PingPongPro v1.0 - User Manual

Sebastian Uhrig (suhrig@students.uni-mainz.de)

September 27, 2014

Table of contents

1. What's new?

v1.0:

This is the initial release of *PingPongPro*.

2. Introduction

Piwi-interacting RNAs (piRNAs) are a class of small non-coding RNAs, predominantly active in the germ line. There, they limit the detrimental effect of transposons on the genome. They do so by forming a complex with *Piwi* proteins. The complex binds and then cleaves mRNA molecules of active transposons. The resulting mRNA fragments induce the production of more piRNAs, thus reinforcing the anti-transposon response in a feed-forward loop called the "*ping-pong* cycle" [2][3].

A key question in *ping-pong* cycle research is identification of piRNAs which are amplified through the cycle. *PingPongPro* is a command-line tool for detecting *ping-pong* cycle activity in piRNA-Seq data.

3. Quick start

Pre-compiled binaries of *PingPongPro* can be downloaded from here:

http://sourceforge.net/projects/pingpongpro/files/

PingPongPro takes a file with aligned reads in *SAM* or *BAM* format as input (see http://samtools.sourceforge.net/ for more information). This file is passed to *PingPongPro* using the parameter `-i`. For example:

```
pingpongpro -i alignments.bam
```

The program then writes a list of putative *ping-pong* signatures to the file `ping-pong_signatures.tsv`. The column `FDR` in this file indicates how confident *PingPongPro* is that a given *ping-pong* signature is not just a random alignment of reads, but truly a product of *ping-pong* cycle activity. FDRs close to 0 are good and FDRs close to 1 are bad.

In addition, *PingPongPro* can find transposable elements that are suppressed through the *ping-pong* cycle, when run with the parameter `-T`:

```
pingpongpro -i alignments.bam -T 1000
```

PingPongPro produces a file named `predicted_transposons.tsv`, which contains a list of regions, which are possibly suppressed by the *ping-pong* cycle. The column `qValue` indicates *PingPongPro*'s confidence. Again, values close to 0 (ideally < 0.01) are good and values close to 1 are bad. The argument `1000` tells *PingPongPro* to merge *ping-pong* signatures within a window of 1000 bp into one region.

PingPongPro can check a list of transposable elements of interest for *ping-pong* cycle activity. The list must take the form of an annotation file in `GFF`, `GTF` or `BED` file format (see http://genome.ucsc.edu/FAQ/FAQformat.html for more information). It is passed to *PingPongPro* using the `-t` parameter:

```
pingpongpro -i alignments.bam -t annotated_transposons.bed
```

PingPongPro then generates a file named `transposons.tsv`, which assigns a q-value to every transposon given in the file `annotated_transposons.bed`. The file is to be interpreted alike the file `predicted_transposons.tsv`.

Lastly, *PingPongPro* can generate browser track files. These files can be loaded into the *UCSC Genome Browser* (http://genome.ucsc.edu/), *IGV* (https://www.broadinstitute.org/igv/home) or other genome browsers, for visualization of *ping-pong* signatures and *ping-pong* regions. *PingPongPro* must be launched with the switch `-b`, if browser tracks should be generated. For example:

```
pingpongpro -i alignments.bam -T 1000 -b
```

4. How *PingPongPro* works

4.1. Detecting *ping-pong* signatures

The *ping-pong* cycle produces short RNA molecules which are complementary at their 5' ends over a length of exactly ten nucleotides. In piRNA-Seq data, these molecules appear as stacks of reads on opposite strands, which overlap by ten nucleotides [5][7].

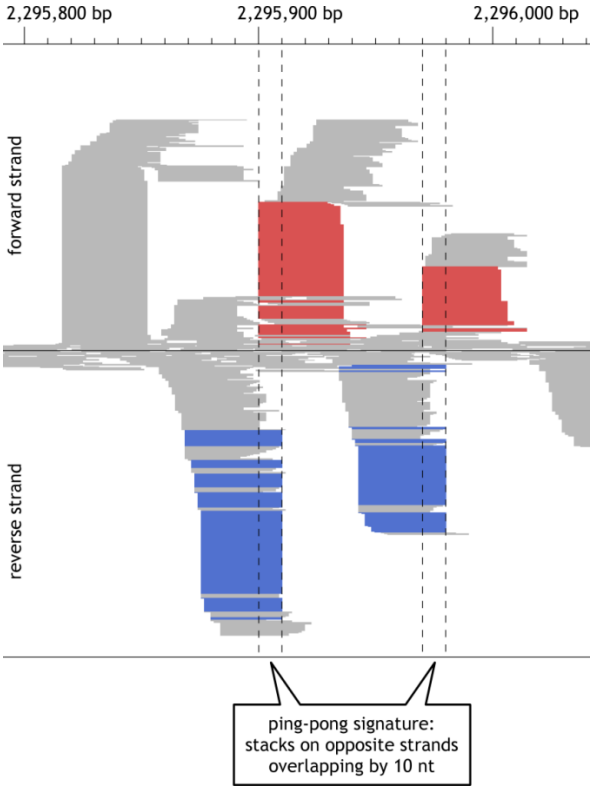

Figure 1: Two examples of *ping-pong* signatures.
The figure on the left shows the coverage around the genomic coordinate chr2L:2,295,900 (which is inside the transposon *copia*) of a piRNA-Seq dataset from *D. melanogaster* (available at the *NCBI Sequence Read Archive* [10] via the SRA accession number `SRR499784` [7]).
There are three stacks of reads on the forward strand and two stacks on the reverse strand which stick out. Four of them have been highlighted, because they are likely to result from *ping-pong* cycle activity. Reads thought to belong to these stacks are colored red on the forward strand and blue on the reverse strand. The stacks satisfy all the characteristics of a typical *ping-pong* signature:
They are all aligned at their 5' ends and the stacks on the forward strand overlap with the stacks on the reverse strand by ten nucleotides. The characteristic overlap is marked by dashed lines.
The left-most stack on the forward strand (not highlighted) is probably not to be attributed to the *ping-pong* cycle, because there is no stack on the reverse strand overlapping it by ten nucleotides.

PingPongPro scans a piRNA-Seq dataset for this pattern (also known as "*ping-pong* signatures"). *Ping-pong* signatures are not always as evident as in Figure 1. When the stacks are not that prominent, they are hard to distinguish from random alignments of reads, which only happen to overlap by ten nucleotides by chance. *PingPongPro* therefore evaluates a total of three properties that are characteristic for true *ping-pong* signatures:

- **Stack height:** High stacks of reads are the most reliable indicator of a true *ping-pong* signature.
- **Independence of the local coverage:** High stacks are not necessarily the result of *ping-pong* cycle activity. In regions with good coverage, the probability is increased that reads stack up on each other. Conversely, a relatively small stack likely is to be attributed to the *ping-pong* cycle, when there are hardly any other reads in the vicinity. In conclusion, stacks that are considerably higher than what can be explained by the local coverage (i.e., independent of the local coverage) are indicators of true *ping-pong* signatures.
- **Adenine bias:** piRNAs naturally have a preference for uracil at their 5' ends. Since piRNAs amplified through the *ping-pong* cycle are complementary over a length of ten nucleotides, these piRNAs also show a preference for adenine at position ten. Consequently, stacks of reads with adenine at position ten are a third indicator of true *ping-pong* signatures.

PingPongPro examines the properties of every putative *ping-pong* signature. In order to decide, if this combination of properties is peculiar to *ping-pong* signatures, it counts how many stacks there are in the dataset which have the same combination of properties, but which overlap by some other length than ten nucleotides. If there are many, then *PingPongPro* concludes that this combination of properties is easily satisfied by random alignments of reads and not something which is peculiar to *ping-pong* signatures. In this case, the signature gets a bad score. If, on the other hand, the signature has properties which are rarely seen in combination with arbitrary overlaps, then the signature gets a good score.

The score is directly based on relative frequencies. For example, if *PingPongPro* finds 100 stacks with an overlap of ten nucleotides and 18 stacks with the same combination of properties, but with different overlaps, then it estimates that about 18 % of the putative *ping-pong* stacks must be due to chance. It cannot tell which of the putative *ping-pong* stacks are the product of *ping-pong* cycle activity and which are due to chance. Therefore, all of them are assigned a score of 0.82 (100 % minus 18 %). As a result, only 82 % of the reads of these signatures will be counted and the rest is discarded. Ultimately, this weighting of signatures ensures that only those reads are counted which are probably not to be attributed to coincidence.

4.2. Finding transposable elements with *ping-pong* cycle activity

In order to decide if a given transposon is suppressed by the *ping-pong* cycle, *PingPongPro* counts the number of reads that are part of putative *ping-pong* signatures within the region of the transposon. The reads are weighted before counting, as explained in the previous section. *PingPongPro* then counts how many reads are part of stacks which overlap by some other length than ten nucleotides. Again, reads are weighted before counting. If the transposon is *ping-pong*-regulated, then the abundance of reads which are part of putative *ping-pong* signatures should be considerably higher than the abundance of reads which are part of arbitrarily overlapping stacks. The difference in abundance is normalized to a z-score, which, in turn, is converted to a p-value. The lower the p-value, the more confident *PingPongPro* is that the transposon shows signs of *ping-pong* cycle activity.

5. Installation

5.1. Using pre-compiled executables

Pre-compiled executables of *PingPongPro* do not require any installation. They only need to be downloaded and extracted from a compressed archive.

Linux

32-bit and 64-bit binary distributions of *PingPongPro* are available for *Linux* at:

http://sourceforge.net/projects/pingpongpro/files/

After downloading the respective `.tar.gz`-file for your architecture, extract it with `tar`:

```
tar -xvzf pingpongpro-*-linux*.tar.gz
```

Windows

32-bit and 64-bit binary distributions of *PingPongPro* are available for *Windows* at:

http://sourceforge.net/projects/pingpongpro/files/

After downloading the respective `.zip`-file for your architecture, extract it by right-clicking the `.zip`-file and choosing `Extract all …` from the context menu.

5.2. Compiling from source

PingPongPro builds on the *SeqAn* library (http://www.seqan.de/). You first need to download the *SeqAn* API, before you can compile *PingPongPro* from source.

Linux

Follow the instructions for installing the *SeqAn* API on *Linux*:

http://trac.seqan.de/wiki/Tutorial/GettingStarted/LinuxMakefiles

Skip the `Hello World!` example. Instead, create a sandbox for *PingPongPro*:

```
cd seqan-trunk
./util/bin/skel.py repository sandbox/workspace
./util/bin/skel.py app pingpongpro sandbox/workspace
```

Download the source code of *PingPongPro* from:

http://sourceforge.net/projects/pingpongpro/files/

Extract it to `seqan-trunk/sandbox/workspace`:

```
tar -xvzf pingpongpro-*-source.tar.gz
mv pingpongpro-*-source/* sandbox/workspace/apps/pingpongpro/
```

Compile *PingPongPro*:

```
cd ../seqan-trunk-build/debug
cmake .
make pingpongpro
```

The compiled executable can then be found in `seqan-trunk-build/debug/bin/`.

Windows

Download the contribution package for *SeqAn* on *Windows* as explained here:

> http://trac.seqan.de/wiki/HowTo/InstallContribsWindows

Follow the instructions for installing the *SeqAn* API on *Windows*:

> http://trac.seqan.de/wiki/Tutorial/GettingStarted/WindowsVisualStudio

Skip the `Hello World!` example. Instead, create a sandbox for *PingPongPro*:

```
cd seqan-trunk
python util\bin\skel.py repository sandbox\workspace
python util\bin\skel.py app pingpongpro sandbox\workspace
```

Download the source code of *PingPongPro* from:

> http://sourceforge.net/projects/pingpongpro/files/

Extract it to `seqan-trunk\sandbox\workspace` with a program that is capable of extracting files from `.tar.gz`-archives. Then move all the files to the newly created sandbox:

```
move pingpongpro-*-source\* sandbox\workspace\apps\pingpongpro\
```

Rerun `cmake` to register the newly created application:

```
cd ..\seqan-trunk-build\vs9
cmake .
```

Open the *Visual Studio* solution:

```
seqan-trunk-build\release\sandbox\workspace\apps\pingpongpro\
seqan_sandbox_workspace_apps_pingpongpro.sln
```

In *Visual Studio*, go to `Build` → `Build Solution (F7)`. The compiled executable can then be found in `seqan-trunk-build\debug\bin\Debug\`.

5.3. Additional pre-requisites for plotting

PingPongPro uses *R-Project* (http://www.r-project.org/) [8] to generate plots. If you want *PingPongPro* to generate plots (option `-p`), then install *R-Project*.

Linux

Instructions on how to install *R-Project* on *Linux* can be found here:

> http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Unix_002dalikes

Make sure `Rscript` is in your `PATH` environment variable, before running *PingPongPro*:

```
export PATH=/path/to/R_HOME/bin:$PATH
```

Windows

Instructions on how to install *R-Project* on *Windows* can be found here:

http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Windows

Make sure `Rscript` is in your `PATH` environment variable, before running *PingPongPro*:

```
set PATH=C:\path\to\R_HOME\bin;%PATH%
```

6. Command-line options

`-h, --help`

Print a short help about available command-line options and exit.

Default: off

`--version`

Print information about the version of *PingPongPro* and exit.

Default: off

`-b, --browserTracks`

By default, *PingPongPro* produces output in TSV format, only. When `-b` is specified, it creates additional browser track files, which are suitable for display in common genome browsers, such as the *UCSC Genome Browser* (http://genome.ucsc.edu/) [4] or the *Integrative Genomics Viewer* (https://www.broadinstitute.org/igv/home) [9]. See section 8.3 for more information on the types of tracks that are generated.

Default: off

`-s, --min-stack-height NUMBER_OF_READS`

Ping-pong signatures with one or both stacks smaller than `NUMBER_OF_READS` are omitted from the output. They are still considered to find suppressed transposons, however.

Default: 0

`-i, --input PATH`

Specify the path to a file with high-throughput sequencing data that *PingPongPro* shall scan for signs of *ping-pong* cycle activity. See section 7.1 for accepted file types.

On *Linux*, you can use the special file `/dev/stdin` to make *PingPongPro* read data from standard input.

Specifying this option is mandatory.

Default: none

`-l, --min-alignment-length LENGTH`
`-L, --max-alignment-length LENGTH`

PingPongPro ignores alignments given in the input file that are shorter or longer than the typical length of a piRNA (between 24 and 32 nucleotides). Use the parameter `-l` to define the minimum length and the parameter `-L` to define the maximum length of alignments that should not be ignored by *PingPongPro*.

Default: `24` for `-l` and `32` for `-L`

`-m, --multi-hits METHOD`

Very often, reads of piRNAs map to several loci, because the genome contains multiple copies of a transposable element that is targeted by a particular piRNA. Usually, such multi-mapping reads ("multi-hits") are ignored during analysis of RNA-Seq data, because it is unclear where the read really originated from. In piRNA-Seq data however, it would lead to a major fraction of the reads being lost, if multi-hits were ignored. *PingPongPro* can handle multi-hits in one of several ways:

`weighted`: A multi-hit is counted as the reciprocal of the number of alignments of the same read. For example, if a read has 5 alignments, it is counted as one fifth.

`unique`: No distinction is made between multi-hits and unique hits. Multi-hits are counted as one full read. This option is useful, if the input data has already been filtered for multi-hits, such that all but one randomly selected multi-hit have been deleted from the input file.

`discard`: Multi-hits are not counted at all. This option can be used to compare the effect of multi-hits on the output of *PingPongPro*.

Default: `weighted`

`-o, --output PATH`

Specify the path to a directory, where *PingPongPro* should put all output files. The directory is created, if it does not exist yet. The parent directory must exist, though.

Existing files are overwritten without warning. If this option is omitted, then *PingPongPro* writes all files to the current working directory. It is therefore recommended to run *PingPongPro* from an empty directory, if this option is not specified.

Default: current working directory

`-p, --plot`

This option instructs *PingPongPro* to generate plots. Refer to section 8.4 for information about the types of plots, which are generated.

PingPongPro uses *R-Project* for rendering of plots. Make sure the program `Rscript` is available in the `PATH` environment variable (see section 5.3).

Default: off

`-t, --transposons PATH`

PingPongPro can check a given list of transposons for *ping-pong* cycle activity. If `PATH` points to a file with coordinates of transposons, then *PingPongPro* will produce statistics about *ping-pong* cycle activity within the regions of the given transposons. See section 7.2 for file formats that are accepted as input. Section 8.2 describes the output that is produced.

This option can be specified multiple times. All given files will be processed, but only one output file is produced, in which the transposons of all input files are merged.

Default: none

`-T, --predict-transposons RANGE`

If a list of annotated transposons is not available, then *PingPongPro* can predict the regions of transposons (or more generally speaking, regions with *ping-pong* cycle activity). It does so by defining regions around *ping-pong* signatures dynamically. *Ping-pong* signatures that are not more than `RANGE` base pairs apart from each other will be merged into a contiguous region. Regions with less than two *ping-pong* signatures or a shorter length than 30 base pairs are not reported.

Default: `1000`

`-v, --verbose`

When this option is given, then *PingPongPro* reports its current progress and the time it took to run each step to standard error.

Default: off

7. Accepted input formats

7.1. High-throughput sequencing data

PingPongPro scans high-throughput sequencing data given via the parameter `-i` for signs of *ping-pong* cycle activity. Accepted file types for the high-throughput sequencing data are the *Sequence Alignment/Map* (*SAM*) format and the *Binary-Compressed Sequence Alignment/Map* (*BAM*) format. See http://samtools.sourceforge.net/ for more information on these file formats. Only single-end alignments can be processed by *PingPongPro*. Paired-end alignments are treated like single-end alignments. The *SAM* / *BAM* input file does not need to be sorted or indexed.

7.2. Annotations of transposable elements

PingPongPro can check a list of transposable elements given in parameter `-t` for *ping-pong* cycle activity. It accepts various formats. The format is determined by the file extension. In general, only text files are accepted. Every line represents the coordinates of a single copy of a transposon. Lines that cannot be parsed are discarded silently. The chromosome names in the annotation file must be identical to those in the file with high-throughput sequencing data.

BED file format (file extension `.bed`)

Refer to http://genome.ucsc.edu/FAQ/FAQformat.html#format1 for information about the *BED* file format. *PingPongPro* only uses the fields `chrom` (1), `chromStart` (2), `chromEnd` (3), `name` (4) and `strand` (6). All other columns are ignored. The fourth field (`name`) serves as the identifier of a transposon.

Comma-separated values (file extension `.csv`)

`.csv`-files are expected to have at least five fields. The fields must be separated by exactly one comma. Fields can optionally be enclosed by double-quotes. The following table lists the fields and their meanings. All other columns are ignored.

Table 1: Fields in `.csv`-formatted transposon annotations

#	Field name	Description
1	`identifier`	A name that uniquely identifies the transposon
2	`strand`	The strand which the transposon is located on (either `+` or `−`)
3	`chromosome`	The chromosome which the transposon is located on
4	`start`	Start coordinate of the transposon (zero-based)
5	`end`	End coordinate of the transposon (half-open, i.e., the coordinate points to the first base pair after the end of the transposon)

Example:

```
FBti0015567,-,chr2L,20740303,20748120
```

General Feature Format (file extension `.gff`)

Refer to http://genome.ucsc.edu/FAQ/FAQformat.html#format3 for information about the *General Feature Format*. *PingPongPro* only uses the fields `seqname` (1), `start` (4), `end` (5), `strand` (7) and `group` (9). All other columns are ignored. Notably, the field `feature` (3) is not evaluated, such that all lines in the file are checked for *ping-pong* cycle activity, regardless of their feature type. The ninth field (`group`) serves as the identifier of a transposon.

Gene Transfer Format (file extension `.gtf`)

Refer to http://genome.ucsc.edu/FAQ/FAQformat.html#format4 for information about the *Gene Transfer Format*. *PingPongPro* parses `.gtf`-files in the exact same way as files of the *General Feature Format*.

Tab-separated values (file extension `.tsv`)

PingPongPro parses `.tsv`-files in the exact same way as `.csv`-files, with two exceptions: Fields must be separated by exactly one tab. And fields cannot optionally be enclosed in double-quotes.

Example:

```
FBti0015567	-	chr2L	20740303	20748120
```

8. Output

8.1. *Ping-pong* signatures

PingPongPro outputs the list of all stacks which overlap by ten nucleotides in a tab-separated file named `ping-pong_signatures.tsv`. The following table lists the meanings of the columns in the file:

Table 2: Output of *ping-pong* signatures in `.tsv`-format

#	Field name	Description
1	`contig`	The chromosome or scaffold which the *ping-pong* signature is located on.
2	`position`	The coordinate of the 5' end of the stack on the forward strand (zero-based).
3	`FDR`	Estimated fraction of signatures that have the same combination of properties, but that are not true *ping-pong* signatures. Lower values indicate higher confidence that the signature is not due to coincidence. For example, if there are 100 putative *ping-pong* signatures and 18 stacks with the same combination of properties, but with a different overlap, then `FDR` has a value of approximately 0.18 (18 divided by 100).
4	`stackHeightOnPlusStrand`	The number of reads that make up the stack on the forward strand (may be fractional, if it contains multi-hits).
5	`stackHeightOnMinusStrand`	The number of reads that make up the stack on the reverse strand (may be fractional, if it contains multi-hits).

8.2. *Ping-pong* cycle activity per transposon

If the parameter `-t` or `-T` is specified, *PingPongPro* produces statistics about *ping-pong* cycle activity within the regions of given (`-t`) or predicted (`-T`) transposons. Statistics about given transposons are written to a tab-separated file named `transposons.tsv`; statistics about predicted transposons are written to a tab-separated file named `predicted_transposons.tsv`. Both files have the same format:

Table 3: Output of *ping-pong* cycle activity per transposon in `.tsv`-format

#	Field name	Description
1	`identifier`	For annotated transposons, the identifier of the transposon given in the annotation file; for predicted transposons, a dynamically generated identifier based on coordinates.

#	Field name	Description
2	`strand`	The strand which the transposon is located on (either + or −). In the file for predicted transposons, this field always has the value +. This is because *PingPongPro* currently does not try to guess the strand.
3	`contig`	The chromosome or scaffold which the transposon is located on.
4	`start`	Start coordinate of the transposon (zero-based).
5	`end`	End coordinate of the transposon (half-open, i.e., the coordinate points to the first base pair after the end of the transposon).
6	`pValue`	P-value indicating the confidence that there is *ping-pong* cycle activity within the region of the transposon.
7	`qValue`	P-value corrected for multiple testing using the false discovery rate procedure by Benjamini and Hochberg, 1995.
8	`pingPongReads`	The number of reads that overlap by ten nucleotides within the region of the transposon (after the reads have been weighted by *PingPongPro*'s scoring system).
9	`normalizedPingPongReads`	RPKM-normalized value of `pingPongReads`, i.e., the value of `pingPongReads` divided by the length of the transposon in kbp and divided by the total number of reads in the input file per million. This figure should be suitable to compare the *ping-pong* cycle activity between transposons and between samples.
10	`discardedPingPongReads`	The number of reads which overlap by ten nucleotides within the region of the transposon but which have been discarded by *PingPongPro*'s scoring system. The sum of `pingPongReads` and `discarded PingPongReads` equals the unweighted number of reads which overlap by ten nucleotides within the region of the transposon.
11	`strandRatio`	The unweighted number of *ping-pong* reads on the forward strand divided by the unweighted number of *ping-pong* reads on the reverse strand.

8.3. Browser tracks

When *PingPongPro* is run with the parameter `-b`, it creates browser track files, which are suitable for display in common genome browsers, such as the *UCSC Genome Browser* (http://genome.ucsc.edu/) or the *Integrative Genomics Viewer* (https://www.broadinstitute.org/igv/home).

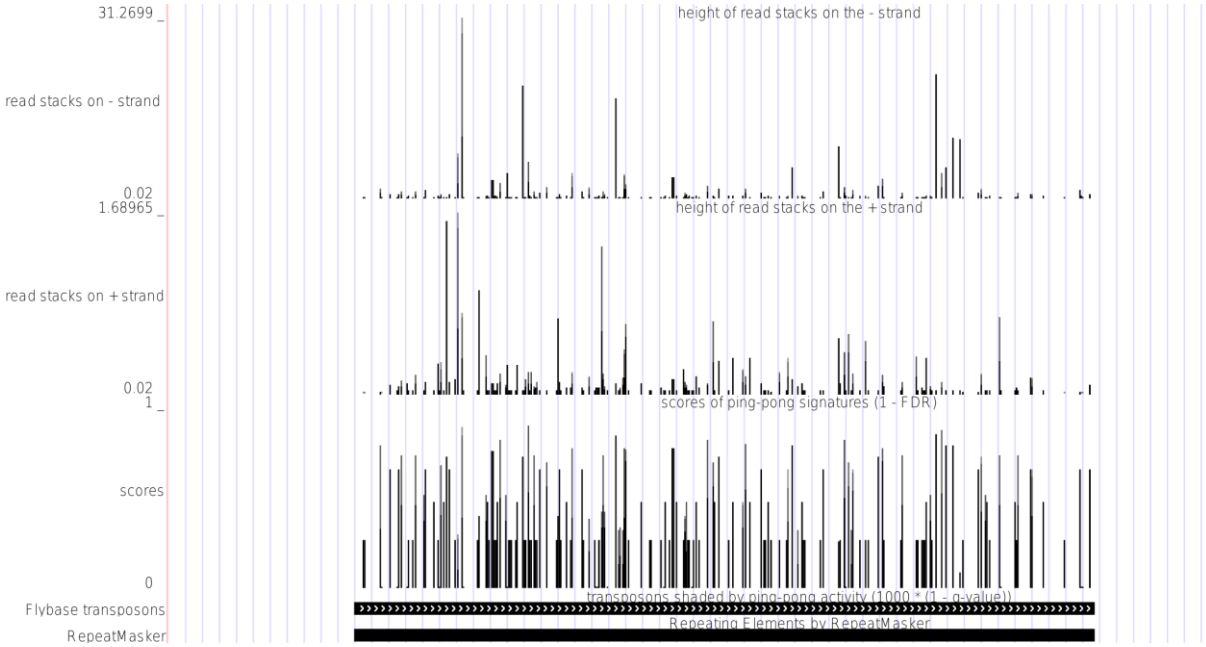

Figure 2: Visualization of *ping-pong* signatures in the *UCSC Genome Browser*.
PingPongPro is able to generate output in formats suitable for visualization in genome browsers like the *UCSC Genome Browser* and *IGV*. This figure is a screenshot of the *UCSC Genome Browser*. It depicts *ping-pong* signatures within the transposable element `FBti0019102` with the genomic coordinate chr2L:1220184-1227592 in *D. melanogaster* (available at the *NCBI Sequence Read Archive* via the *SRA* accession number `SRR010960` [5]).
The topmost row (`read stacks on - strand`) shows the heights of stacks on the reverse strand; the second row (`read stacks on + strand`) shows the heights of stacks on the forward strand. Most of the stacks have fractional heights, because they are mostly made up of multi-mapping reads. The third row (`scores`) shows the scores that *PongPongPro* calculated for the *ping-pong* signatures. The heights of the bars range between 0 and 1 and represent the scores of *ping-pong* signatures. The fourth row (`Flybase transposons`) shows the location of annotated transposons. Transposons are shaded according to their score: Transposons with high scores are darker and transposons with low scores are brighter. The last row (`RepeatMasker`) shows regions with repetitive elements.
The transposon is covered with *ping-pong* signatures from start to end, which is an indication that it is thoroughly suppressed through the *ping-pong* cycle.

Ping-pong signatures

Three `.bedGraph`-files are generated for the detected *ping-pong* signatures. Refer to http://genome.ucsc.edu/FAQ/FAQformat.html#format1.8 for more information about the *bedGraph* file format.

The file `ping-pong_signatures_read_stacks_on_minus_strand.bedGraph` contains the heights of the stacks on the reverse strand of *ping-pong* signatures. The file `ping-pong_signatures_read_stacks_on_plus_strand.bedGraph` contains the heights of the stacks on the forward strand of *ping-pong* signatures. The file `ping-`

`pong_signatures_scores.bedGraph` contains the scores of *ping-pong* signatures. The scores are calculated as 1 minus the value of the `FDR` field in the output file `ping-pong_signatures.tsv` (see section 8.1). So the closer the score is to 1, the more likely it is a true *ping-pong* signature and not a random alignment of reads; the closer the score is to 0, the more unlikely it is a true *ping-pong* signature.

Annotated transposons

A `.bed`-file is generated for detected *ping-pong* cycle activity within the annotated transposons that have been passed to *PingPongPro* via the parameter `-t`. Refer to http://genome.ucsc.edu/FAQ/FAQformat.html#format1 for more information about the *BED* file format.

The file `transposons.bed` contains the transposons given in the annotation file. The `score` column reflects how confident *PingPongPro* is that there is *ping-pong* cycle activity within the transposon. The score is calculated as 1000 minus the value of the `qValue` field in the output file `transposons.tsv` multiplied by 1000. So a score near 1000 indicates high confidence and a score near 0 indicates low confidence.

Predicted transposons

If *PingPongPro* is run with the parameter `-T`, then it can generate browser tracks for predicted regions with *ping-pong* cycle activity. The browser track is named `predicted_transposons.bed` and has the same format as the *BED*-file for annotated transposons. The only difference is that the content is based on the output file `predicted_transposons.tsv`.

8.4. Plots

When *PingPongPro* is run with the parameter `-p`, it generates graphical plots. It uses *R-Project* for rendering of plots. So the program `Rscript` must be available in the `PATH` environment variable (see section 5.3). In general, *PingPongPro* first generates an *R* script with the name of the plot and the file extension `.R`. The script contains the data to be plotted and commands for the generation of the plot. This script is then executed with `Rscript`, which renders the plot as a *PDF* file.

Plots for the scoring of *ping-pong* signatures

As explained in section 4.1, *PingPongPro* scores putative *ping-pong* signatures based on how likely it is that the signature is just a random alignment of reads, which happen to overlap by ten nucleotides. In order to estimate how often a signature with a certain combination of properties occurs by chance, it counts how often such signatures occur with overlaps other than ten nucleotides.

The file `ping-pong_signature_z-scores.pdf` contains one bar plot for every possible combination of properties (height score, independence of the local coverage, adenine bias). The bar plots show how often signatures with a certain combination of properties were counted with an overlap of ten nucleotides and how often with other overlaps. The counts are normalized using z-scores. When the bar for an overlap of ten nucleotides is remarkably higher than other bars, this means that this combination of properties is

peculiar to true *ping-pong* signatures. Putative *ping-ping* signatures with these properties get good scores.

Plots for the scoring of transposable elements

As explained in section 4.2, *PingPongPro* decides if a transposon shows signs of *ping-pong* cycle activity based on the relative abundance of reads with an overlap of ten nucleotides compared to reads with other overlaps. If the former is significantly higher, the transposon is assigned a high score; otherwise, it is assigned a low score.

The file `transposons_z-scores.pdf` contains one bar plot for every transposon that was given in the annotation file via the parameter `-t`. The file `predicted_transposons_z-scores.pdf` contains the same data for predicted transposons (if the parameter `-T` was specified). The bar plots show the abundance of reads with an overlap of ten nucleotides within the region of the transposon compared to the abundance of reads with other overlaps. The read counts have been weighted according to *PingPongPro*'s scoring system and are normalized using z-scores.

9. References

[1] Benjamini Y, Hochberg Y: *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing*. Journal of the Royal Statistical Society 1995, Series B (Methodological), 57(1):289-300.

[2] 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(6):1089-103.

[3] Gunawardane LS, Saito K, Nishida KM, Miyoshi K, Kawamura Y, Nagami T, Siomi H, Siomi MC: *A slicer-mediated mechanism for repeat-associated siRNA 5' end formation in Drosophila*. Science 2007, 315(5818):1587-90.

[4] Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D. *The human genome browser at UCSC*. Genome Research 2002, 12(6):996-1006.

[5] Li C, Vagin CC, Lee S, Xu J, Ma S, Xi H, Seitz H, Horwich MD, Syrzycka M, Honda BM, Kittler ELW, Zapp ML, Klattenhoff C, Schulz N, Theurkauf WE, Weng Z, Zamore PD: *Without Argonaute3, Aubergine-bound piRNAs collapse but Piwi-bound piRNAs persist*. Cell 2009. 137(3):509–521.

[6] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; 1000 Genome Project Data Processing Subgroup: *The Sequence Alignment/Map format and SAMtools*. Bioinformatics 2009, 25(16):2078-9.

[7] Preall JB, Czech B, Guzzardo PM, Muerdter F, Hannon GJ: *shutdown is a component of the Drosophila piRNA biogenesis machinery*. RNA 2012, 18(8):1446-57.

[8] R Development Core Team (2008): *R: A language and environment for statistical computing. R Foundation for Statistical Computing*, Vienna, Austria. ISBN 3-900051-07-0, URL: http://www.R-project.org [accessed: April 9, 2014]

[9] Thorvaldsdóttir H, Robinson JT, Mesirov JP: *Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration*. Briefings in Bioinformatics 2012.

[10] Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, Dicuccio M, Edgar R, Federhen S, Feolo M, Geer LY, Helmberg W, Kapustin Y, Khovayko O, Landsman D, Lipman DJ, Madden TL, Maglott DR, Miller V, Ostell J, Pruitt KD, Schuler GD, Shumway M, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Tatusov RL, Tatusova TA, Wagner L, Yaschenko E: *Database resources of the National Center for Biotechnology Information*. Nucleic Acids Research 2008, 36 (Database issue): D13-21.