piPipes Manual

08-15-2014

piPipes is a set of pipelines developed in the Zamore Lab and ZLab to analyze piRNA/transposon from different Next Generation Sequencing libraries (including *small RNA-seq*, *RNA-seq*, *Genome-seq*, *ChIP-seq*, *CAGE/Degradome-Seq*).

Please see the main page for a brief introduction and Wiki pages (https://github.com/bowhan/piPipes/wiki) for detailed description for each pipeline.

piPipes installation and genome preparation

This document explains how to obtain **piPipes** from Github and how to install genome files.

To obtain piPipes

From Github

To clone the directory from Github. You will need to have `git` installed on your system. If not, you will need to download git here.

```
# enter your directory to store softwares
# the genome sequence and annotations will be stored under the piPipes directory
# so allow extra ~8.5 G for dm3 (fly), ~90 G for mm9 (mouse), ~131 G for hg19 (human)
git clone git@github.com:bowhan/piPipes.git
```

From release page

Alternatively, you can obtain **piPipes** from its release page. Note that you will not be able to easily make upgrades without `git`.

Set up

To make symbol links to **piPipes** main script, so that you can find **piPipes** without explicitly typing the absolute path:

```
# enter the piPipes directory
ln -s piPipes $HOME/bin/piPipes
ln -s piPipes_debug $HOME/bin/piPipes_debug
# if successfully done, when you type:
$ which piPipes
~/bin/piPipes
```

Other softwares

piPipes has most of the third-party tools pre-compiled and included in the `bin` directory. They will be automatically found when you run **piPipes**. To avoid mixing them with your own versions, we do not recommend to add `/piPipes/bin` to the `$PATH`.

However, there are some tools that we find them hard to ship so the user will need to install if haven't done so.

```
# 1. R
# Please follow instructions on http://www.r-project.org/ to install R
#! if successfully installed:
$ which Rscript
~/bin/Rscript
# ! Note the newer version of R has a different behavior for read.table ().
# Please use version earlier than R 3.1.0.
# http://stackoverflow.com/questions/22962917/barplot-failure-in-r-3-1-0-read-csv-converting-what-should-be-n
# Try to keep only one version of R in your system or $PATH
# Many of the "bugs" reported by our users were caused by multiple versions of R!

# FYI: in the installation pipeline, piPipes will try to install the following packages
# It would be nice if they are manually installed and confirmed
## from CRAN
```

```
RColorBrewer
ggplot2
ggthemes
gplots
multicore
scales
reshape
gridExtra
gdata
RCircos
## from Bioconductor
cummeRbund


# 2. HTSeq-count
# Please follow instructions on http://www-huber.embl.de/users/anders/HTSeq/doc/install.html
# to install HTSseq-count
#! if successfully installed:
$ which htseq-count
~/bin/htseq-count


# 3. MACS2
# Please follow instructions on https://github.com/taoliu/MACS/blob/master/INSTALL.rst
# to install MACS2
#! if successfully installed:
$ which macs2
~/bin/macs2


# 4. Perl Module Statistics::Descriptive; install it through
cpan Statistics::Descriptive
#! if successfully installed:
$ perl -MStatistics::Descriptive -e "print \"Installed.\\n\";"
Installed.
```

To update piPipes

```
# if you have git
# enter the piPipe directory
git pull
```

Reinstall (start from scratch)

```
# if you have git
# enter the piPipe directory
git reset --hard origin/master
```

To install genome

piPipes provides a uniform interface for different organisms/genomes. Due to the limit on file size of github, genome sequence and annotation files have to be downloaded separately. The user will need to perform an **installation** to download the files and prepare them for other pipelines to use.

To install a specific genome in one step:

```
piPipes install -g dm3    # fly genome dm3
piPipes install -g dm6    # fly genome new release, BDGP6
piPipes install -g mm9    # mouse genome mm9
piPipes install -g hg19   # human genome hg19
```

Many computing clusters only have internet access on the 'head node', which should only be used to submit jobs but not to run jobs. To separate downloading and preparation steps:

```
# under the "head" node: with internet access but no computing power
piPipes install -g dm3 -D
# finish the work under a computing node
piPipes install -g dm3
# Some steps take advantage of multiple CPUs, so providing more than one CPUs using ‘-c‘
# accelerates the installation process.
piPipes install -g dm3 -c 8
```

Notes:

- piPipes uses `wget --continue` so downloading will resume if the installation is disrupted.
- piPipes also only runs steps that haven't succeeded.
- During the installation, the user will be prompted to define the length of siRNAs and piRNAs. Our lab uses 20-22 nt for fly/mouse siRNA, 23–29 nt for fly piRNA and 23–35 for mouse piRNA. This information is stored in `common/dm3/variables` files and users can change the values manually later.

Genome Assembly Supported

Currently, *Drosophila melanogaster* and *Mus Musculus* piRNAs are the most well studied. **piPipes** is optimized for those two species (assembly version *dm3* and *mm9* from UCSC). For other organisms, due to either the relatively immature piRNA cluster annotation or the authors' poor knowledge, some functions may not be performed. But we really would like to cooperate with experts to make **piPipes** more generic in terms of the organism it supports. Please contact us if you would like to help.

File organization

All the files for a specific genome are stored under the `/path/to/piPipes/common/`. For example, fly files are stored under `/path/to/piPipes/common/dm3`. Most of them are in gziped BED format

dm3

piPipes downloads the annotation from iGenome, which misses the **chrU** and **X-TAS**. **piPipes** thus downloads **chrU.fa** from UCSC, and put X-TAS.fa in the Github repository.

For piRNA cluster annotation, **piPipes** uses the one from Brennecke, et al., Cell, 2007.

For transposons, **piPipes** uses two different annotations. *transposon* sequences are from flyBase and *repBase* sequences are from repBase. The *transposon* annotation has been used in the Zamore Lab since Li, et al., Cell, 2009. But the repBase annotation separated Long Terminal Repeat (LTR) of a retrotransposon from the middle part. So the LTR derived sequences do not become multi-mappers simply due to the presence of two LTR in a transposon sequence.

BDGP6 (Berkeley Drosophila Genome Project Release 6)

piPipes has incorporated the new assembly of fruitfly genome release 6.

```
# to install the new release
piPipes install -g dm6
```

Since it was just released (July 2014), iGenome or UCSC has not incorporated it. We used most of the annotation files from flyBase. Several notes:

1.piRNA cluster

Using the converter tool provided by flyBase, we tried to make the new annotation of piRNA clusters. However, 46 clusters cannot be successfully found in the new assembly, mostly due to **"maps to more than one scaffold".**

We now only keep the 96 ones that can be successfully mapped. But we are planning to use new data (higher depth) to annotate new clusters.

For more information, please read file `common/dm6/Brennecke.piRNAcluster.bed6.converted.failed`

2.Repeat Masker

We ran repeatMasker again using the following parameter to identify transposon site.

Note that by providing `-species drosophila`, we were using the transposon sequences from repBase instead of the sequences from flyBase.

```
# using flyBase transposon sequences
RepeatMasker \
    -pa 24 \
    -s \
    -low \
    -lib dmel-all-transposon-r6.01.fasta \
    -gff dmel-all-chromosome-r6.01.fasta \
    1> flyBase.stdout \
    2> flyBase.stderr
# using repBase
RepeatMasker \
    -pa 24 \
    -s \
    -low \
    -species drosophila \
    -gff dmel-all-chromosome-r6.01.fasta \
    1> repBase.stdout \
    2>repBase.stderr
```

3.GTF file

The gtf file obtained from flyBase `ftp://ftp.flybase.net/releases/FB2014_04/dmel_r6.01/gtf/dmel-all-r6.01.gtf.gz` cannot be correctly processed by `gtfToGenePred` from kent tools, due to the presence of "trans-splicing" of `mdg4`.

```
invalid gffGroup detected on line: 3R   FlyBase CDS 21375060    21375912    3.000000    -   0   gene_id "FBgn
GFF/GTF group FBtr0084081 on 3R+, this line is on 3R-, all group members must be on same seq and strand
# the rest trans-splicing ones include

FBtr0084079
FBtr0084080
FBtr0084081
FBtr0084082
FBtr0084083
FBtr0084084
FBtr0084085
FBtr0307759
FBtr0307760
```

We thus removed all the `mdg4` annotations.

```
grep -v mdg4
```

mm9

piPipes downloads the annotation from iGenome.

piPipes uses the piRNA cluster annotation from Li, et al., Mol Cell, 2013 and transposon annotation from repBase.

hg19

piPipes downloads the annotation from iGenome.

piPipes uses the piRNA cluster annotation from Rosenkranz, et al., BMC Bioinformatics, 2013 and transposon annotation from repBase.

other genomes

In order for **piPipes** to perform its full function on other genomes, the following steps should be completed:

1.Annotate piRNA cluster, provide it in BED format. Provide the sequence and name it `${GENOME}.piRNAcluster.fa`.

Run `proTRAC` or `piClust` to produce piRNA cluster annotation.

```
    Rosenkranz D and Zischler H. 2012. proTRAC--a software for probabilistic piRNA cluster detection,
visualization and analysis. BMC Bioinformatics 13: 5.
    Jung, I., Park, J. C. & Kim, S. piClust: A density based piRNA clustering algorithm.
Comput Biol Chem (2014).
```

2.Get gene structure annotations from UCSC table browser or through the mySQL interface. We have already included those files for many organisms in the `common` folder. If the folder already exist, no need to do this step.

```
# currently those genomes have been done for this step
bosTau7
rn5
danRer7
TARI10
hg19
mm9
dm3
```

3.Edit the `genomic_features` file under the genome folder (like *dm3* or *mm9*). See below.

4.The genome sequence should be provided and named as `$GENOME.fa`. **piPipes** builds **bowtie** index of the **genome sequence** for *small RNA pipeline*, **STAR** index for *RNA-seq and degradome pipeline* and **Bowtie2** index for *Genome-seq pipeline*.

5.The rRNA sequence should be provided and named as `rRNA.fa`. **piPipes** builds **bowtie** index of the **rRNA** for *small RNA*, **bowtie2** index for *normal RNA*.

6.The transposon consensus sequence should be provided and named as `${GENOME}.repBase.fa`. **piPipes** builds **bowtie** index of the **repBase/transposon/piRNA cluster** for *small RNA*.

Basic piPipes directory structure

```
|-- piPipes/ # top directory
|   |-- piPipes # main bash script to run
|   |-- piPipes_debug # main bash script to run, debug mode
|   |-- bin/ # binrary executables
|       |-- piPipes_smallRNA.sh # smallRNA seq pipeline, single sample mode
|       |-- piPipes_smallRNA2.sh # smallRNA seq pipeline, dual sample mode
|       |-- piPipes_RNASeq.sh # RNA-seq pipeline, single sample mode
|       |-- piPipes_RNASeq2.sh # RNA-seq pipeline, dual sample mode
|       |-- piPipes_DegradomeSeq.sh # Degradome-seq pipeline
|       |-- piPipes_ChIPSeq.sh # ChIP-seq pipeline, single sample mode
|       |-- piPipes_ChIPSeq2.sh # ChIP-seq pipeline, dual sample mode
|       |-- piPipes_GenomeSeq.sh # Genomic Seq pipeline
|       |-- ... # binaries like bowtie, STAR, cufflinks ...
|   |-- src/ # source codes
|       |-- bed2_to_bedGraph.cpp # piPipes source codes
|       |-- third_party/ # source codes of other tools; use this if the precompiled ones don't work
|       |-- ...
|   |-- common/ # where annotations and sequences been stored
|       |-- mm9/
|       |-- dm3/
|           |-- dm3.fa # genome sequence
|           |-- genomic_features # very important configuration file, see below
|           |-- Brennecke.piRNAcluster.bed6.gz # one the the annotation file, in bed format
|           |-- BowtieIndex/
|           |-- ...
|       |-- dm6/
|       |-- hg19/
|       |-- genome_supported.txt # storing the names of genome that has been installed
|       |-- RepBase19.02.fasta.tar.gz # transposon consensus sequences from repBase
|       |-- reformat_repBase_for_eXpress.sh # eXpress only takes the first token of Fasta name...
```

common folder

piPipes downloads annotations from **iGenome** **(UCSC version)**, which usually includes **genomic sequence (fasta)**, **rRNA (fasta)**, **transcriptome (gtf)** to be used by piPipes. **piPipes** includes the **repBase(fasta)** in the github for *dm3* and *mm9*. For other genomes, please retrieve the `repBase.fa` and name it `${GENOME}.repBase.fa` in the `common/${GENOME}` directory. For example, run:

```
# enter the directory unarchived from RepBase19.02.fasta.tar.gz
$ cat humrep.ref humsub.ref > ../hg19/hg19.repBase.fa
```

genomic features

piPipes includes a bunch of **genomic features (bed)** in the **genomic_features** file under the directory of each genome. Please also include them in the `common/${GENOME}` directory and add them in the `TARGET array` in `common/${GENOME}/genomic_features`. Follow the following example to set up:

```
# variables for small RNA pipeline intersecting
    MASK=$COMMON_FOLDER/UCSC.rRNA+tRNA+nonCoding.bed6.gz
    # tRNA, rRNA, nonCoding RNA (flyBase) from UCSC table browser
    piRNA_Cluster=$COMMON_FOLDER/Brennecke.piRNAcluster.bed6.gz
    # piRNA cluster defined in Brennecke, et al,. Cell, 2007; no strand information
    piRNA_Cluster_42AB=$COMMON_FOLDER/Brennecke.piRNAcluster.42AB.bed6.gz
    # 42AB
    piRNA_Cluster_20A=$COMMON_FOLDER/Brennecke.piRNAcluster.20A.bed6.gz
    # 20A
```

```
    piRNA_Cluster_flam=$COMMON_FOLDER/Brennecke.piRNAcluster.flam.bed6.gz
    # flam
    repeatMasker=$COMMON_FOLDER/UCSC.RepeatMask.bed
    # repeatMakser obtained from UCSC
    repeatMasker_IN_Cluster=$COMMON_FOLDER/UCSC.RepeatMask.inCluster.bed.gz
    # repeat masker identified region that fall into piRNA cluster
    repeatMasker_OUT_Cluster=$COMMON_FOLDER/UCSC.RepeatMask.outCluster.bed.gz
    # repeat masker identified region that fall outside piRNA cluster
    Trn=$COMMON_FOLDER/Zamore.transposon.bed.gz
    # transposon region used in Li, et al., Cell, 2009. More conserved than repeat masker
    Trn_IN_Cluster=$COMMON_FOLDER/Zamore.transposon.inCluster.bed.gz
    # transposon region in cluster
    Trn_OUT_Cluster=$COMMON_FOLDER/Zamore.transposon.outCluster.bed.gz
    # transposon region out cluster
    Trn_GROUP0=$COMMON_FOLDER/Zamore.transposon.group0.bed.gz
    # transposons that failed to pass threshold in Li, et al., Cell, 2009.
    # More conserved than repeat masker
    Trn_GROUP1=$COMMON_FOLDER/Zamore.transposon.group1.bed.gz
    # group 1 transposon in Li, et al., Cell, 2009, mainly germline
    Trn_GROUP2=$COMMON_FOLDER/Zamore.transposon.group2.bed.gz
    # group 2 transposon in Li, et al., Cell, 2009
    Trn_GROUP3=$COMMON_FOLDER/Zamore.transposon.group3.bed.gz
    # group 3 transposon in Li, et al., Cell, 2009, mainly somatic
    flyBase_Gene=$COMMON_FOLDER/UCSC.flyBase.Genes.bed12.gz
    # flyBase gene
    flyBase_Exon=$COMMON_FOLDER/UCSC.flyBase.Exons.bed.gz
    # flyBase exons
    flyBase_Intron=$COMMON_FOLDER/UCSC.flyBase.Introns.bed.gz
    # flyBase introns
    flyBase_Intron_xRM=$COMMON_FOLDER/UCSC.flyBase.Introns_xRM.bed.gz
    # flyBase introns that subtract repeatMasker
    flyBase_5UTR=$COMMON_FOLDER/UCSC.flyBase.5UTR.bed.gz
    # flyBase 5' UTR
    flyBase_CDS=$COMMON_FOLDER/UCSC.flyBase.CDS.bed.gz
    # flyBase CDS
    flyBase_3UTR=$COMMON_FOLDER/UCSC.flyBase.3UTR.bed.gz
    # flyBase 3' UTR
    cisNATs=$COMMON_FOLDER/cisNATs.bed.gz
    # cis-NATs
    structural_loci=$COMMON_FOLDER/structured_loci.bed.gz
    # structural loci
    lincRNA=$COMMON_FOLDER/lincRNA.Young.bed6.gz
    # linc RNA identified in 'Identification and properties of 1,119 candidate lincRNA loci in the
    # Drosophila melanogaster genome. Genome Biol Evol. 2012;4(4):427-42.'
    unannotated=$COMMON_FOLDER/unannotated_genome.bed.gz
    # unannoated region, basically all the genome segments between annotations defined above

# TARGETS is used in small RNA-seq and degradome-seq pipeline
    declare -a TARGETS=( \
    "piRNA_Cluster" \
    "piRNA_Cluster_42AB" \
    "piRNA_Cluster_20A" \
    "piRNA_Cluster_flam" \
    "repeatMasker" \
    "repeatMasker_IN_Cluster" \
    "repeatMasker_OUT_Cluster" \
    "Trn" \
    "Trn_IN_Cluster" \
```

```
    "Trn_OUT_Cluster" \
    "Trn_GROUP1" \
    "Trn_GROUP2" \
    "Trn_GROUP3" \
    "Trn_GROUP0" \
    "flyBase_Gene" \
    "flyBase_Exon" \
    "flyBase_Intron" \
    "flyBase_Intron_xRM" \
    "flyBase_5UTR" \
    "flyBase_CDS" \
    "flyBase_3UTR" \
    "cisNATs" \
    "structural_loci" \
    "lincRNA" \
    "unannotated" )

# TARGETS_SHORT is used for "cis-Ping-Pong" analysis between degradome/small RNA.
# Since this step uses multi-threading itself, we are not able to run each feature simultaneously
# thus a few less important ones have been removed
    declare -a TARGETS_SHORT=( \
    "piRNA_Cluster" \
    "piRNA_Cluster_42AB" \
    "piRNA_Cluster_20A" \
    "piRNA_Cluster_flam" \
    "repeatMasker" \
    "Trn" \
    "Trn_GROUP1" \
    "Trn_GROUP2" \
    "Trn_GROUP3" \
    "Trn_GROUP0" \
    "flyBase_Gene" \
    "flyBase_Exon" \
    "flyBase_Intron_xRM" \
    "flyBase_5UTR" \
    "flyBase_3UTR" \
    "lincRNA" )

# variables for small RNA direct mapping
    declare -a DIRECT_MAPPING=( "transposon" "repBase" "piRNAcluster" )

# gtf files for rnaseq/deg/cage htseq-count
    Genes_transposon_Cluster=$COMMON_FOLDER/dm3.genes+transposon+piRNACluster.gtf
    Genes_repBase_Cluster=$COMMON_FOLDER/dm3.genes+repBase+piRNACluster.gtf
    declare -a HTSEQ_TARGETS=( "Genes_transposon_Cluster" "Genes_repBase_Cluster" )
```

piPipes small RNA pipeline

This document explains how to run the **piPipes** small RNA pipeline and how to interpret the output.

This pipeline provides comprehensive analysis on piRNAs from a Fastq file generated by Next Generation Sequencing. It also provides very limited analysis on microRNAs (miRNAs).

This small RNA pipeline contains two modes: single-sample mode and dual-sample mode.

Single-sample mode provides analysis on single library and dual-sample mode offers side-by-side comparison between two libraries.

Example 1. Run single small RNA library

Install the fly *dm3* genome, if you haven't done so

```
# require internet access
# You will be prompted to define the length of siRNA and piRNA.
# For fly, our lab uses 20-22 nt for siRNAs and 23-29 nt for piRNAs.
piPipes install -g dm3

# if you would like to try the new BDGP6 assembly, then
piPipes install -g dm6
```

Download small RNA sample data from NCBI SRA and remove adaptors

In this example, we used `fastq-dump` to obtain the data from NCBI SRA and `cutadapt` to remove adaptors. The user does not need to use these two specific programs. As soon as the adaptors are removed, it should be fine. Compressing the Fastq file is also optional. **piPipes** generally can takes gzipped file as input.

```
# Use fastq-dump from SRATools (http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software)
# to download data and convert it to fastq; this step requires internet access.
# Use cutadapt (https://code.google.com/p/cutadapt/) to remove adaptor.
# those two programs do not come with piPipes so please install them if you don't have them
fastq-dump -F -Z SRR010951 | \
cutadapt -a TCGTATGCCG -O 6 -m 18 --discard-untrimmed - | \
gzip > Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz

# Note: We used the escape symbol \ throughout this document for clarity purpose (So that we can
# break the command into multiple lines. And also because Markdown language does not do text-wrap
# for codes)/ You can remove the \ symbol and put everything on one line, like
fastq-dump -F -Z SRR010951 | cutadapt -a TCGTATGCCG -O 6 -m 18 --discard-untrimmed - | gzip > ...
```

Check usage message

```
piPipes small -h
```

Using default parameters

```
# -i: input file, fastq file with barcodes/adaptors removed, can be compressed by gzip.
# -g: genome (dm3) to use. Need the genome dm3 to be installed first
# -o: output directory (optional) If not provided, will use current working directory
# 1>: write the STDOUT to this file (optional)
```

```
# 2>: write the STDERR to this file (optional)
piPipes small \
    -i Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz \
    -g dm3 \
    -o Zamore.SRA.ago3_het.ox.ovary.piPipes_out \
    1> Zamore.SRA.ago3_het.ox.ovary.piPipes.stdout \
    2> Zamore.SRA.ago3_het.ox.ovary.piPipes.stderr
```

Debug mode with more information printed to stderr

```
piPipes_debug small \
    -i Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz \
    -g dm3 \
    -o Zamore.SRA.ago3_het.ox.ovary.piPipes_out \
    1> Zamore.SRA.ago3_het.ox.ovary.piPipes.stdout \
    2> Zamore.SRA.ago3_het.ox.ovary.piPipes.stderr
```

Run the pipeline with optional parameter

```
# Additional parameter:
# -N: Reads used to normalize library;
#  for un-oxidized library, we recommend to use "miRNA";
#  for oxidized library, we recommend to use "uniqueXmiRNA" (unique genomic mappers
#  excluding miRNA and rRNA) or "siRNA" (non-transposon derived siRNA,
#  including cis-NATs and structural loci, this one is 'dm3' only)
#  Note that you can try multiple normalization methods for the same file using the same
#  output path, the outputs won't overwrite each other.
#  And the common steps will not ran. But don't run them simultaneously: run the second one
#  after the first one finish.
#  However, if you use different output directory, you can run them simultaneously.

# -c: number of CPU to use; use multiple CPUs will significantly improve the speed.
# -F: A list of Fasta files, delimited by comma, used to do filtering. Reads mappable to those
#  sequences are removed.
# -P: A list of Fasta files, delimited by comma, used to do pre-genomic mapping and analysis.
#  In this example: after removing reads mappable to rRNA and miRNA hairpin, reads are mapped to
#  miniWhite sequence first. Only the non-miniWhite-mappers are mapped to virus sequence.
#  And only the non-miniWhite, non-virus mappers will be used in the genome mapping
#  and further analysis. A few analysis will be done for miniWhite and virus mappers.

# -O: A list of Fasta files, delimited by comma, used to do post-genomic mapping and analysis.
#  For example here: after removing reads mappable rRNA, miRNA hairpin and genome, reads are mapped
#  to gfp sequence first. Only the non-genome non-gfp mappers are mapped to
#  luciferase sequence. A few analysis will be done for miniWhite and virus mappers.

# For -P and -O:
# (1) If more than one sequences are stored in each Fasta file, they will be treated equally.
# (2) Please only use letters and numbers as filename.
# (3) Use $HOME instead of ~ to indicate the home directory. Unless present at the
# beginning of a string, ~ is not properly expanded BEFORE piPipes can even see it
piPipes small \
    -i Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz \
    -g dm3 \
    -o Zamore.SRA.ago3_het.ox.ovary.piPipes_out \
```

```
    -N uniqueXmiRNA \
    -c 12 \
    -F $HOME/extra_fasta/primer_dimer \
    -P $HOME/extra_fasta/mini_white.fa,$HOME/extra_fasta/virus.fa \
    -O $HOME/extra_fasta/gfp.fa,$HOME/extra_fasta/luciferase.fa \
    1> Zamore.SRA.ago3_het.ox.ovary.piPipes.stdout \
    2> Zamore.SRA.ago3_het.ox.ovary.piPipes.stderr
```

Interpretation of the output files

The output folder should contain following folders

```
# output of Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz
input_read_files/
rRNA_mapping/
hairpins_mapping/
bigWig_normalized_by_unique/
genome_mapping/
intersect_genomic_features/
pdfs/
post_genome_mapping/
pre_genome_mapping/
summaries/
transposon_piRNAcluster_mapping_normalized_by_unique/
Zamore.SRA.ago3_het.ox.ovary.trimmed.basic_stats
```

`input_read_files/` contains input files for various mapping. All of them are in "insert format"

```
# insert format has two fields, sequence and number of time this sequence been read
$ head -3 Zamore.SRA.ago3_het.ox.ovary.trimmed.insert
CCTCCGACTTTTAGCGCTATC   1
TTTGATACAGTGAGGATAGAT   1
TAAGGTTCACTGTAGAGAACCAAGT   1
# insert file directly converted from the input Fastq
Zamore.SRA.ago3_het.ox.ovary.trimmed.insert
# insert file with rRNA mappable reads removed
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.insert
# insert file with reads mappable to miRNA hairpin
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.insert
# insert file with reads mappable to miRNA hairpin removed
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.insert
# insert file with reads mappable to genome; if -P is used, the filename will be different
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.al.insert
# insert file with reads non-mappable to genome; this file will be used for -O
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.insert
```

`rRNA_mapping/` contains **species** information on rRNA mapping. **piPipes** removes reads mappable to rRNA first since rRNA has been known to be the main source of contamination in small RNA, especially fly 2S that is 30 nt and easily gets cloned.

Note: it has been reported that small RNAs can be produced from tRNA and snoRNA. Thus tRNAs and snoRNAs are not included here. However, if the user would like to remove them (or whatever sequence), provide their sequences in a Fasta file and feed it to piPipes by `-P`.

```
# since we already compiled reads with the same sequence, shown here are species information
# the reads information can be found in Zamore.SRA.ago3_het.ox.ovary.trimmed.basic_stats
cat Zamore.SRA.ago3_het.ox.ovary.trimmed.rRNA.log
```

```
# reads processed: 755156
# reads with at least one reported alignment: 31350 (4.15%)
# reads that failed to align: 723806 (95.85%)
Reported 31350 alignments to 1 output stream(s)
# so 4.15% of the SPECIES are mappable to rRNA
```

`hairpins_mapping/` contains data on microRNA hairpin mapping.

```
# BED2 format is a derivative of BED format. Field 1,2,3,6 have same meaning as in BED.
#   It uses field 7 to record the original sequence, field 4 to record the number of reads
#   this sequence appears in the original Fastq, field 5 to record the number of times
#   this sequence can be assigned to the genome (or any index used)
# the filename v0m1 indicate the bowtie mapping parameter -v 0 -m 1, meaning
#   no mismatch allowed and only report unique mappers
# Note that if mismatch is allowed, sequence with sequencing error will be a different species
# and occupy a different line, even thought the first 3 field and field 6 (strand) can be the same
$ head -3 Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.v0m1.bed2
dme-mir-284 9   28  1   1   +   CCTGGAATTAAGTTGACTG
dme-mir-283 56  82  1   1   +   TATGAAACACTCGGAATTTCAGTTGG
dme-mir-989 140 160 1   1   +   TGATGTGACGTAGTGGAACA

# relative.bed2 modified the second and third fields of the BED2 format to report
#   the relative distance of the 5' and 3' ends to the miRBase annotated miRNA
# field 4 still records the number of reads this sequence appeared
# field 5 indicates whether this sequence is defined as
# 5' arm (1), loop (2) or 3' arm (3) or undefined (0).
#   If more than half of the read overlap with the annotated 5' arm miRNA, it's marked as 5' arm.
#   If more than half of the read overlap with the annotated 3' arm miRNA, it's marked as 3' arm.
$ head -5 Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.v0m1.bed2.relative
dme-mir-989 2   1   1   3   +   TGATGTGACGTAGTGGAACA
dme-mir-989 4   1   1   3   +   ATGTGACGTAGTGGAACA
dme-mir-989 0   -2  126 3   +   TGTGATGTGACGTAGTGGA
dme-mir-989 0   0   10  3   +   TGTGATGTGACGTAGTGGAAC
dme-mir-989 0   -1  10  3   +   TGTGATGTGACGTAGTGGAA

# the following file store the length distribution of reads that can uniquely be assigned to a miRNA.
#   Note that some miRNAs cannot be uniquely mapped to the hairpin index.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.v0m1.lendis

# sum file store the summary information:
# miRNA name, 5' arm miRNA number of reads, 5' arm miRNA 5' heterogeneity,
# 5' arm miRNA 3' heterogeneity, 3' arm miRNA number of reads, 3' arm miRNA 5' heterogeneity,
# 3' arm miRNA 3' heterogeneity.
# heterogeneity is calculated based on :
# The 3'-to-5' exoribonuclease Nibbler shapes the 3' ends of microRNAs bound to Drosophila Argonaute1.
#   Curr Biol. 2011 Nov 22;21(22):1878-87.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.v0m1.sum
```

`pre_genome_mapping/` contains output of pre-genome mapping. It is empty since we didn't use `-P` option.

`genome_mapping/` contains output of genomic mapping:

```
# although piPipes directly maps reads to miRNA hairpin before genomic mapping,
# it is still useful to know where miRNA come from in the genome.
# thus the pipeline also provides the coordinate of miRNA hairpin derived reads
# in BED2 format; note that dm3v0a means no mismatch but report all mappers.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.dm3v0a.log
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.dm3v0a.bed2
```

```
# all the multiple mappers can be retrieved using the fifth fields
$ awk '$5>1'  Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.hairpin.dm3v0a.bed2 | head -3
chr3LHet    1092010 1092029 1   3   -   TTAAATATCTGTGTGTGAA
chr2RHet    1322896 1322915 1   3   -   TTAAATATCTGTGTGTGAA
chr2RHet    549436  549455  1   3   -   TTAAATATCTGTGTGTGAA
# this sequence TTAAATATCTGTGTGTGAA was only read once but can be mapped to 3 loci

# the following 3 files store log and loci for genome mapping
# all: unique and multiple mappers; unique: unique mappers only
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.log
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.bed2
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.unique.bed2

# this file has added hairpin mapper information
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.unique.+hairpin.bed2

# the following two file separate siRNAs and piRNAs from the rest of the reads,
# based on the length defined by the user during the "install" step of this genome
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.siRNA.bed2
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.piRNA.bed2

# the following 3 files store the length distribution. The filenames are self-explanatory
# note that pdfs of length distribution has been generated and stored in the pdfs folder.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.unique.bed2.lendis
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.bed2.+hairpin.lendis
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.unique.bed2.+hairpin.lendis
```

`bigWig_normalized_by_unique/` store files useful for UCSC genome browser.

```
# bigWig format can be used by UCSC genome browser via URL (without uploading)
# Watson and Crick strands were separated.  The single has been normalized by the method
# the user chose. Note that the filename contains this information, so if the user decides to run
# a second time with different normalization method, they won't get overwritten.
# Also note that ONLY the 5' end of reads were used, since the SEED sequence was defined there and
# 3' end of small RNAs usually have heterogeneity

# the following 2 file contains the information on all mappers. The weight of each reads has been
# partitioned by the number of loci it can be mapped.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.sorted.Watson.bigWig
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.sorted.Crick.bigWig

# the following 2 file contains the information on unique mappers.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.sorted.uniq.Watson.bigWig
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.sorted.uniq.Crick.bigWig

# the following 4 files contain only the piRNAs signal. They were simply separated by the length.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.piRNA.sorted.Watson.bigWig
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.piRNA.sorted.Crick.bigWig
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.piRNA.sorted.uniq.Crick.bigWig
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.piRNA.sorted.uniq.Watson.bigWig

# Example of loading this to UCSC genome browser.
# 1. upload those bigWig files to a server
# 2. go to "Add Custom Tracks" in UCSC genome browser
# 3. pasting something like (expand the ellipsis)
track
name=Ago3het(+) maxHeightPixels=25 alwaysZero=on autoScale=on yLineMark=0 yLineOnOff=on type=bigWig
color=255,0,0 visibility=full
```

```
bigDataUrl=http://X/Zamore.SRA.ago3_het.ox...all.piRNA.sorted.uniq.Watson.bigWig

track
name=Ago3het(-) maxHeightPixels=25 alwaysZero=on autoScale=on yLineMark=0 yLineOnOff=on type=bigWig
color=0,0,255 visibility=full \
bigDataUrl=http://X/Zamore.SRA.ago3_het.ox...all.piRNA.sorted.uniq.Crick.bigWig
```

`intersect_genomic_features/` contains information on siRNAs/piRNAs that has been assigned to each genomic feature.

```
# the genomic feature is defined in /piPipes/common/dm3/genomic_features.
# please see the installation document for more details
# the following are all files for one genomic feature "pirna cluster"

# including 3 length distribution for uniquely mapped reads with all length, siRNA and piRNA
# multiple mappers can be assigned to more than one genomic features;
# and many of the genomic features are not exclusive to each other;
# so we decide to only look at unique mappers
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster

# the following 3 files are nucleotide composition 30 nt upstream and 30 nt downstream of the 5' end
# of the unique mappers that assigned to this genomic feature.
# To avoid few abundance reads dominates this analysis, we treat each sequence equally here
# as if they were all sequenced once
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster

# the following 3 files are cis-ping-pong values for unique mappers assigned to each feature
# ping-pong value of distance N = Sum (reads1 x reads2)
# read1 and read2 have their 5' end N nucleotide away from each other and they are on opposite strand.
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0.all.x_rpmk_MASK.bed2.intersect_with_piRNA_Cluster

# all the data for each genomic feature will be drawn as one page of pdf.
# Please see our manuscript or the Github wiki page for example.
```

`post_genome_mapping/` contains output of post-genome mapping. It is empty since we didn't use `-O` option.

`transposon_piRNAcluster_mapping_normalized_by_unique/` contains output that direct mapped reads to piRNA clusters and transposons instead of the genome.

```
# direct mapping to transposon, the annotation from flyBase
# the target used here for direct mapping can be configured in /piPipes/common/dm3/genomic_features
# summary file is used to generate pdf for reads signal across each sequence:

# ../pdfs/Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.transposon.pdf
transposon.log
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.transposon.a2.insert.bed2
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.transposon.a2.summary

# direct mapping to repBase annotation
# ../pdfs/Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.repBase.pdf
repBase.log
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.repBase.a2.insert.bed2
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.repBase.a2.summary
```

```
# direct mapping to piRNA cluster
# ../pdfs/Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.piRNAcluster.pdf
piRNAcluster.log
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.piRNAcluster.a2.insert.bed2
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.piRNAcluster.a2.summary
```

`Zamore.SRA.ago3_het.ox.ovary.trimmed.basic_stats` and has basic information on the library. `Zamore.SRA.ago3_het.ox.ovary` contains the reads number been partitioned to each genomic features. See the explanation for the pie chart below for more information.

```
$ cat Zamore.SRA.ago3_het.ox.ovary.trimmed.basic_stats
total reads as input of the pipeline    1884325
rRNA reads with 2 mismatches    311539
miRNA hairpin reads 125490
genome mapping reads (-rRNA; +miRNA_hairpin)    1246229
genome mapping reads (-rRNA; -miRNA_hairpin)    1120739
genome unique mapping reads (-rRNA; +miRNA_hairpin) 352468
genome unique mapping reads (-rRNA; -miRNA_hairpin) 226978
genome multiple mapping reads (-rRNA; -miRNA_hairpin)   893761
```

`pdfs/` contains all the pdf output:

```
# Summary for genomic mapping.
# this figure starts with a pie chart. Unique and multi-mappers are included here.
# reads are normalized by the number of times they map first
# then normalized by the number of genomic feature
# for example, if a sequence is read 300 times and can be mapped to 3 loci:
# 1 in piRNA cluster, which is also annotated as repeat by repeatMasker
# the other 2 outside piRNA cluster but still in repeats
# piRNA cluster will get 100/3/2 reads
# repeats get 100/3/2 + 100/3*2 reads

# Following the pie chart are length distribution for unique/multi mappers, +/- miRNA hairpin-mapper

# Followed by a bunch of figures with two figures representing information on a genomic feature
# different from the genomic feature in the pie chart, reads here are counted separately and some
# reads will get counted more than once. Although it might give false positive result but can
# guarantee that no information is lost.

# Two figures for each genomic feature include: unique_species and all_reads
# <unique_species>
## length distribution: unique mappers only, reads information is used
## nucleotide percentage: unique mappers in species, meaning that each sequence contribute equally
## ping-pong: unique mappers only, reads information is used

# <all_reads>
## length distribution: unique+multi mappers; multi-mappers normalized by number of loci
## nucleotide percentage: unique mappers in reads; could be overwhelmed by a few abundance sequences
## ping-pong: unique+multi mappers; multi-mappers normalized by number of loci
Zamore.SRA.ago3_het.ox.ovary.trimmed.piPipes.small_RNA_pipeline.1.0.0.pdf


# mapping signal across different sequences by direct mapping;
# unique+multi mappers with multi-mappers normalized by number of loci
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.transposon.pdf
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.repBase.pdf
Zamore.SRA.ago3_het.ox.ovary.trimmed.x_rRNA.x_hairpin.dm3v0a.un.piRNAcluster.pdf
```

Example 2. Run dual-sample mode to compare small RNA libraries from two samples

Run single-sample mode for each library first

```
fastq-dump -F -Z SRR010951 | \
    cutadapt -a TCGTATGCCG -O 6 -m 18 --discard-untrimmed - | \
    gzip > Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz
piPipes small \
    -i Zamore.SRA.ago3_het.ox.ovary.trimmed.fq.gz \
    -g dm3 \
    -o Zamore.SRA.ago3_het.ox.ovary.piPipes_out \
    1> Zamore.SRA.ago3_het.ox.ovary.piPipes.stdout \
    2> Zamore.SRA.ago3_het.ox.ovary.piPipes.stderr

fastq-dump -F -Z SRR010952 | \
    cutadapt -a TCGTATGCCG -O 6 -m 18 --discard-untrimmed - | \
    gzip > Zamore.SRA.ago3_mut.ox.ovary.trimmed.fq.gz
piPipes small \
    -i Zamore.SRA.ago3_mut.ox.ovary.trimmed.fq.gz \
    -g dm3 \
    -o Zamore.SRA.ago3_mut.ox.ovary.piPipes_out \
    1> Zamore.SRA.ago3_mut.ox.ovary.piPipes.stdout \
    2> Zamore.SRA.ago3_mut.ox.ovary.piPipes.stderr
```

Run dual-sample mode

```
#   -a: directory with output from single-sample mode
# -b: directory with output from single-sample mode
# -g: dm3 genome
# -o:   output directory
# -A:   Sample name for sample #1
# -B:   Sample name for sample #2
# -N:   Normalization method; We recommend miRNA for un-oxidized sample and siRNA for oxidized
piPipes small2 \
    -a Zamore.SRA.ago3_het.ox.ovary.piPipes_out \
    -b Zamore.SRA.ago3_mut.ox.ovary.piPipes_out \
    -g dm3 \
    -o Zamore.SRA.ago3_het_vs_ago3_mut.piPipes_out \
    -A ago3Het \
    -B ago3Mut \
    -N siRNA
```

Interpretation of the output files

The output folder should contain the following folders:

```
# microRNA comparison
hairpin_compare/
# transposon comparison
transposon_abundance/
# piRNA cluster comparison
piRNA_cluster_abundance/
# PDF output
pdfs/
```

`hairpin_compare/`

```
# abundance of miRNA sequences with different 5' and 3' end
ago3.miRNA.relative.abundance.normalized_by_sirna
# field 1:  miRNA name
# field 2:  relative distance to the 5' end of miRBase annotated miRNA
# field 3:  relative distance to the 3' end of miRBase annotated miRNA
# field 4:  normalized reads for 5' arm miRNA in sample A
# field 5:  normalized reads for 5' arm miRNA in sample B
# field 6:  normalized reads for 3' arm miRNA in sample A
# field 7:  normalized reads for 3' arm miRNA in sample B
# this file is used to generate "balloonplot" that can be found in pdfs folder
# Example and further explanation can be found in our manuscript and the Github Wiki
```

`transposon_abundance/` contains reads information assigned to each transposon. *piPipes* uses genomic coordinates in this function and reads are partitioned by the number of times they are mapped.

```
# ParaFly file, can be ignored
2933316381.para
2933316381.para.completed

# Normalized reads for each transposon
ago3Het.transposon.abundance.normalized_by_sirna
ago3Mut.transposon.abundance.normalized_by_sirna

# mean length for each transposon
ago3Het.transposon.mean_len.normalized_by_sirna
ago3Mut.transposon.mean_len.normalized_by_sirna

# mean length for each transposon, with 0 removed
ago3Het.transposon.mean_len.normalized_by_sirna.no_zero
ago3Mut.transposon.mean_len.normalized_by_sirna.no_zero

$ head -3 ago3Het.transposon.abundance.normalized_by_sirna
FBgn0003122_pogo    0   60418.58    121559.96
FBgn0043969_diver   1   438188.83   1218590.01
suffix  0   19172.24    594398.33
# field 1:  name of the transposon
# field 2:  transposon family defined in Li, et al., Cell, 2009
# field 3:  Normalized piRNA reads assigned to this transposon in the SENSE direction
# field 4:  Normalized piRNA reads assigned to this transposon in the ANTISENSE direction

$ head -3 ago3Mut.transposon.mean_len.normalized_by_sirna
FBgn0003122_pogo    0   25.80   26.50
FBgn0043969_diver   1   25.36   25.21
suffix  0   26.05   25.04
# field 1:  name of the transposon
# field 2:  transposon family defined in Li, et al., Cell, 2009
# field 3:  Mean length of piRNA reads assigned to this transposon in the SENSE direction
# field 4:  Mean length of piRNA reads assigned to this transposon in the ANTISENSE direction

# those files are used to draw abundance and mean length scatter plots.
```

`piRNA_cluster_abundance/` contains reads information assigned to each piRNA cluster. It is very similar to transposon.

`pdfs/` contains figures generated

```
# microRNA balloon plot
ago3.miRNAballoon.normalized_by_sirna.pdf

# transposon abundance
ago3Het_vs_ago3Mut.transposon.abundance.normalized_by_sirna.csv
ago3Het_vs_ago3Mut.transposon.abundance.normalized_by_sirna.pdf
# transposon mean length
ago3Het_vs_ago3Mut.transposon.mean_len.normalized_by_sirna.csv
ago3Het_vs_ago3Mut.transposon.mean_len.normalized_by_sirna.pdf
# piRNA cluster abundance
ago3Het_vs_ago3Mut.piRNAcluster.abundance.normalized_by_sirna.csv
ago3Het_vs_ago3Mut.piRNAcluster.abundance.normalized_by_sirna.pdf
```

Please see example figures from the Github Wiki page or our manuscript.

Flowchart and example figures from our manuscript

piPipes RNA-seq pipeline

This document explains how to run the **piPipes** RNA-seq pipeline and how to interpret the output.

This pipeline provides analysis on genes and transposons abundance at the transcriptome level using paired-end RNA-seq reads generated by Next Generation Sequencing.

The RNA-seq pipeline contains two modes: single-sample mode and dual-sample mode.

Note: this pipeline can also be utilized for general gene quantification.

Example 1. Run single RNA-seq library

Install the mouse mm9 genome, if you haven't done so

```
# require internet access
piPipes install -g mm9
```

Download RNA-seq sample data from NCBI SRA

```
# use fastq-dump from SRATools (http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software)
# to download data and convert to fastq; require internet access
# --split-3 split the left and right read from paired-end RNA-seq data
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_het.testis.14dpp.rep1 SRR765641
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_het.testis.14dpp.rep2 SRR765642
# the two libraries are two replicates from the same genotype
```

Check usage message

```
piPipes rna
# or
piPipes rna -h
```

Using default parameters

```
# -l: left read of  RNA-seq
# -r: right read of RNA-seq
# -g: genome to use
# -o: output directory
piPipes rna \
  -l Zamore.RSQ.amyb_het.testis.14dpp.rep1_1.fastq.gz \
  -r Zamore.RSQ.amyb_het.testis.14dpp.rep1_2.fastq.gz \
  -g mm9 \
  -o Zamore.RSQ.amyb_het.testis.14dpp.rep1.piPipes_out \
  1> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stdout \
  2> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stderr
```

Debug mode with more information printed to stderr

```
piPipes_debug rna \
  -l Zamore.RSQ.amyb_het.testis.14dpp.rep1_1.fastq.gz \
  -r Zamore.RSQ.amyb_het.testis.14dpp.rep1_2.fastq.gz \
  -g mm9 \
  -o Zamore.RSQ.amyb_het.testis.14dpp.rep1.piPipes_out \
  1> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stdout \
  2> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stderr
```

Run the pipeline with optional parameters

```
# -l: left read of  RNA-seq
# -r: right read of RNA-seq
# -g: genome to use
# -o: output directory
# -L: ligation based RNA-seq construction method.
#     \1 has the same direction as the original RNA.
#     this option is by default off, meaning that dUTP based method is by default.
# -c: number of CPUs to use; using more CPU will significantly increase the speed
# -B: rounds of batch algorithm to use in eXpress. Use default unless it takes too long
# Read more in:
# Roberts A and Pachter L (2012). Streaming fragment assignment for real-time
# analysis of sequencing experiments. Nature Methods.
piPipes_debug rna \
  -l Zamore.RSQ.amyb_het.testis.14dpp.rep1_1.fastq.gz \
  -r Zamore.RSQ.amyb_het.testis.14dpp.rep1_2.fastq.gz \
  -g mm9 \
  -c 24 \
  -B 15 \
  -o Zamore.RSQ.amyb_het.testis.14dpp.rep1.piPipes_out \
  1> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stdout \
  2> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stderr
```

Interpretation of the output files

The output directory should contain the following directories and files:

```
rRNA_mapping/
input_read_files/
cufflinks_output/
bigWig/
htseq_count/
genome_mapping/
Zamore.RSQ.amyb_het.testis.14dpp.basic_stats
pdfs/
gene_transposon_cluster_direct_mapping/
```

`Zamore.RSQ.amyb_het.testis.14dpp.basic_stats` contains the basic statistical information on the library

```
$ cat Zamore.RSQ.amyb_het.testis.14dpp.basic_stats
total_input_reads:  32594566
rRNA_reads: 722247
genomie_mapper_reads:   26732381
genomie_unique_mapper_reads:    23749430
genomie_multiple_mapper_reads:  2982951
genomie_unmappable_reads:   5139938
```

`rRNA_mapping/` contains log file of rRNA mapping (Bowtie2)

```
$ cat Zamore.RSQ.amyb_het.testis.14dpp.rRNA.log
32594566 reads; of these:
  32594566 (100.00%) were paired; of these:
    31872319 (97.78%) aligned concordantly 0 times
    722247 (2.22%) aligned concordantly exactly 1 time
    0 (0.00%) aligned concordantly >1 times
2.22% overall alignment rate
```

`input_read_files/` contains input Fastq file for genome mapping by STAR

```
# x_rRNA means rRNA mappable sequences have been removed
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.1.fq
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.2.fq
```

`genome_mapping/` contains data for genome mapping

```
# outputs of STAR. Please see its document for detailed explanation
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.SJ.out.tab
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Log.progress.out
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Log.out
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Log.final.out

# unmappable sequence in Fastq format, might be used to map sequences that
# are not in the assembled genome, like transgene
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Unmapped.out.mate1
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Unmapped.out.mate2
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.STAR.log

# genome alignment file, sorted by COORDINATES
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.bam
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.bam.bai

# genome aligment file in normalized.bedpe format and bed12 format
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.all.normalized.bedpe
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.unique.normalized.bedpe
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.bed12

# normalized bedpe format is a modified BEDPE format (please see BEDTools document for BEDPE)
# briefly, each line represents an alignment for a pair.
# each field, from left to right, represents:
#   chromosome of mate1
#   start of mate1 (0-based, included)
#   end of mate1 (0-based, not included)
#   chromosome of mate2
#   start of mate2 (0-based, included)
#   end of mate2 (0-based, not included)
#   name of this pair, different from small RNA pipeline, reads in RNA-seq pipeline are not compiled
#   normalized read. If this pair can be mapped to N loci in the genome, this number if 1/N
# Note that we called STAR using --outFilterMultimapScoreRange 1, thus only the best mappers
# are reported
# strand of mate1
# strand of mate2
$ head Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.all.normalized.bedpe
chr15   89135450    89135627    chr15   89135620    89135802    FCC1GJJACXX:6:1101:15553:94247  1   +   -
chr5    64702364    64702464    chr5    64702308    64702408    FCC1GJJACXX:6:1101:16706:94222  1   -   +
chr5    54045627    54045727    chr5    54045421    54045521    FCC1GJJACXX:6:1101:15621:94056  1   -   +
```

```
chr9     21440366    21440708    chr9    21440233    21440333    FCC1GJJACXX:6:1101:17366:94052  1   -   +
chr3     97943583    97945464    chr3    97943552    97945433    FCC1GJJACXX:6:1101:16755:94111  1   -   +
chrM     5706    5806    chrM    5596    5696    FCC1GJJACXX:6:1101:17540:94078  1   -   +
chr15    12195313    12195413    chr15   12193559    12195305    FCC1GJJACXX:6:1101:16406:94075  1   -   +
chrM     1230    1330    chrM    1176    1276    FCC1GJJACXX:6:1101:16781:94117  1   -   +
chr12    70260428    70260528    chr12   70260297    70260397    FCC1GJJACXX:6:1101:17995:94098  0.5 -   +
chr12    70462221    70462321    chr12   70462352    70462452    FCC1GJJACXX:6:1101:17995:94098  0.5 +   -

# normalized bedpe file with only unique mappers
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.unique.normalized.bedpe

# unique mappers in bed12 format. \1 and \2 occupy two lines
# the strand has been reset to follow the original RNA:
# for dUTP method, \1 has been reversed; for ligation method, \2 has been reversed
$ head -2 Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.bed12
chr10    3000542 3000642 FCC1GJJACXX:6:1202:3559:62122/2 1    +    3000542 3000642 255,0,0 1    100 0
chr10    3000597 3000697 FCC1GJJACXX:6:1202:3559:62122/1 1    +    3000597 3000697 255,0,0 1    100 0
```

`cufflinks_output/` contains the output from Cufflinks

No transposon sequence is included in this GTF file, so the output can be used for general purpose

```
# piPipes ran cufflinks using gene only GTF without transposon annotation
# as well as --compatible-hits-norm option.
# the depth calculated by cufflinks is used in the entire pipeline to represent depth
# in most of piRNA mutants, transposon reads have huge amount of increase
# using gene compatible reads is less biased than using total reads

# see cufflinks document for more information
skipped.gtf
transcripts.gtf
isoforms.fpkm_tracking
# this document contains FPKM values for annotated gene
genes.fpkm_tracking
Zamore.RSQ.amyb_het.testis.14dpp.cufflinks.log
```

`bigWig/` contains bigWig files for UCSC genome browser. The signal contains unique mappers only and has been normalized to the library depth, which is calculated by cufflinks using annotation compatible reads.

```
# Watson and Crick strands are separated
# bedGraph files are intermediates file. they might be removed in the future version

Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.Crick.bedGraph
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.Watson.bedGraph
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.Crick.bigWig
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.sorted.unique.Watson.bigWig
```

`htseq_count/` contains outputs from HTSeq

```
# HTSeq is an alternative abundance calculation tool of Cufflinks and eXpress
# Different from Cufflinks and eXpress, which use EM-algorithm to assign multi-mappers to different
# transcript isoforms, HTSeq only count unique mappers.

# We uses a annotation comprised of Genes, piRNA cluster and repBase (repeatMasker)
# strict and union are two ways HTSeq supports, please see HTSeq document for more information

# S and AS represent sense and antisense;
# We realized that in dUTP method, ~10% of reads lost their strand information
```

```
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Genes_repBase_Cluster.htseqcount.strict.S.out
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Genes_repBase_Cluster.htseqcount.strict.AS.out
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Genes_repBase_Cluster.htseqcount.union.S.out
Zamore.RSQ.amyb_het.testis.14dpp.x_rRNA.mm9.Genes_repBase_Cluster.htseqcount.union.AS.out
```

`gene_transposon_cluster_direct_mapping/` contains output from eXpress Different from the "genome mapping + Cufflinks/HTSeq" strategy, this method directly aligns input reads to transcriptome and estimates transcript abundance from there. Same as Cufflinks, eXpress also uses "EM algorithm" for multiple-mappers assignment.

```
# direct alignment by Bowtie2
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.log
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.bam
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.bed

# transposon sizes file, required to make bigWig
transposon.sizes

# bigWig file for each transcripts (gene, piRNA cluster and repBase transposon)
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.plus.bedGraph
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.minus.bedGraph
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.plus.bigWig
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.minus.bigWig

# ParaFly file, ignore
bigWigSummary.para
bigWigSummary.para.completed

# summary file used to draw plots representing signal across different
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.sorted.unique.bigWig.summary

# output of eXpress calculation. No normalization is used in results.xprs.
# Please see its document for detailed explanation.
# Note that the data will be organized and used in the dual-sample mode
# we suggest to look at data there
results.xprs
params.xprs
Zamore.RSQ.amyb_het.testis.14dpp.gene+cluster+repBase.eXpress.log

# normalized results, using the gene compatible reads calculated by Cufflinks
results.xprs.normalized
```

Example 2. Run dual-sample mode to compare RNA-seq libraries from two samples, each with two biological replicates

Run single-sample mode for each library

```
# download all the data: two replicates for each samples mybl1+/- and mybl1-/-
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_het.testis.14dpp.rep1 SRR765641
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_het.testis.14dpp.rep2 SRR765642
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_mut.testis.14dpp.rep1 SRR765647
fastq-dump -F --split-3 --gzip -A Zamore.RSQ.amyb_mut.testis.14dpp.rep2 SRR765648

# run the four libraries separately
piPipes rna \
  -l Zamore.RSQ.amyb_het.testis.14dpp.rep1_1.fastq.gz \
  -r Zamore.RSQ.amyb_het.testis.14dpp.rep1_2.fastq.gz \
```

```
  -g mm9 \
  -o Zamore.RSQ.amyb_het.testis.14dpp.rep1.piPipes_out \
  1> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stdout \
  2> Zamore.RSQ.amyb_het.testis.14dpp.rep1.stderr
piPipes rna \
  -l Zamore.RSQ.amyb_het.testis.14dpp.rep2_1.fastq.gz \
  -r Zamore.RSQ.amyb_het.testis.14dpp.rep2_2.fastq.gz \
  -g mm9 \
  -o Zamore.RSQ.amyb_het.testis.14dpp.rep2.piPipes_out \
  1> Zamore.RSQ.amyb_het.testis.14dpp.rep2.stdout \
  2> Zamore.RSQ.amyb_het.testis.14dpp.rep2.stderr
piPipes rna \
  -l Zamore.RSQ.amyb_mut.testis.14dpp.rep1_1.fastq.gz \
  -r Zamore.RSQ.amyb_mut.testis.14dpp.rep1_2.fastq.gz \
  -g mm9 \
  -o Zamore.RSQ.amyb_mut.testis.14dpp.rep1.piPipes_out \
  1> Zamore.RSQ.amyb_mut.testis.14dpp.rep1.stdout \
  2> Zamore.RSQ.amyb_mut.testis.14dpp.rep1.stderr
piPipes rna \
  -l Zamore.RSQ.amyb_mut.testis.14dpp.rep2_1.fastq.gz \
  -r Zamore.RSQ.amyb_mut.testis.14dpp.rep2_2.fastq.gz \
  -g mm9 \
  -o Zamore.RSQ.amyb_mut.testis.14dpp.rep2.piPipes_out \
  1> Zamore.RSQ.amyb_mut.testis.14dpp.rep2.stdout \
  2> Zamore.RSQ.amyb_mut.testis.14dpp.rep2.stderr
```

Run dual-sample mode

```
# -a: comma delimited directories with RNA-seq single library mode, for sample A
# -b: comma delimited directories with RNA-seq single library mode, for sample B
# -g: mouse mm9 genome
# -o: output directory
# -A: name for sample A, used in output
# -B: name for sample B, used in output
piPipes rna2 \
  -a Zamore.RSQ.amyb_het.testis.14dpp.rep1.piPipes_out,Zamore.RSQ.amyb_het.testis.14dpp.rep2.piPipes_out \
  -b Zamore.RSQ.amyb_mut.testis.14dpp.rep1.piPipes_out,Zamore.RSQ.amyb_mut.testis.14dpp.rep2.piPipes_out \
  -g mm9 \
  -o Zamore.RSQ.amyb_het.vs.amyb_mut.14dpp \
  -A amybHet14 \
  -B amybMut14 \
  1> Zamore.RSQ.amyb_het.vs.amyb_mut.14dpp.piPipes.stdout \
  2> Zamore.RSQ.amyb_het.vs.amyb_mut.14dpp.piPipes.stderr
```

Interpretation of the output files

The dual-sample mode is very simple:

For genes quantification, it uses `Cuffdiff` to call differentially expressed genes/transcripts from genome-mapping approach and `cummeRbund` to generate figures.

For transposon quantification, it generates scatter-plot from `eXpress` output.

```
# output
cuffdiff_output/
pdfs/
```

```
amybHet14.results.xprs
amybMut14.results.xprs
```

`cuffdiff_output/` contains output of `Cuffdiff`.

```
# Cuffdiff and cummeRbund output
var_model.info
amyb.cuffdiff.log
isoform_exp.diff
tss_group_exp.diff
gene_exp.diff
cds_exp.diff
splicing.diff
promoters.diff
cds.diff
isoforms.fpkm_tracking
tss_groups.fpkm_tracking
cds.fpkm_tracking
# fpkm values for genes
genes.fpkm_tracking
# fpkm values for isoforms
isoforms.count_tracking
tss_groups.count_tracking
cds.count_tracking
genes.count_tracking
isoforms.read_group_tracking
tss_groups.read_group_tracking
cds.read_group_tracking
genes.read_group_tracking
read_groups.info
run.info
bias_params.info
cuffData.db
```

`pdfs/` contains pdfs

```
# the following files are generated by Cuffdiff + cummeRbund, for genes
# they should be self-explanatory
# please see cummeRbund document for detailed information
amyb.genes.csDensity.pdf
amyb.genes.csBoxplot.pdf
amyb_amybHet14_vs_amybMut14.genes.csScatter.pdf
amyb_amybHet14_vs_amybMut14.genes.csVolcano.pdf
amyb.genes.csHeatMap_top50.pdf
amyb.genes.csExpressionBarplot_top50.pdf
amyb.isoforms.csDensity.pdf
amyb.cummeRbund.log
amyb.isoforms.csBoxplot.pdf

# the following files are generated from eXpress output; for gene + piRNA cluster + transposon
# summary file
amybHet14_vs_amybMut14.gene_transposon_cluster.abundance.csv
# this scatter-plot has each dot representing a gene isoforms, a non-coding RNA, piRNA cluster and
# transposon usually we notice that gene and non-coding RNA align pretty well on the diagonal
# but transposon reads shifted pretty badly to one side
amybHet14_vs_amybMut14.gene_transposon_cluster.abundance1.pdf
# this is a simplified version using contour lines to represent gene and non-coding RNA transcripts
# because we found that there are too many dots in the previous pdf and very hard to manipulate/view
amybHet14_vs_amybMut14.gene_transposon_cluster.abundance2.pdf
```

Please see example figures from the Github Wiki page or our manuscript.

Flowchart and example figures from our manuscript

piPipes Degradome/RACE/CAGE-seq pipeline

This document explains how to run **piPipes** Degradome/CAGE/RACE-seq pipeline and how to interpret the output.

This pipeline provides analysis on (1) abundance (2) 5' feature and (3) relative distance to piRNAs for genes and transposons derived, 5' mono-phosphated transcripts using single-end or paired-end Degradome-seq reads generated by Next Generation Sequencing.

Example 1. Run single Degradome-seq library

Install the mouse mm9 genome, if you haven't done so

```
# require internet access
piPipes install -g mm9
```

Download Degradome-seq and small RNA-seq (optional) sample data from NCBI SRA

```
# use fastq-dump from SRATools (http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software)
# to download data and convert to fastq; require internet access
fastq-dump -F --gzip -A Pillai.CAGE.miwi_het.testis.adult SRR363963

# to analyze the cleavage signature between small RNA and degradome/RACE, we also need
# download the small RNA-seq data for the same/similar sample
fastq-dump -F -Z SRR363958 | \
  cutadapt -a TCGTATGCCG -O 6 -m 18 --discard-untrimmed - | \
  gzip > Pillai.SRA.wild_type.testis.adult.trimmed.fq.gz
```

Check usage message

```
piPipes deg
# or
piPipes deg -h
```

Using default parameters, without small RNA related analysis

```
# -i: input fastq or gzipped fastq for the degradome-seq (single-end)
# -g: use mouse genome mm9
# -c: number of CPUs to use
# -o: output directory
piPipes deg \
  -i Pillai.CAGE.miwi_het.testis.adult.fastq.gz \
  -g mm9 \
  -c 8 \
  -o Pillai.CAGE.miwi_het.testis.adult.piPipes_out \
  1> Pillai.CAGE.miwi_het.testis.adult.piPipes.stdout \
  2> Pillai.CAGE.miwi_het.testis.adult.piPipes.stderr
```

Run the pipeline with small RNA analysis (recommended)

The purpose of degradome cloning in piRNA field is to capture the cleavage product of PIWI proteins. However, due to the ubiquitousness of 5' monophosphate and the instability of cleavage product, only a small portion of the reads correspond to the cleavage product. Then it is necessary to analyze the relative 5' to 5' distance between small RNA and degradome reads.

```
# Run small RNA pipeline first
# -i: input small RNA sequencing data in Fastq or gzipped Fastq format
# -g: use mouse genome mm9
# -c: number of CPUs to use
# -o: output directory
piPipes small \
  -i Pillai.SRA.wild_type.testis.adult.trimmed.fq.gz \
  -g mm9 \
  -o Pillai.SRA.wild_type.testis.adult.piPipes_output \
  -c 8 \
  1> Pillai.SRA.wild_type.testis.adult.piPipes.stdout \
  2> Pillai.SRA.wild_type.testis.adult.piPipes.stderr

# Run degradome pipeline with small RNA data
# -i: input Fastq or gzipped Fastq for the degradome-seq (single-end)
# -g: use mouse genome mm9
# -c: number of CPUs to use
# -o: output directory
# -s: directory with small RNA pipeline output
piPipes deg \
  -i Pillai.CAGE.miwi_het.testis.adult.fastq.gz \
  -g mm9 \
  -s Pillai.SRA.wild_type.testis.adult.piPipes_output \
  -c 8 \
  -o Pillai.CAGE.miwi_het.testis.adult.piPipes_out \
  1> Pillai.CAGE.miwi_het.testis.adult.piPipes.stdout \
  2> Pillai.CAGE.miwi_het.testis.adult.piPipes.stderr
```

Interpretation of the output files

The output directory should contain the following folders and files:

```
cufflinks_output/
htseq_count/
rRNA_mapping/
input_read_files/
genome_mapping/
bigWig/
summaries/
pdfs/
bedtools_count/
gene_transposon_cluster_direct_mapping/
map_small_RNA/
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.basic_stats
bowtie_index/
```

`rRNA_mapping/` contains log files of rRNA mapping using `Bowtie2`

Similar to RNA-seq, degradome pipeline also removes rRNA mappable reads first.

`genome_mapping/` contains output for genome mapping

```
# STAR output
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.SJ.out.tab
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.Log.progress.out
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.Log.out
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.Log.final.out
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.Unmapped.out.mate1
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.STAR.log

# genome alignment file in bam format, sorted by coordinates
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.bam
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.bam.bai

# genome alignment file in bed12 format
# since degradome/CAGE requires the 5' end of the reads being precisely mapped
# alignments with soft-clipping on the 5' ends are removed
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.unique.bed12
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.bed12

# "unique bead12" file contains only unique mappers
# each line is an alignment of a sequence
$ head Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.unique.bed12
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:32:14333:13809 1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:8:5133:15456   1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:15:18950:16133 1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:33:3872:8475   1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:11:2584:14460  1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:15:9578:3676   1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:28:10944:3671  1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:6:13520:3867   1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:18:14786:3150  1   +   3218970 3219075 255,0,0 1   105 0
chr10   3218970 3219075 HWUSI-EAS702:65:FC:1:29:18350:4783  1   +   3218970 3219075 255,0,0 1   105 0

# "all bed12" file contains signals from all mappers
# each line is no longer an alignment but accumulated signal from a species alignment
# field 4:  accumulated signal from multi count. for example, 26 means that there are
# 26 reads from the library can be mapped here; but they could be mapped to elsewhere as well
# field 5:  accumulated signal from reads that have been apportioned to the number
# of loci they can be mapped to. For example, if a read can be mapped to 2 loci, its contribution
# is 0.5. And from all the 26 reads that can be mapped here, their contribution is 0.999...
$ head Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.bed12
chr1    100010440   100010544   26  0.999999000000000415334 -   100010440   100010544   1   1   104 0
chr1    100011803   100011908   1   0.0384615000000000026303    +   100011803   100011908   2   1   105 0
chr1    100089433   100089521   3   3   -   100089433   100089521   3   1   88  0
chr1    100101123   100101228   1   0.142857000000000011752 -   100101123   100101228   4   1   105 0
chr1    100101492   100101597   1   1   -   100101492   100101597   5   1   105 0
chr1    100101531   100101636   1   0.0500000000000000027756    -   100101531   100101636   6   1   105 0
chr1    10015089    10017223    58  58  -   10015089    10017223    7   2   55,40   0,2094
chr1    100181199   100181293   1   0.333332999999999990415 +   100181199   100181293   8   1   94  0
chr1    100184297   100184402   28  1.55263279999999959102  +   100184297   100184402   9   1   105 0
chr1    100187852   100187957   1   0.0151515000000000000374    +   100187852   100187957   10  1   105 0
```

`bedtools_count/` contains nucleotide percentage surrounding different genomic features. If small RNA data is provided, the pipeline also calculates the "cis Ping-Pong" signature between small RNA and degradome. See the `pdfs/` folder for the figure output.

```
# the data are used to draw percentage plots in pdfs folder
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.x_rpmk_MASK.bed12
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.x_rpmk_MASK.bed12.intersect_with_hybrid_
```

```
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.x_rpmk_MASK.bed12.intersect_with_prepach
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.x_rRNA.mm9.sorted.noS.all.x_rpmk_MASK.bed12.intersect_with_Zamore_
```

`bigWig/` contains bigWig files that can be used in UCSC genome browser.

`summaries/` contains summary files with statistics on different genomic feature. Note that tRNA, snoRNA et al. has been removed before doing the interacting analysis. Thus the counts in the summary table are smaller than the one in the basic.stats file.

```
feature total_lib_all_mapper_reads  total_feature_all_mapper_reads  feature_all_mapper_percentage   \
feature_sense_all_mapper_reads  feature_antisense_all_mapper_reads  feature_all_mapper_sense_fraction   \
total_lib_unique_mapper_reads   total_feature_unique_mapper_reads   feature_unique_mapper_percentage   \
feature_sense_unique_mapper_reads   feature_antisense_unique_mapper_reads   \
feature_unique_mapper_sense_fraction    total_lib_unique_mapper_species \
total_feature_unique_mapper_species feature_unique_mapper_percentage    \
feature_sense_unique_mapper_species feature_antisense_unique_mapper_species \
feature_unique_mapper_sense_fraction
piRNA_Cluster_EXON  3643781 86474   0.024   79067   7407    0.914   49220   84057   1.708   76909   7148    0
prepachytene_piRNA_Cluster_EXON 3643781 4931    0.001   4785    146 0.970   49220   4726    0.096   4590    1
hybrid_piRNA_Cluster_EXON   3643781 4720    0.001   4410    310 0.934   49220   4694    0.095   4385    309 0
pachytene_piRNA_Cluster_EXON    3643781 76824   0.021   69872   6952    0.910   49220   74637   1.516   67934
```

`gene_transposon_cluster_direct_mapping/` contains direct mapping output to transcriptome (gene + piRNA cluster + transposon concensus sequence) as well as abundance evaluation by eXpress

```
# mapping output
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.gene+cluster+repBase.log
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.gene+cluster+repBase.bam

# abundance estimation by eXpress
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.gene+cluster+repBase.eXpress.log
results.xprs
params.xprs
```

`bowtie_index/` contains `bowtie` index build from degradome reads themselves or the sequence retrieved from the genome surrounding the 5' end of degradome reads. They are used as an index for small RNA mapping. The coordinates tell the relative distance between small RNA and degradome.

```
# index was build from the degradome directly; if paired-end is used, only the \1 read is used.
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.4.ebwt
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.3.ebwt
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.1.ebwt
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.2.ebwt
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.rev.1.ebwt
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.rev.2.ebwt

# 200 nucleotide surrounding the 5' end of the degradome reads were taken out, reverse-
# complemented and used to build an index for small RNA to map
# the index was build from degradome reads assigned to different genomie features
# Note that one coordinate is used once, even more than one read can be mapped to
# this coordinate.
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.piRNA_Cluster_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.4.bt2
```

```
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.prepachytene_piRNA_Cluster_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.hybrid_piRNA_Cluster_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.pachytene_piRNA_Cluster_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NM_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.Zamore_NR_EXON.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.refSeq_GENE.RC.ext200.unique.rev.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.4.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.3.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.2.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.rev.1.bt2
Pillai.CAGE.miwi_het.testis.adult.fastq.gz.repeatMasker.RC.ext200.unique.rev.2.bt2
```

`map_small_RNA/` contains the mapping result of small RNA to (1) degradome index; (2) 401 nt (200 x 2 + 1) index

```
# mapping of small RNA directly to the degradome index
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.map_to.Pillai.CAGE.miwi
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.map_to.Pillai.CAGE.miwi
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.map_to.Pillai.CAGE.miwi
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.map_to.Pillai.CAGE.miwi
# mapping of small RNA to the 401 nt index
# from left to right: + mapping 5' end, + mapping 3' end, - mapping 3' end, - mapping 5' end
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
```

```
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
Pillai.SRA.wild_type.testis.adult.trimmed.x_rRNA.x_hairpin.mm9v1.all.x_rpmk_MASK.bed2.piRNA_map_to.Pillai.CAG
```

`pdfs/` contain the pdf outputs

Please see example figures from the Github Wiki page or our manuscript.

Flowchart and example figures from our manuscript

piPipes ChIP-seq pipeline

This document explains how to run **piPipes** ChIP-seq pipeline and how to interpret the output.

This pipeline provides analysis on the abundance of a specific chromatin signal on genes and transposons using single-end or paired-end ChIP-seq reads generated by Next Generation Sequencing.

The ChIP-seq pipeline contains two modes: single-sample mode and dual-sample mode. Please see the examples below.

Example 1. Run single sample ChIP-seq library

Install the fly dm3 genome, if you haven't done so

```
# require internet access
piPipes install -g dm3
```

Download ChIP-seq sample data from NCBI SRA

ChIP-seq usually contains two libraries for each experiment: input control and immunoprecipitates.

```
# use fastq-dump from SRATools (http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software)
# to download data and convert to fastq; require internet access
# --split-3 split the left and right read from paired-end ChIP-seq data
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.input.nos_piwi   SRR646594
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.H3K9me3.nos_piwi SRR646595
```

Check usage message

```
piPipes chip
# or
piPipes chip -h
```

Using default parameters

```
# -l: left read of  ChIP-seq IP
# -r: right read of ChIP-seq IP
# -L: left read of  ChIP-seq input
# -R: right read of ChIP-seq input
# -g: genome to use
# -o: output directory
piPipes chip \
  -l Hannon.ChIP.H3K9me3.nos_piwi_1.fastq.gz \
  -r Hannon.ChIP.H3K9me3.nos_piwi_2.fastq.gz \
  -L Hannon.ChIP.input.nos_piwi_1.fastq.gz \
  -R Hannon.ChIP.input.nos_piwi_2.fastq.gz \
  -g dm3 \
  -B \
  -o Hannon.ChIP.H3k9me3.nos_piwi.piPipes_out \
  1> Hannon.ChIP.H3k9me3.nos_piwi.piPipes.stdout \
  2> Hannon.ChIP.H3k9me3.nos_piwi.piPipes.stderr
```

Run the pipeline with optional parameter

```
# -B: broad peak; used for H3K9me3; for transcriptional factor, don't use this option
# -x: extend this many nucleotide on both sides of genomic features for TSS/TES analysis
#     Note that for meta-analysis of gene body, the extended length is proportional to the size
#     of the gene body (1:1:1) so this parameter does not influence meta-gene
# -M: Path to BED files for meta-plot analysis on user-defined regions.
#     The files should be delimited by comma.

# the following three options toggle the usage of multi-mappers (unambiguous mappers)
# -u: Only use unique mappers. default: on
# -m: Use both unique and multi-mappers. For multi-mappers, let Bowtie2 randomly report one locus
#     from the best alignment pool. default: off
# -e: Use both unique and multi-mappers. For multi-mappers, use Expectation-Maximization algorithm
#     implemented by CSEM to allocate them. Only alignments passing 0.5 CSEM posterior are kept.

piPipes chip \
  -l Hannon.ChIP.H3K9me3.nos_piwi_1.fastq.gz \
  -r Hannon.ChIP.H3K9me3.nos_piwi_2.fastq.gz \
  -L Hannon.ChIP.input.nos_piwi_1.fastq.gz \
  -R Hannon.ChIP.input.nos_piwi_2.fastq.gz \
  -g dm3 \
  -B \
  -x 100 \
  -m \
  -M region_of_interest1.bed,region_of_interest2.bed \
  -o Hannon.ChIP.H3k9me3.nos_piwi.piPipes_out \
  1> Hannon.ChIP.H3k9me3.nos_piwi.piPipes.stdout \
  2> Hannon.ChIP.H3k9me3.nos_piwi.piPipes.stderr
```

Interpretation of the output files

The output folder should contain the following folders:

```
# folder raw data for mega analysis
aggregate_output/
# bigWig files for UCSC genome browser
bigWig/
# BAM files for genome mapping
genome_mapping/
# outputs from MACS2, the peak calling program used
macs2_peaks_calling/
# log
nos_piwi.callpeak.log
# pdfs for meta-analysis
pdfs/
```

`genome_mapping/` contains `bam` files for genomic alignments. Note that input and IP are aligned separately.

** Based on the option of `-u -m -e`, the bam file could contain different conteins**

- if `-u` is used (by default), reads are mapped to the genome by `bowtie2`, the bam file contains only the unique mapper
- if `-m` is used, for each of the multi-mapper reads, `bowtie2` will randomly report one alignment from the best alignments.
- if `-e` is used, reads are aligned to the genome using `bowtie` with `-a -m 100 --best --strata` (to report best alignments for multi-mappers with <= 100 loci). Then the bam file will be used by `csem`, which uses Expectation–maximization algorithm to allocate multi-mappers. `piPipes` then filter the alignments and only keep the ones with

crem posterior higher than 0.5. The reason to use `bowtie` instead of `bowtie2` is that `crem` currently is incompatible with `bowtie2` output.

Note that `bowtie` is fundamentally a string matching algorithm, it can allow up to 3 mismatches and no indels. `bowtie2` uses score matrix based alignment algorithm and are tolerable with more mismatches and indels. Thus for long reads (>70), we recommend using `-u`. For targets enriched in repetitive region, including H3K9me2/3 or Rhino, we recommend to use `-m` so that we don't lose important information.

```
# alignments for the input
Hannon.ChIP.H3K9me3.dm3.Input.b2.log
Hannon.ChIP.H3K9me3.dm3.Input.b2.sorted.bam
Hannon.ChIP.H3K9me3.dm3.Input.b2.sorted.bam.bai

# alignments for the IP
Hannon.ChIP.H3K9me3.dm3.IP.b2.log
Hannon.ChIP.H3K9me3.dm3.IP.b2.sorted.bam
Hannon.ChIP.H3K9me3.dm3.IP.b2.sorted.bam.bai
```

`macs2_peaks_calling/` contains output files from `MACS2`.

```
# bedGraph files for IP and input alignments
Hannon.ChIP.H3K9me3_treat_pileup.bdg
Hannon.ChIP.H3K9me3_control_lambda.bdg

# loci of peaks called by MACS2
Hannon.ChIP.H3K9me3_peaks.xls

# information on the "broad" regions called by MACS2 in BED format
Hannon.ChIP.H3K9me3_peaks.broadPeak

# information on both the "broad" regions and narrow peaks
Hannon.ChIP.H3K9me3_peaks.gappedPeak

# log for peak calling, --SPMR option is used so the signal was normalized
Hannon.ChIP.H3K9me3.callpeak.SPMR.log

# bedGraph files for signal enrichment (IP over input) with three different methods
Hannon.ChIP.H3K9me3.ppois.bdg
Hannon.ChIP.H3K9me3.FE.bdg
Hannon.ChIP.H3K9me3.logLR.bdg
```

`bigWig/` contains `bigWig` files for UCSC genome browser

```
# bigWig files for IP/input signal enrichment from three different methods.
Hannon.ChIP.H3K9me3.ppois.bigWig
Hannon.ChIP.H3K9me3.FE.bigWig
Hannon.ChIP.H3K9me3.logLR.bigWig
```

`aggregate_output/` contains accumulated signal around start site (TSS), end site (TES), and the whole region (meta). They are generated by `bwtool aggregate`.

By default, **piPipes** extends 100 bp around the start/end and calculate the accumulated signals. The size can be toggled using `-x` option. For meta-gene analysis, the extension length is the same as the gene body, so it is not affected by `-x`.

```
# piPipes used three different methods to calculate the signal enrichment of IP/input,
# Poisson P value, Folder Enrichment and logLR.
# for each position from -100 to 100
# the data is used to draw pdf
```

```
$ head piRNA_Cluster_42AB.starts.txt
piRNA_Cluster_42ABstart Poisson P value -100    0.121330
piRNA_Cluster_42ABstart Fold Enriched   -100    0.452620
piRNA_Cluster_42ABstart logLR   -100    -0.115620
piRNA_Cluster_42ABstart Poisson P value -99 0.121330
piRNA_Cluster_42ABstart Fold Enriched   -99 0.452620
piRNA_Cluster_42ABstart logLR   -99 -0.115620
piRNA_Cluster_42ABstart Poisson P value -98 0.121330
piRNA_Cluster_42ABstart Fold Enriched   -98 0.452620
piRNA_Cluster_42ABstart logLR   -98 -0.115620
```

Example 2. Run dual-sample mode to compare ChIP-seq libraries from two samples

Run single-sample mode for each library

```
# download all the data
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.input.nos_white    SRR646592
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.H3K9me3.nos_white  SRR646593
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.input.nos_piwi     SRR646594
fastq-dump -F --split-3 --gzip -A Hannon.ChIP.H3K9me3.nos_piwi   SRR646595

# run single-sample pipeline for control sample
piPipes chip \
  -l Hannon.ChIP.H3K9me3.nos_white_1.fastq.gz \
  -r Hannon.ChIP.H3K9me3.nos_white_2.fastq.gz \
  -L Hannon.ChIP.input.nos_white_1.fastq.gz \
  -R Hannon.ChIP.input.nos_white_2.fastq.gz \
  -g dm3 \
  -c 24 \
  -B \
  -x 100 \
  -o Hannon.ChIP.H3K9me3.nos_white.piPipes_out \
  1> Hannon.ChIP.H3K9me3.nos_white.piPipes.stdout \
  2> Hannon.ChIP.H3K9me3.nos_white.piPipes.stderr

# run single sample pipeline for experimental sample
piPipes chip \
  -l Hannon.ChIP.H3K9me3.nos_piwi_1.fastq.gz \
  -r Hannon.ChIP.H3K9me3.nos_piwi_2.fastq.gz \
  -L Hannon.ChIP.input.nos_piwi_1.fastq.gz \
  -R Hannon.ChIP.input.nos_piwi_2.fastq.gz \
  -g dm3 \
  -c 24 \
  -B \
  -x 100 \
  -o Hannon.ChIP.H3K9me3.nos_piwi.piPipes_out \
  1> Hannon.ChIP.H3K9me3.nos_piwi.piPipes.stdout \
  2> Hannon.ChIP.H3K9me3.nos_piwi.piPipes.stderr
```

Run dual-sample pipeline using the outputs of single-sample mode

```
piPipes chip2 \
  -a Hannon.ChIP.H3K9me3.nos_white.piPipes_out \
  -b Hannon.ChIP.H3K9me3.nos_piwi.piPipes_out \
```

```
  -g dm3 \
  -c 24 \
  -o Hannon.ChIP.H3K9me3.nos_white.vs.nos_piwi \
  -A nos_white \
  -B nos_piwi \
  1> Hannon.ChIP.H3K9me3.nos_white.vs.nos_piwi.stdout \
  2> Hannon.ChIP.H3K9me3.nos_white.vs.nos_piwi.stderr
```

Interpretation of the output files

```
# pdfs output
pdfs/
# peak calling output from sample 1, without in-library normalization
macs2_peaks_calling_no_normalization_nos_white/
# peak calling output from sample 2, without in-library normalization
macs2_peaks_calling_no_normalization_nos_piwi/
# differential peaks calling between sample 1 and sample 2
differential_peaks_calling/
# aggregate analysis on differentially expressed peaks identified
aggregate_output/
```

`pdfs/` folder contains the TSS/TES/meta analysis on the newly identified regions that are differentially enriched/depleted in mutant.

`macs2_peaks_calling_no_normalization_nos_white/` contains peak calling of `MACS2` on control sample

`macs2_peaks_calling_no_normalization_nos_piwi/` contains peak calling of `MACS2` on experimental sample

`differential_peaks_calling/` contains outputs of `MACS2` on differential peak calling

```
# log file
nos_.nos_white_vs_nos_piwi.bdgdiff.log

# common peaks enriched in IP over input for both samples, in BED format
nos_.nos_white_vs_nos_piwi_c3.0_common.bed

# peaks enriched in sample A, in BED format
nos_.nos_white_vs_nos_piwi_c3.0_cond1.bed

# peaks enriched in sample B, in BED format
nos_.nos_white_vs_nos_piwi_c3.0_cond2.bed
```

`aggregate_output/` contains text outputs for TSS/TES/meta analysis on the newly identified regions that are differentially enriched/depleted in mutant. The data has been used to draw the figures in `pdfs/` folder.

Please see example figures from the Github Wiki page or our manuscript.

Flowchart and example figures from our manuscript

piPipes Genome-seq pipeline

This document explains how to run **piPipes** Genome-seq pipeline and how to interpret the output.

This pipeline provides analysis on transposon insertion/deletion as well as general structural variation (SV) events using paired-end Genome-seq reads generated by Next Generation Sequencing.

Example. Run Genome-seq library

Install the fly dm3 genome, if you haven't done so

```
# require internet access
piPipes install -g dm3
```

Download Genome-seq sample data from NCBI SRA

```
# use fastq-dump from SRATools (http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software)
# to download data and convert to fastq; require internet access
fastq-dump -F --split-3 --gzip -A Theurkauf.GSQ.w1xHar.21day.ovary SRR333512
```

Check usage message

```
piPipes dna
# or
piPipes dna -h
```

Using default parameters

```
# -l: left read of  Genome-seq
# -r: right read of Genome-seq
# -g: genome to use
# -o: output directory
piPipes dna \
  -l Theurkauf.GSQ.w1xHar.21day.ovary_1.fastq.gz \
  -r Theurkauf.GSQ.w1xHar.21day.ovary_2.fastq.gz \
  -g dm3 \
  -o Theurkauf.GSQ.w1xHar.21day.ovary.piPipes_out \
  1> Theurkauf.GSQ.w1xHar.21day.ovary.piPipes.stdout \
  2> Theurkauf.GSQ.w1xHar.21day.ovary.piPipes.stderr
```

Run the pipeline with optional parameter

```
# -l: left read of  Genome-seq
# -r: right read of Genome-seq
# -g: genome to use
# -o: output directory
# -d: VCF filtering depth, passed to "vcfutils.pl varFilter -D" and "retroseq.pl -call -depth"
# -e: Transgene sequence in fasta format. piPipes calls TEMP to identify new transposon
#     insertion site
```

```
# -M: Use mrFast and VariationHunter. mrFast requires large amount of memory so
#     it's off by default.
piPipes dna \
  -l Theurkauf.GSQ.w1xHar.21day.ovary_1.fastq.gz \
  -r Theurkauf.GSQ.w1xHar.21day.ovary_2.fastq.gz \
  -g dm3 \
  -d 200 \
  -e P.fa \
  -M  \
  -o Theurkauf.GSQ.w1xHar.21day.ovary.piPipes_out \
  1> Theurkauf.GSQ.w1xHar.21day.ovary.piPipes.stdout \
  2> Theurkauf.GSQ.w1xHar.21day.ovary.piPipes.stderr
```

Interpretation of the output files

The output should contain the following directories:

```
mrFast_VariationHunter_output/
bigWig/
TEMP_output/
break_dancer_out/
bwa_bcftools_output/
retroSeq_discovering/
pdfs/
```

`mrFast_VariationHunter_output/` contains the output of `mrFast` and `Variation Hunter`. By default, `mrFast` and `Variation Hunder` are not ran because `mrFast` uses memory that is proportional to the **input Fastq**. In most cases, we were unable to finish running.

`bwa_bcftools_output/` contains the genome alignments as well as SNP calling using `bcftools`.

```
# genome alignment by BWA MEM algorithm and sorted by coordinates
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.log
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.bam
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bam
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bam.bai
# SNP calling output
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.var.raw.bcf
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.var.flt.vcf
# bedGraph, used to produce the bigWig file
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bdg

# genome alignment by BWA ALN algorithm, only used by TEMP
Theurkauf.GSQ.w1xHar.21day.1_sequence.sai
Theurkauf.GSQ.w1xHar.21day.2_sequence.sai
Theurkauf.GSQ.w1xHar.21day.bwa-aln.sorted.bam
Theurkauf.GSQ.w1xHar.21day.bwa-aln.sorted.bam.bai
```

`bigWig/` directory contains files that can be used by UCSC genome browser.

```
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bigWig
```

`TEMP_output/` contains output by TEMP. This is the algorithm used in:

Adaptation to P element transposon invasion in Drosophila melanogaster. *Cell.* 2011 Dec 23;147(7):1551-63.

If feeding this pipeline with `-e` option and a Fasta file, `TEMP` will be used to locate the insertion of this sequence in the genome. Our lab constantly uses this function to map transgene insertion site.

```
# intermediates files
dm3.repBase.fa
/project/umw_phil_zamore/hanb/tmp/GENOME/Theurkauf.GSQ.w1xHar.21day.ovary_1.piPipes_out/bwa_bcftools_output/T
/project/umw_phil_zamore/hanb/tmp/GENOME/Theurkauf.GSQ.w1xHar.21day.ovary_1.piPipes_out/bwa_bcftools_output/T
dm3.repBase.fa.bwt
dm3.repBase.fa.pac
dm3.repBase.fa.ann
dm3.repBase.fa.amb
dm3.repBase.fa.sa
Theurkauf.GSQ.w1xHar.21day.bwa-aln.unpair.uniq.transposons.bed
Theurkauf.GSQ.w1xHar.21day.bwa-aln.unpair.uniq.transposons.filtered.bed
Theurkauf.GSQ.w1xHar.21day.bwa-aln.uniq.transposons.filtered.wGap.class.bed
Theurkauf.GSQ.w1xHar.21day.bwa-aln.insertion.bp.bed
Theurkauf.GSQ.w1xHar.21day.bwa-aln.clipped.reads.aln
Theurkauf.GSQ.w1xHar.21day.bwa-aln.insertion.refined.bp

# files with transposon insertion events
Theurkauf.GSQ.w1xHar.21day.bwa-aln.insertion.refined.bp.summary

$ head Theurkauf.GSQ.w1xHar.21day.bwa-aln.insertion.refined.bp.summary
Chr Start   End TransposonName  TransposonDirection Class   VariantSupport  Frequency   Junction1   Junction1
chr2L   27827   28327   DM176   sense   singleton   1   0.0222  28077   0   28077   0   1   0
chr2L   44874   45374   BLOOD_I sense   singleton   1   0.0294  45124   0   45124   0   1   0
chr2L   290721  291221  POGON1  antisense   singleton   3   0.2500  291182  1   291185  1   0   1
chr2L   290721  291221  POGO    antisense   singleton   3   0.2500  291182  1   291185  1   0   1
chr2L   362397  362897  ROO_LTR sense   singleton   1   0.0370  362647  0   362647  0   0   1
chr2L   493282  493782  R1_DM   antisense   singleton   1   0.0909  493532  0   493532  0   0   1
chr2L   512615  512722  LTRMDG3_DM  sense   1p1 8   0.3200  512702  3   512702  0   1   4
chr2L   639122  639211  BEL_LTR antisense   1p1 12  0.4800  639191  2   639195  2   3   5
chr2L   735384  735884  ROO_LTR antisense   2p  7   0.3500  735883  2   735888  3   0   2
```

`retroSeq_discovering/` contains output of retroSeq, which is another algorithm seeking transposon movement

```
# intermediate file
exd.file
Theurkauf.GSQ.w1xHar.21day.dm3.bwa.sorted.retroSeq
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bam.vcf.PE.vcf.candidates

# this two files contain the identified loci
# see https://github.com/tk2/RetroSeq/wiki/RetroSeq-Tutorial for more information
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bam.vcf.PE.vcf
Theurkauf.GSQ.w1xHar.21day.dm3.bwa-mem.sorted.bam.vcf.PE.vcf.bed
```

`break_dancer_out/` contains outputs of Break Dancer for discovering general structural variation in the genome. It is **NOT** specialized in transposon related insertion/deletion.

```
# intermediate files
Theurkauf.GSQ.w1xHar.21day.breakdancer
Theurkauf.GSQ.w1xHar.21day.breakdancer.NA.2.fastq
Theurkauf.GSQ.w1xHar.21day.breakdancer.NA.1.fastq
Theurkauf.GSQ.w1xHar.21day.breakdancer.SV.bed
# outputs
# please see https://github.com/kenchen/breakdancer for detailed explanation
Theurkauf.GSQ.w1xHar.21day.breakdancer.out
Theurkauf.GSQ.w1xHar.21day.breakdancer.out.for_Circos
```

`pdfs` contains the output pdf Circos plot representing overall looking of `TEMP` `retroSeq` and `Break Dancer`.

```
# from innert to outer: break dancer, retroSeq and TEMP
# piRNA cluster loci has been marked
Theurkauf.GSQ.w1xHar.21day.summary.circos.pdf
```

Please see example figures from the Github Wiki page or our manuscript.

Flowchart and example figures from our manuscript

Benchmark of piPipes

In this document we presents the running time and space usage by individual pipeline of `piPipes`. The figure can be downloaded from here or on the Wiki.

Details

All the runnings were performed on Massachusetts Green High Performance Computing Cluster using `piPipes` commit `ab50e8a2fae33edefcb7749e95cbf54a600c1c50`.

Small RNA-seq

We randomly sampled `N` millions of reads from an unpublished HiSeq SE50 small RNA-seq library with 27,990,838 reads and ran `piPipes` small RNA pipeline with 8 CPUs.

```
for i in `seq 1 3 26`; do
    seqtk sample -s$((RANDOM%100)) $SMALLRNA_FQ ${i}000000 | \
        gzip > ${i}M.fq.gz && \
        date +"%m-%d-%k-%M" > ${i}.time && \
    piPipes small \
        -i ${i}M.fq.gz \
        -g dm3 \
        -o ${i}M.out && \
    date +"%m-%d-%k-%M" >> ${i}.time && \
    du -skh ${i}M.out > ${i}.size && \
    rm -rf ${i}M.out ${i}M.fq.gz
done
```

RNA-seq

We randomly sampled `N` millions of reads from an unpublished HiSeq PE100 RNA-seq library with 15,963,640 pairs and ran `piPipes` RNA-seq pipeline with 8 CPUs.

```
for i in `seq 1 15`; do
    SEED=$((RANDOM%100)) && \
    seqtk sample -s $SEED $RNA_FQ1 ${i}000000 | \
        gzip > ${i}M.r1.fq.gz && \
    seqtk sample -s $SEED $RNA_FQ2 ${i}000000 | \
        gzip > ${i}M.r2.fq.gz && \
    date +"%m-%d-%k-%M" > ${i}.time && \
    piPipes rna \
        -l ${i}M.r1.fq.gz \
        -r ${i}M.r2.fq.gz \
        -g dm3 \
        -o ${i}M.out && \
    date +"%m-%d-%k-%M" >> ${i}.time && \
    du -skh  ${i}M.out > ${i}.size && \
    rm -rf ${i}M.out ${i}M.r1.fq.gz ${i}M.r2.fq.gz
done
```

Degradome-seq

We randomly sampled `N` millions of reads from an unpublished HiSeq PE100 Degradome-seq library with 15,963,640 pairs. Then we ran `piPipes` Degradome-seq pipeline with 8 CPUs, and with small RNA library that has 23,712,713 genome-mappable reads (small RNA-seq data wasn't sampled).

```
for i in ‘seq 1 7‘; do
    SEED=$((RANDOM%100)) && \
    seqtk sample -s$SEED $DEG_FQ1 ${i}000000 | \
        gzip > ${i}M.r1.fq.gz && \
    seqtk sample -s$SEED $DEG_FQ2 ${i}000000 | \
        gzip > ${i}M.r2.fq.gz && \
    date +"%m-%d-%k-%M" > ${i}.time && \
    piPipes deg \
        -l ${i}M.r1.fq.gz \
        -r ${i}M.r2.fq.gz \
        -g dm3 \
        -o ${i}M.out \
        -s $SMALL_RNA_OUTPUT && \
    date +"%m-%d-%k-%M" >> ${i}.time && \
    du -skh  ${i}M.out > ${i}.size && \
    rm -rf ${i}M.out ${i}M.r1.fq.gz ${i}M.r2.fq.gz
done
```

ChIP-seq

We randomly sampled `N` millions of reads from an unpublished HiSeq PE100 ChIP-seq library with 17,980,776/180,11,302 pairs for INPUT and IP. Then we ran `piPipes` ChIP-seq pipeline with 8 CPUs.

```
for i in ‘seq 2 2 10‘; do
    SEED=$((RANDOM%100)) && \
    seqtk sample -s$SEED $CHIP_INPUT_1 ${i}000000 | \
        gzip > ${i}M.input.r1.fq.gz && \
    seqtk sample -s$SEED $CHIP_INPUT_2 ${i}000000 | \
        gzip > ${i}M.input.r2.fq.gz && \
    seqtk sample -s$SEED $CHIP_IP_1 ${i}000000 | \
        gzip > ${i}M.IP.r1.fq.gz && \
    seqtk sample -s$SEED $CHIP_IP_2 ${i}000000 | \
        gzip > ${i}M.IP.r2.fq.gz && \
  date +"%m-%d-%k-%M" > ${i}.time && \
  piPipes chip \
    -l ${i}M.IP.r1.fq.gz \
    -r ${i}M.IP.r2.fq.gz \
    -L ${i}M.input.r1.fq.gz \
    -R ${i}M.input.r2.fq.gz \
    -g dm3 \
    -o ${i}M.out && \
  date +"%m-%d-%k-%M" >> ${i}.time && \
  du -skh  ${i}M.out > ${i}.size && \
  rm -rf ${i}M.input.r1.fq.gz ${i}M.input.r2.fq.gz ${i}M.IP.r1.fq.gz ${i}M.IP.r2.fq.gz ${i}M.out
  done
```

Genome-seq

We randomly sampled `N` millions of reads from a published (SRR333512) HiSeq PE100 Genome-seq library with 18,042,217 reads and ran `piPipes` Genome-seq pipeline with 8 CPUs and without running `mrfast` and `VariationHunter`.

```
for i in ‘seq 2 2 10‘; do
    SEED=$((RANDOM%100)) && \
    seqtk sample -s$SEED $GENOME_FQ1 ${i}000000 | \
        gzip > ${i}M.r1.fq.gz && \
    seqtk sample -s$SEED $GENOME_FQ2 ${i}000000 | \
        gzip > ${i}M.r2.fq.gz && \
```

```
    date +"%m-%d-%k-%M" > ${i}.time && \
    piPipes dna \
        -l ${i}M.r1.fq.gz \
        -r ${i}M.r2.fq.gz \
        -g dm3 \
        -o ${i}M.out && \
    date +"%m-%d-%k-%M" >> ${i}.time && \
    du -skh  ${i}M.out > ${i}.size && \
    rm -rf ${i}M.out ${i}M.r1.fq.gz ${i}M.r2.fq.gz
done
```