

ssviz: A small RNA-seq visualizer and analysis toolkit

Diana HP Low

Institute of Molecular and Cell Biology
Agency for Science, Technology and Research (A*STAR), Singapore
dlow@imcb.a-star.edu.sg

May 1, 2024

Contents

1	Introduction	1
1.1	A typical small RNA sequencing analysis	1
2	Installing and loading the ssviz package	2
3	Using <i>ssviz</i>	3
3.1	Loading the <i>ssviz</i> package	3
3.2	Reading bam files	3
3.3	Read counts in small RNA sequencing	4
3.4	Plotting bam properties	4
3.5	Plotting region density	6
3.6	piRNA ping-pong analysis	6
3.7	Computing nucleotide frequency (composition)	7
4	Further work	7
	References	8

1 Introduction

Small RNA sequencing enables the discovery and profiling of microRNAs, piRNAs and other non-coding RNA for any organism, even without prior genome annotation. The *ssviz* package is intended firstly as a visual aid, and secondly to provide more specialized analysis catering for either miRNA or piRNA analysis.

1.1 A typical small RNA sequencing analysis

To understand the workings and conventions of this package, Figure 1 outlines a typical workflow for a small RNA sequencing run. Here it is assumed that the data is produced on the Illumina platform, either GAIIX or HiSeq 2000.

First, pre-processing on the raw reads is done with tools like fastx-toolkit. Small RNA reads are typically shorter than the high-throughput sequencing length, so there is a need for adapter trimming and removal of adapter contaminants. Also, as reads are often repeated, it is thus favourable to collapse identical sequences into a single read (but keeping note of the total number of reads). Based

on `fastx-toolkit`, collapsed read names are in the form of `readname-readcount`. This point is crucial as `ssviz` will take into account this naming convention in plots and computations.

After pre-processing, reads are then mapped to list of contaminants (snoRNA, snRNA, tRNA, etc) and then either to the genome and small RNA databases (eg. miRBank, RepBase, piRNABank) or both. Once mapped, tools like `ssviz` can be used to analyze the resulting data.

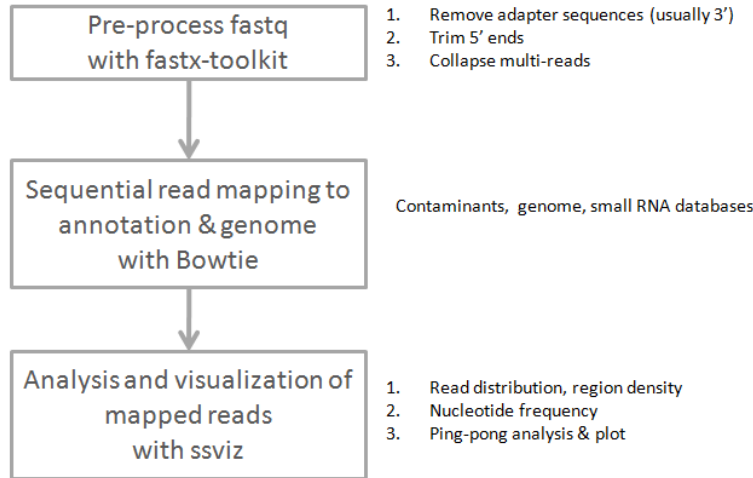


Figure 1: A typical small RNA sequencing workflow

2 Installing and loading the `ssviz` package

We recommend that users install the package via Bioconductor, since this will automatically detect and install all required dependencies. The Bioconductor installation procedure is described at <http://www.bioconductor.org/docs/install/>. To install `ssviz`, launch a new R session, and in a command terminal either type or copy/paste:

```
if (!requireNamespace("BiocManager", quietly=TRUE))
  install.packages("BiocManager")
BiocManager::install("ssviz")
```

3 Using *ssviz*

3.1 Loading the *ssviz* package

To load the *ssviz* package in the R environment, simply type:-

```
library(ssviz)
```

The *ssviz* package also includes all the data generated in this vignette for easy reference. Data can be loaded by typing:

```
data(ssviz)
```

3.2 Reading bam files

ssviz analyzes bam files that have been mapped either to a genome or to small RNA seq annotations. Bam files can be loaded into the workspace using `readBam`. The package comes with two example datasets (control and treatment). `readBam` is a wrapper for the `scanBam` function and writes the bam file contents into a convenient data frame. For more information, please refer to the `Rsamtools` package.

```
bam.files <- dir(system.file("extdata", package = "ssviz"), full = TRUE, patt = "bam$")
ctrlbam <- readBam(bam.files[1])
treatbam <- readBam(bam.files[2])
```

Firstly, we can view the contents of the bam file, by simply typing the name of the object, for example `ctrlbam`.

```
ctrlbam
## DataFrame with 9976 rows and 13 columns
##      qname      flag      rname      strand      pos      qwidth      mapq
##      <character> <integer> <factor> <factor> <integer> <integer> <integer>
## 1      74300-21      16      chr1      -      3005526      24      255
## 2      95290-16      16      chr1      -      3005696      23      255
## 3      114695-13      0      chr1      +      3005832      24      255
## 4      97521-15      16      chr1      -      3008048      30      255
## 5      100554-15      16      chr1      -      3008818      29      255
## ...      ...      ...      ...      ...      ...      ...      ...
## 9972     111742-13      16      chr1      -      16828975      27      255
## 9973     24886-63      0      chr1      +      16833561      27      255
## 9974     102926-15      16      chr1      -      16834169      25      255
## 9975     122260-12      0      chr1      +      16838393      29      255
## 9976     108613-14      16      chr1      -      16838446      26      255
##      cigar      mrnm      mpos      isize      seq
##      <character> <factor> <integer> <integer> <DNAStrngSet>
## 1      24M      NA      NA      0      CATATTGTGC...TCCTTTGTGA
## 2      23M      NA      NA      0      GGTTCCTTCCAGCTTCTGGCTAT
## 3      24M      NA      NA      0      AATTTTCTGA...CCGCCAGACT
## 4      30M      NA      NA      0      TTCCAGTTTT...GCCTTTGTGA
## 5      29M      NA      NA      0      GTTTCACTGT...TAGTTTCTGT
## ...      ...      ...      ...      ...      ...
```


number of reads that was mapped, to make sure than the comparisons are on the same scale. Part (i) can be obtained directly from the loaded bam DataFrame (if `fastx-toolkit` was used). Part (ii) can be obtained as an output of the mapper, eg. `bowtie`. Typically this number would represent total number of reads sequenced, or mappable to the broadest encompassing index (the genome).

In the bam file, read length is represented by `qwidth`, direction by `strand` and region by `rname` and `pos`. For example, to plot the read length distribution (Figure 2):

```
plotDistro(list(ctrl.count), type = "qwidth", samplenames = c("Control"), ncounts = counts[1])
```

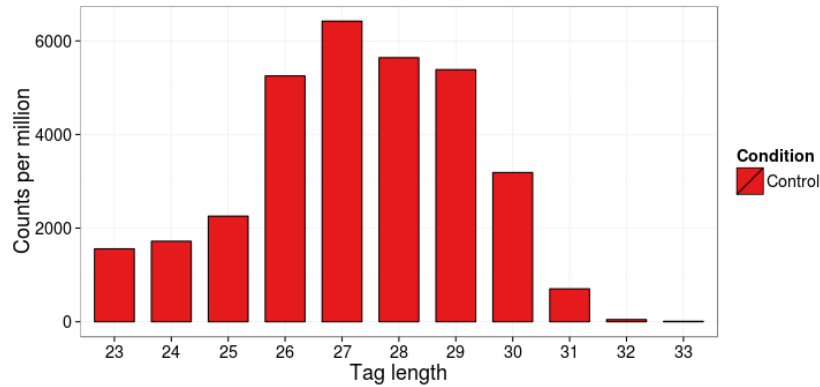


Figure 2: `plotDistro` with a single dataset

To compare two or more datasets, simply include them in the list (Figure 3).

```
plotDistro(list(ctrl.count, treat.count), type = "qwidth", samplenames = c("Control", "Treatment"), ncounts = counts)
```

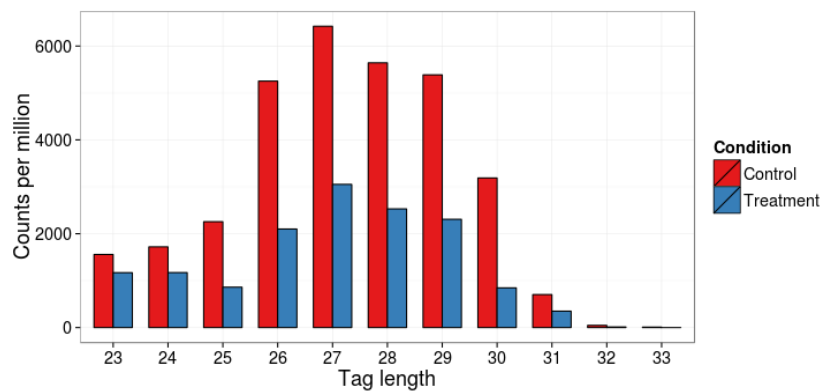


Figure 3: `plotDistro` with two datasets

3.5 Plotting region density

```
region<-'chr1:3015526-3080526'  
plotRegion(list(ctrl.count), region=region)
```

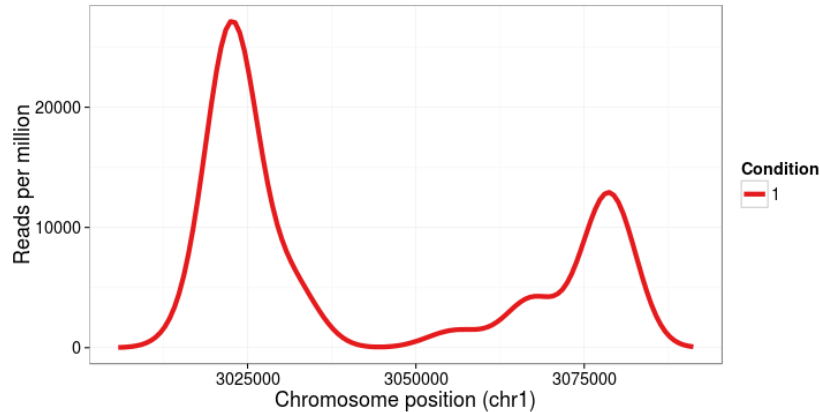


Figure 4: plotRegion

3.6 piRNA ping-pong analysis

PIWI-interacting RNAs (piRNAs) are 23-30-nucleotide-long small RNAs that act as sequence-specific silencers of transposable elements in animal gonads (Kawaoka 2011). The ping-pong mechanism is a proposed method for the amplification of primary Piwi-associated RNAs (piRNAs), which leads to the production of new primary piRNAs from their precursor transcripts, which eventually amplifies the pool of both primary and secondary piRNAs (Brennecke 2007). This positive feedback loop is a secondary biogenesis mechanism that requires complementary transcripts to a pre-existing pool of piRNAs.

Piwi proteins retain the endoribonuclease or Slicer activity that allows them to cleave targets between position 10 and position 11 of their bound piRNA. This cleavage defines the 5' end of a secondary piRNA that is generated from the transposon transcript. Because a very high proportion of piRNAs have a uridine (U) at the first position and because the complementarity between piRNAs and targets is expected to be nearly perfect, secondary piRNAs typically have adenosines at position 10, which base-pairs with the U at the first position of the piRNA. This is reflected as a sharp peak at 10nts when frequency is plotted against overlap length, and also can be seen in the nucleotide frequency plot in the next section.

To compute the overlaps between the sense and anti-sense (amplified) piRNAs, we leverage on the positional information contained in the bam file of "+" strand reads and "-" strand reads, calculates and plots the frequency of overlap (up to the length of the read).

```
pp.ctrl<-pingpong(pctrlbam.count)  
plotPP(list(pp.ctrl), samplenames=c("Control"))
```

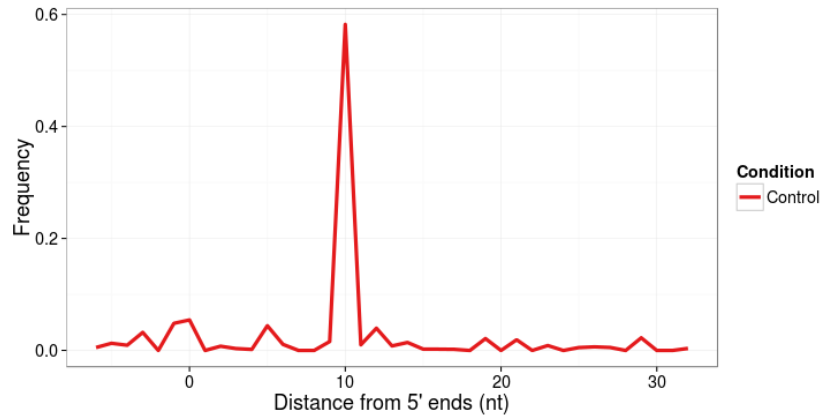


Figure 5: plotPP

3.7 Computing nucleotide frequency (composition)

A known property of primary piRNA is a marked 5-prime uridine bias (Brennecke 2007). The `ntfreq` function allows the user to compute the frequency of nucleotides over a chosen length.

```
pctrlbam.count<-getCountMatrix(pctrlbam)
freq.ctrl<-ntfreq(pctrlbam.count, ntlength=10)
plotFreq(freq.ctrl)
```

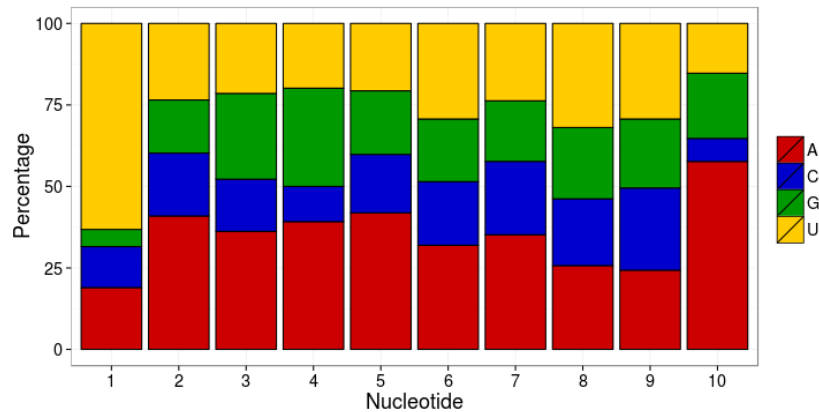


Figure 6: plotFreq

4 Further work

It is the hope that with feedback from the community, `ssviz` would further develop to support and improve the multi-faceted analysis in small RNA sequencing including miRNA target identification and novel RNA discovery. As such, output and working data formats have been kept to the convention most widely utilized for sequencing analysis in R to ensure and enhance cross-compatibility and usage.

References

- [1] FASTX-Toolkit: FASTQ/A short-reads pre-processing tools
- [2] Ruvkun, G. Molecular Biology: Glimpses of a Tiny RNA World. Science 2001, 294, 797-799.
- [3] Brennecke J, Aravin AA, Stark A, Dus M, Kellis M, Sachidanandam R, Hannon GJ. Discrete small RNA-generating loci as master regulators of transposon activity in Drosophila. Cell. 2007;128:1089-1103.
- [4] Thomson T, Lin H. The biogenesis and function of PIWI proteins and piRNAs: progress and prospect. Annu. Rev. Cell. Dev. Biol. 2009;25:355-376.
- [5] Shinpei Kawaoka, Yuji Arai, Koji Kadota, et al. Zygotic amplification of secondary piRNAs during silkworm embryogenesis RNA (2011), 17:00-00

```
sessionInfo()

## R version 4.4.0 RC (2024-04-16 r86468)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 22.04.4 LTS
##
## Matrix products: default
## BLAS: /home/biocbuild/bbs-3.20-bioc/R/lib/libRblas.so
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
## [3] LC_TIME=en_GB LC_COLLATE=C
## [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8 LC_NAME=C
## [9] LC_ADDRESS=C LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4 stats graphics grDevices utils datasets methods
## [8] base
##
## other attached packages:
## [1] ssviz_1.39.0 RColorBrewer_1.1-3 ggplot2_3.5.1
## [4] reshape_0.8.9 Rsamtools_2.21.0 Biostrings_2.73.0
## [7] XVector_0.45.0 GenomicRanges_1.57.0 GenomeInfoDb_1.41.0
## [10] IRanges_2.39.0 S4Vectors_0.43.0 BiocGenerics_0.51.0
##
## loaded via a namespace (and not attached):
## [1] gtable_0.3.5 jsonlite_1.8.8 dplyr_1.1.4
## [4] compiler_4.4.0 highr_0.10 crayon_1.5.2
## [7] tidyselect_1.2.1 Rcpp_1.0.12 bitops_1.0-7
## [10] parallel_4.4.0 scales_1.3.0 BiocParallel_1.39.0
## [13] R6_2.5.1 plyr_1.8.9 generics_0.1.3
```



```
## [16] knitr_1.46          tibble_3.2.1      munsell_0.5.1
## [19] GenomeInfoDbData_1.2.12 pillar_1.9.0      rlang_1.1.3
## [22] utf8_1.2.4           xfun_0.43         cli_3.6.2
## [25] formatR_1.14         withr_3.0.0       magrittr_2.0.3
## [28] zlibbioc_1.51.0      grid_4.4.0        lifecycle_1.0.4
## [31] vctrs_0.6.5          evaluate_0.23     glue_1.7.0
## [34] codetools_0.2-20     fansi_1.0.6       colorspace_2.1-0
## [37] httr_1.4.7           pkgconfig_2.0.3   tools_4.4.0
## [40] UCSC.utils_1.1.0
```