gpu\_lib\_path, and gpu\_include\_path based on your own CUDA environment.

1. sample\_id
2. flowcell\_version
3. sequencing\_kit\_version

For our testing example, Figure 4 shows how the basecalling parameters should be set for the sample vir1.rep1 (Figure 4).

Graphical user interface, text, application, email

Description automatically generated

*Figure 4. The customizable parameter setting section in the NanoTrans.00.Nanopore\_Reads\_Preprocessing.sh bash script.*

Once this is all set. Run the bash script to perform basecalling and quality-control summarization.

cd 00.Long\_Reads

bash NanoTrans.00.Nanopore\_Reads\_Preprocessing.sh

Upon finishing, the basecalled reads in fast5 format will be placed under:  
Project\_Example/00.Long\_Reads/basecalled\_fast5/<sample\_id>

The basecalled reads in fastq format will be place under:

Project\_Example/00.Long\_Reads/basecalled\_fastq/<sample\_id>

The quality control summary report for the basecalled reads will be place under:

Project\_Example/00.Long\_Reads/basecalled\_summary/<sample\_id>

You can repeat this process for all the samples.

#======================================#

**For POD5 reads**

#======================================#

While Oxford Nanopore Technology is developing a new basecalling tool, Dorado, to optimize basecalling with POD5 reads under a GPU environment, we have added an optional script, NanoTrans.00.Nanopore\_Reads\_Preprocessing\_Dorado.sh, to support this. For FAST5 reads, it is strongly recommended that users perform basecalling with NanoTrans.00.Nanopore\_Reads\_Preprocessing.sh, as Dorado is still frequently updated and optimized for POD5 reads.

**[Important Note]**

1. Please be advised that Dorado no longer generates basecalled fast5; consequently, Module 04 (RNA modification) cannot be executed.

The preparation of samples is identical to that previously described. Once this is set up, edit the NanoTrans.00.Nanopore\_Reads\_Preprocessing\_Dorado.sh script to specify the basecalling parameters. There are many parameters in this file but the ones that you need to adjust are only the following:

1. dorado\_run\_device # The default value “cpu” will work fine for all projects. But if your computing server has GPU and CUDA (<https://developer.nvidia.com/cuda-zone>) support. It is recommended to set this to “gpu” to speed up the basecalling process. And in this case, you will also need to adjust the default values for

gpu\_bin\_path, gpu\_lib\_path, and gpu\_include\_path based on your own CUDA environment.

1. sample\_id
2. decode\_models\_name # The RNA model name used for basecalling. For more details, please to see: https://github.com/nanoporetech/dorado?tab=readme-ov-file#rna-models. Default = "rna002\_70bps\_hac@v3".

Once this is all set. Run the bash script to perform basecalling and quality-control summarization.

cd 00.Long\_Reads

bash NanoTrans.00.Nanopore\_Reads\_Preprocessing\_Dorado.sh

Upon finishing, the basecalled reads in fastq format will be place under:

Project\_Example/00.Long\_Reads/basecalled\_fastq/<sample\_id>

The quality control summary report for the basecalled reads will be place under:

Project\_Example/00.Long\_Reads/basecalled\_summary/<sample\_id>

1. **Mapping basecalled reads to the preprocessed reference genome.**

At this step, NanoTrans will map the reads of each sample to the reference genome. A mapping report of each sample will be summarized automatically.

NanoTrans use a space- or tab-delimited master sample table file (e.g., Master\_Sample\_Table.Batch\_Example.txt for the testing example, in which “Batch\_Example” is the specified batch\_id) to control all downstream analysis in a batch-by-batch fashion. Multiple samples from one or two comparison groups can be specified in a single master sample table file. Samples from different batches can be processed at the same time without interference. Such master sample table should contain 6 columns: sample\_id, comparison\_group, replicate\_id, basecalled\_fastq\_file, basecalled\_fast5\_dir, and note. The first 5 columns are mandatory, while the last column is only for user’s self-documentation. All lines started with “#” will be automatically ignored. For the sample id, this should be the same as the one that users used in the 00.Long\_Reads module for reads basecalling and QC summary.

**[Important Note]**

For specifying sample ID and comparison group, again, no “-“ or “\_” should be used. Only alphabet character, numeric numbers, and “.” can be used here. For replicate id, please naming different replicates as “rep1”, “rep2”, ….

For the testing example, NanoTrans has already prepared a sample master sample table file (Master\_Sample\_Table.Batch\_Example.txt) under the subdirectory for module 01-06, which can be used as the template when preparing the master sample table of your own project. Its content is as follows (Figure 5).

Text

Description automatically generated

*Figure 5. The master sample table for the testing example. The file is tab-/space delimited and the last column (i.e., “note”) is optional.*

For the testing example, run this step by typing:

cd 01.Reference\_Genome\_based\_Read\_Mapping

bash NanoTrans.01.Reference\_Genome\_based\_Read\_Mapping.sh

For your own project, please edit the script:

NanoTrans.01.Reference\_Genome\_based\_Read\_Mapping.sh and the master sample table file to adapt them to your own project.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the mapping results for the Batch\_Example.

Batch\_Example/all\_samples.coverage\_summary.txt

# The mapping summary file for all samples defined in Master Sample Table.

Batch\_Example/<sample\_id>

# The subdirectory containing the mapping results for the sample <sample\_id>.

1. **Performing isoform clustering and quantification.**

At this step, NanoTrans will perform isoform clustering and quantification by leveraging the DRS reads to reference mapping results of all samples defined in the master sample table.

For the testing example, run this step by typing:

cd 02.Isoform\_Clustering\_and\_Quantification

bash NanoTrans.02.Isoform\_Clustering\_and\_Quantification.sh

After this step, if you want to check and plot the isoform usage across all samples for a specific gene, you can do so by using the bash script NanoTrans.02.Plot\_Isoform\_Usage.sh pre-shipped here.

Edit this bash script to specify the batch\_id and query\_gene\_id, and then type:

bash NanoTrans.02.Plot\_Isoform\_Usage.sh

For your own project, please edit the task-specific bash script

NanoTrans.02.Isoform\_Clustering\_and\_Quantification.sh NanoTrans.02.Plot\_Isoform\_Usage.sh

as well as the master sample table file to adapt it to your own project.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the processing results for the Batch\_Example.

Batch\_Example/all\_samples\_combined

# The subdirectory containing the final results of this step. You should be able to find all major processing results of this step here and no need to check other subdirectories. The same principle holds for the module 02-06.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.flair\_all\_collapsed.isoforms.fa

# The FASTA sequence file for the clustered isoform set with all samples leveraged. Those isoforms whose IDs start with non-reference Gene/Transcript IDs are novel isoforms identified.

Batch\_Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.flair\_all\_collapsed.isoforms.fa.fai

# The index file for the clustered isoform set with all samples leveraged.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.flair\_all\_collapsed.isoforms.gtf

# The reference-genome-mapping-based GTF file for the clustered isoform set with all samples leveraged. Those isoforms whose IDs start with non-reference Gene/Transcript IDs are novel isoforms identified.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.flair\_all\_collapsed.isoforms.bed

# The reference-genome-mapping-based BED file for the clustered isoform set with all samples leveraged. Those isoforms whose IDs start with non-reference Gene/Transcript IDs are novel isoforms identified.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.flair\_all\_collapsed.isoforms.with\_productivity.bed

# The extended reference-genome-mapping-based BED file for the clustered isoform set with all samples leveraged. A final column is added in this file to show the productivity of the corresponding isoform with the following values: PRO (productive), PTC (premature termination codon, i.e., unproductive), NGO (no start codon), and NST (has start codon but no stop codon).

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.counts\_matrix.tidy.txt

# A tidy version of the tabular summary file contains the isoform quantification results for all samples. Those isoforms whose IDs start with non-reference Gene/Transcript IDs are novel isoforms identified.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.counts\_matrix.tsv

# A raw version of the tabular summary file contains the isoform quantification results for all samples. Those isoforms whose IDs start with non-reference Gene/Transcript IDs are novel isoforms identified.

Batch\_Example/all\_samples\_combined/query\_gene\_isoform\_usage

# The subdirectory containing the output plots (in PDF format) of the NanoTrans.02.Plot\_Isoform\_Usage.sh script.

Batch\_Example/all\_samples\_combined/query\_gene\_isoform\_usage/Batch\_Example.<query\_gene\_id>.isoform\_usage\_bars.pdf

# The isoform splicing structure plot for the <query\_gene\_id>.

Batch\_Example/all\_samples\_combined/query\_gene\_isoform\_usage/Batch\_Example.<query\_gene\_id>.isoform\_usage\_proportion.pdf

# The isoform usage plot for the <query\_gene\_id>.

Batch\_Example/<sample\_id>

# The subdirectory containing the intermediate results of the sample <sample\_id>. Normally no need to check these subdirectories.

For the testing example, after running the bash scripts NanoTrans.02.Isoform\_Clustering\_and\_Quantification.sh and NanoTrans.02.Plot\_Isoform\_Usage.sh , two plots reflecting the isoform usage of the specified query gene (Figure 6 & 7) will be generated under the following path:

Batch\_Example/all\_samples\_combined/query\_gene\_isoform\_usage.

Chart

Description automatically generated

*Figure 6. The isoform usage bar plot generated for the Arabidopsis gene POM1 (AT1G05850) with the testing example.*

Chart

Description automatically generated

*Figure 7. The isoform usage proportion plot generated for the Arabidopsis gene POM1 (AT1G05850) with the testing example.*

1. **Performing differential expression and splicing comparison between different comparison groups.**

At this step, NanoTrans will perform differential expression, splicing, and isoform usage comparison between different comparison groups if both groups have equal to or more than 2 replicates.

Regarding the differential splicing analysis, four types of alternative splicing events are examined with NanoTrans: intron retention (ir), alternative 3’ splicing (alt3), alternative 5’ splicing (alt5), and cassette exons (es).

For the testing example, run this step by typing:

cd 03.Isoform\_Expression\_and\_Splicing\_Comparison

bash NanoTrans.03.Isoform\_Expression\_and\_Splicing\_Comparison.sh

For your own project, please edit the script

NanoTrans.03.Isoform\_Expression\_and\_Splicing\_Comparison.sh

and the master sample table file to adapt it to your own project.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the processing results for the Batch\_Example.

Batch\_Example/Batch\_Example.experimental\_design.yml

# The experimental design summary for all the samples defined in the master sample table based on their comparison group and replicate belongings.

Batch\_Example/all\_samples\_combined

# The subdirectory containing the main processing results of this step. You should be able to find all major processing results of this step here and no need to check other subdirectories. The same principle holds for the module 02-06.

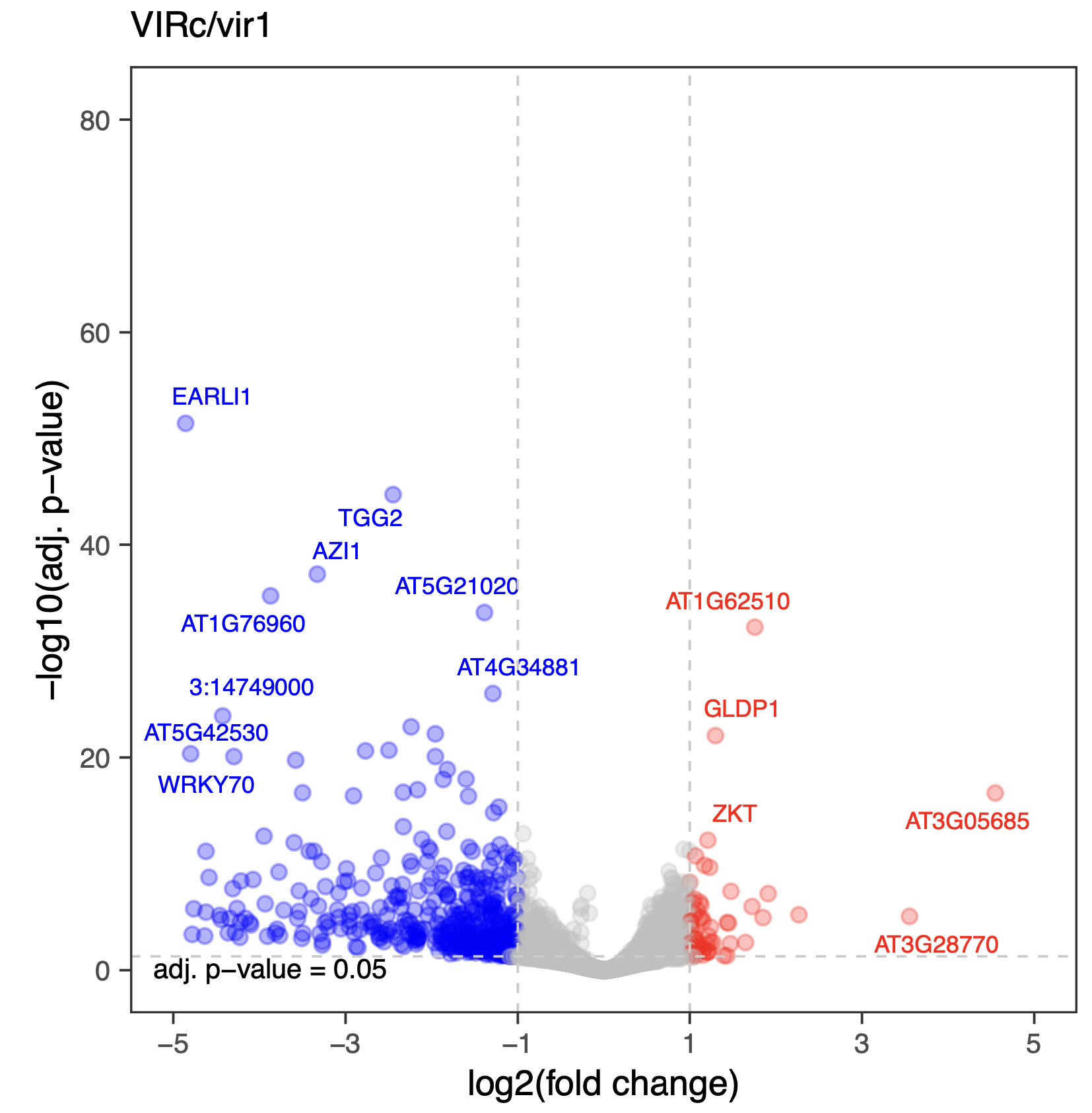
Batch\_Example/all\_samples\_combined/Batch\_Example\_differential\_expression\_output

# The subdirectory containing the final tabular results and volcano plots (when the number of replicates>=2) of the differential expression analysis.

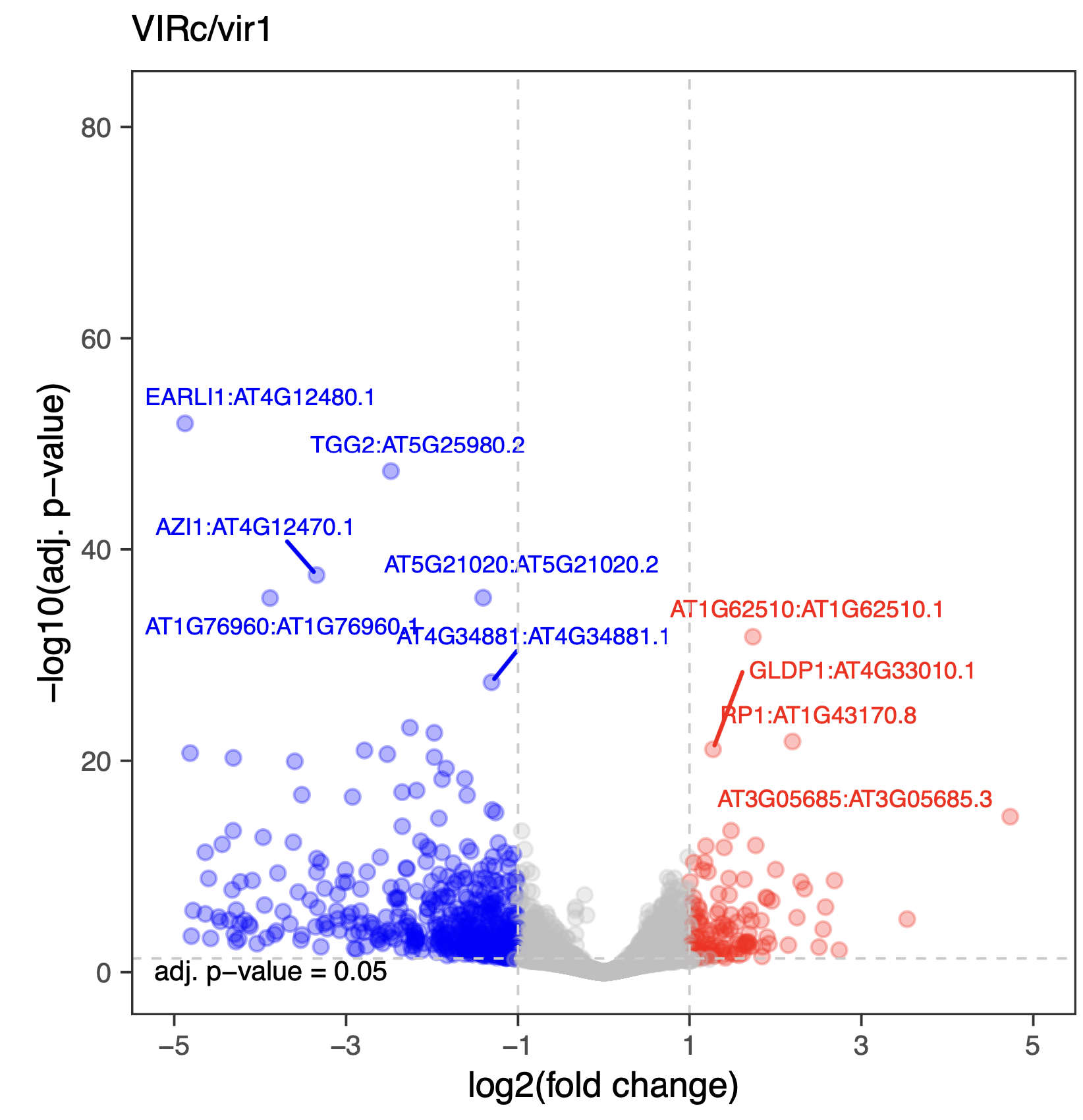
Batch\_Example/all\_samples\_combined/Batch\_Example\_differential\_splicing\_output

# The subdirectory containing the final tabular results of the differential splicing analysis.

For the testing example, after running the bash script NanoTrans.03.Isoform\_Expression\_and\_Splicing\_Comparison.sh, the volcano plots (Figure 8 & 9) for differentially expressed genes and isoforms will be generated under the following path: Batch\_Example/all\_samples\_combined/Batch\_Example\_differential\_expression\_output



*Figure 8. The differential gene expression volcano plot generated for the testing example.*



*Figure 9. The differential isoform expression volcano plot generated for the testing example.*

1. **Performing isoform RNA modification identification.**

At this step, NanoTrans will perform isoform RNA modification identification. Please note that this step requires two comparison groups: one control group and one testing group, so that the module will identify isoform-specific kmers (5-mers) with differential RNA modification rates between the two compared groups.

For the testing example, run this step by typing:

cd 04.Isoform\_RNA\_Modification\_Identification

bash NanoTrans.04.Isoform\_RNA\_Modification\_Identification.sh

After this step, if you want to check and plot the characteristics of differential RNA modifications, such as their distribution around stop codons, you can do so by using the bash script NanoTrans.04.Plot\_Characteristics\_RNA\_Modification.sh pre-shipped here.

For your own project, please edit the script  
NanoTrans.04.Isoform\_RNA\_Modification\_Identification.sh

NanoTrans.04.Plot\_Characteristics\_RNA\_Modification.sh  
and the master sample table file to adapt it to your own project.

**[Important Note]**

This step is computationally intensive, with substantial CPU time and disk space usage.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the processing results for the Batch\_Example.

Batch\_Example/Batch\_Example.experimental\_design.yml

# The experimental design summary for all the samples defined in the master sample table based on their comparison group and replicate belongings.

Batch\_Example/all\_samples\_combined

# The subdirectory containing the main processing results of this step. You should be able to find all major processing results of this step here and no need to check other subdirectories. The same principle holds for the module 02-06.

Batch\_Example/all\_samples\_combined/Batch\_Example.rna\_modification.diffmod.table.tidy.txt

# The tabular summary file for all differential RNA modification sites between the samples from the two comparison groups. This summary table is an extend version (with the gene\_id, and gene\_name columns attached) of the original output table of xpore (https://xpore.readthedocs.io/en/latest/outputtable.html)

Batch\_Example/all\_samples\_combined/Batch\_Example.rna\_modification. majority\_direction\_kmer\_diffmod.table.tidy.txt

# The tabular summary file for filtered differential RNA modification sites between the samples from the two comparison groups. The filtering is done by removing those kmers with their mod\_assignment values different from the majority assignment of the corresponding kmers, therefore restricting the analysis to one modification type per kmer. This summary table is an extend version (with the gene\_id, and gene\_name columns attached) of the original output table of xpore (https://xpore.readthedocs.io/en/latest/outputtable.html)

For the testing example, after running the bash script NanoTrans.04.Isoform\_RNA\_Modification\_Identification.sh, two plots will be generated under the following path: Batch\_Example/all\_samples\_combined/ to show top 20 kmer-gene combinations with most pronounced differential RNA modification rates (Figure 10).

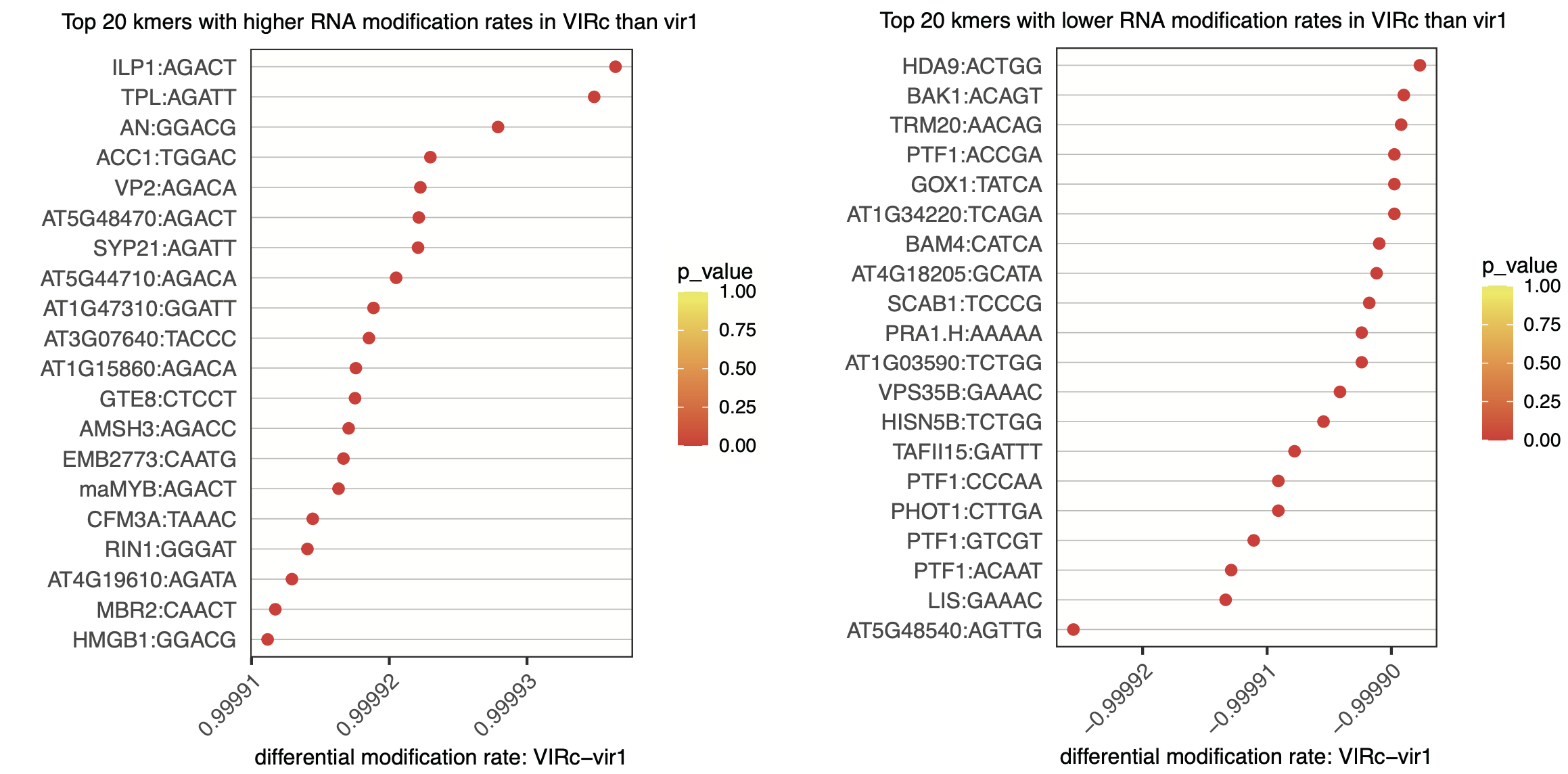


Figure 10. The top 20 kmer-gene combinations with higher (left) or lower (right) RNA modification rates in the VIRc group than the vir1 group.

For the testing example, after running the bash script NanoTrans.04. Plot\_Characteristics\_RNA\_Modification.sh, five plots will be generated under the following path: Batch\_Example/all\_samples\_combined/ to show characteristics of differential modified kmers (Figure 11).

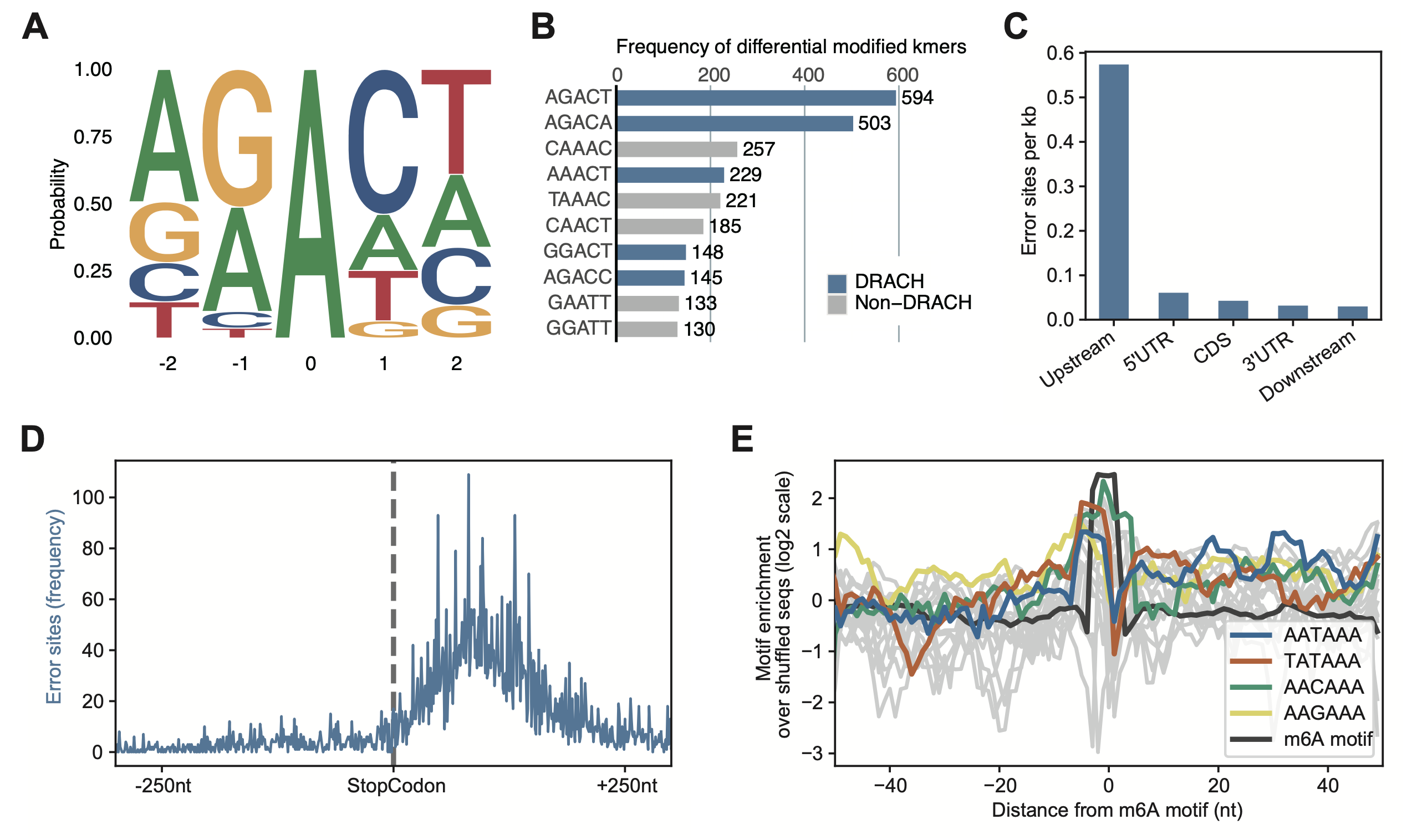


Figure 11. Characteristics of differentially modified kmers between the VIRc and vir1 groups. (A) The sequence logo showed the preference of filtered differentially modified kmers (DMKs, differential modification rates > 0.5, p-value < 0.05, and only kmers with center A were retained). (B) The top 10 DMKs with the highest frequency. (C) Distribution of the DMKs on distinct genomic features. (D) Frequency of DMKs around stop codons. (E) Overlaps between RNA modifications and poly(A) signal sites.

1. **Performing isoform poly(A) tail length profiling.**

At this step, NanoTrans will perform isoform poly(A) tail length profiling. For each isoform, the minimal, mean, median, and maximal poly(A) tail length will be estimated.

For the testing example, run this step by typing:

cd 05.Isoform\_PolyA\_Tail\_Length\_Profiling

bash NanoTrans.05.Isoform\_PolyA\_Tail\_Length\_Profiling.sh

For your own project, please edit the script  
NanoTrans.05.Isoform\_PolyA\_Tail\_Length\_Profiling.sh  
and the master sample table file to adapt it to your own project.

**[Important Note]**

This step is computationally intensive, with substantial CPU time.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the processing results for the Batch\_Example.

Batch\_Example/Batch\_Example.experimental\_design.yml

# The experimental design summary for all the samples defined in the master sample table based on their comparison group and replicate belongings.

Batch\_Example/all\_samples\_combined

# The subdirectory containing the final processing results of this step. You should be able to find all major processing results of this step here and no need to check other subdirectories. The same principle holds for the module 02-06.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.polya\_profiling.summary.txt

# The tabular report for poly(A) tail lengths summary statistics for all samples.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.polya\_profiling.median\_length.violin\_plot.pdf

# The violin plot for the median-based poly(A) tail length estimates of all transcripts from the two comparison groups. NanoTrans will generate this plot when there are two comparison groups defined in the master sample table.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.polya\_profiling.mean\_length.violin\_plot.pdf

# The violin plot for the mean-based poly(A) tail length estimates of all transcripts from the two comparison groups. NanoTrans will generate this plot when there are two comparison groups defined in the master sample table.

Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.polya\_profiling.median\_length.density\_plot.pdf

# The density plot for the median-based poly(A) tail length estimates of all transcripts from the two comparison groups. NanoTrans will generate this plot when there are two comparison groups defined in the master sample table.

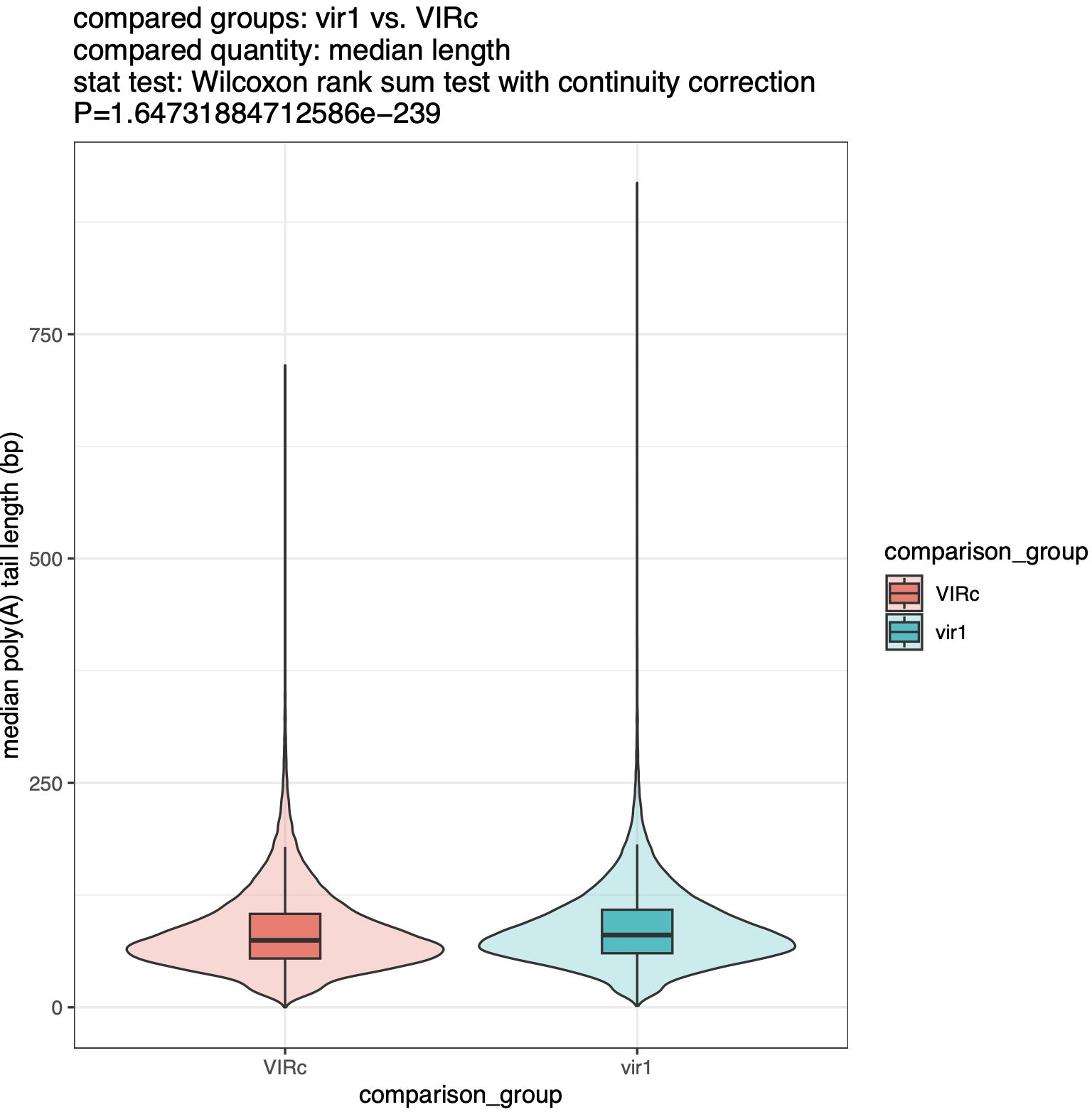
Batch\_Example/all\_samples\_combined/Batch\_Example.all\_samples\_combined.polya\_profiling.mean\_length.density\_plot.pdf

# The density plot for the mean-based poly(A) tail length estimates of all transcripts from the two comparison groups. NanoTrans will generate this plot when there are two comparison groups defined in the master sample table.

Batch\_Example/<sample\_id>

# The subdirectory containing the results for the sample <sample\_id>. Normally no need to check these files.

For the testing example, after running the bash script NanoTrans. 05.Isoform\_PolyA\_Tail\_Length\_Profiling.sh, six plots will be generated under the following path: Batch\_Example/all\_samples\_combined/ for visualizing the poly(A) tail length difference between the two comparison groups (Figure 12-15).



*Figure 12. The violin plot for the median-based poly(A) tail length estimates of all transcripts from the two comparison groups.*

Chart, line chart

Description automatically generated

*Figure 13. The density plot for the median-based poly(A) tail length estimates of all transcripts from the two comparison groups.*

Chart, diagram

Description automatically generated

*Figure 14. The violin plot for the mean-based poly(A) tail length estimates of all transcripts from the two comparison groups.*

Chart, line chart

Description automatically generated

*Figure 15. The density plot for the mean-based poly(A) tail length estimates of all transcripts from the two comparison groups.*

1. **Performing gene fusion detection.**

At this step, NanoTrans will perform gene fusion detection. The identified candidate gene fusion events will be reported in tabular list ranked by their confidence levels: “high confidence,” “low confidence,” and “potential trans-splicing” classes. Regarding these confidence levels, “high confidence” fusion is supported by two or more reads with breakpoints aligning to exon boundaries; “low confidence” fusions are also supported by two or more reads, but breakpoints do not align to exon boundaries; “potential trans-splicing” events are supported by a single read, with breakpoints aligning to exon boundaries. In general, you can focus on those events labeled as “high confidence”.

For the testing example, run this step by typing:

cd 06. Gene\_Fusion\_Detection

bash NanoTrans.06.Gene\_Fusion\_Detection.sh

For your own project, please edit the script  
NanoTrans.06.Gene\_Fusion\_Detection.sh  
and the master sample table file to adapt it to your own project.

**Major outputs when running this step for the testing example:**

Batch\_Example

# The subdirectory containing the processing results for the Batch\_Example.

Batch\_Example/Batch\_Example.experimental\_design.yml

# The experimental design summary for all the samples defined in the master sample table based on their comparison group and replicate belongings.

Batch\_Example/all\_samples\_combined

# The subdirectory containing the main processing results of this step. You should be able to find all major processing results of this step here and no need to check other subdirectories. The same principle holds for the module 02-06.

Batch\_Example/all\_samples\_combined/<sample\_id>.gene\_fusion.transcripts.txt

# The tabular report for gene fusion events identified in the sample <sample\_id>.

Batch\_Example/all\_samples\_combined/<sample\_id>.gene\_fusion.transcripts.fa

# The FASTA sequence file for fused transcripts identified in the sample <sample\_id>.

Batch\_Example/all\_samples\_combined/gene\_fusion\_plots

# The subdirectory containing plots for the gene fusion events identified in the sample <sample\_id>. For each identified potential gene fusion event, NanoTrans plots the genome arrangement of all transcripts associated with the left and right donor partners and highlights the corresponding genomic regions corresponding to the fused transcript.

Batch\_Example/<sample\_id>

# The subdirectory containing the gene fusion identification results for the sample <$sample\_id>. Normally no need to check these subdirectories as all important results have been recorded in the Batch\_Example/all\_samples\_combined subdirectory already.

For the testing example, after running the bash script NanoTrans. 06.Gene\_Fusion\_Detection.sh, a few potential gene fusion candidate events will be identified for each sample but all of them are labeled as “potential trans-splicing”, which do occur in non-tumor samples such as those used for our testing example.

1. **Summarizing the major results.**

At this step, NanoTrans will integrate major results generated by above steps into one HTML. Users can browse these results to obtain a preliminary understanding, such as volcano plots showing differentially expressed genes. Moreover, users can retrieve information on differential expression, differential isoform usage, etc., by entering the gene name of interest in the search box at the top right corner of the corresponding table.

For the testing example, run this step by typing:

cd 07.Report

bash NanoTrans.07.Report.sh

For your own project, please edit the script  
NanoTrans.07.Report.sh  
and the master sample table file to adapt it to your own project.

**Major outputs when running this step for the testing example:**

NanoTrans\_Report\_Batch\_Example.html

# The HTML containing major results generated by each module.