Chapter 12

Computing siRNA and piRNA Overlap Signatures

Christophe Antoniewski

Abstract

High-throughput sequencing approaches opened the possibility to precisely map full populations of small RNAs to the genomic loci from which they originate. A bioinformatic approach revealed a strong tendency of sense and antisense piRNAs to overlap with each other over ten nucleotides and had a major role in understanding the mechanisms of piRNA biogenesis. Using similar approaches, it is possible to detect a tendency of sense and antisense siRNAs to overlap over 19 nucleotides. Thus, the so-called overlap signature which describes the tendency of small RNA to map in a specific way relative to each other has become the approach of choice to identify and characterize specific classes of small RNAs. Although simple in essence, the bioinformatic methods used for this approach are not easily accessible to biologists. Here we provide a python software that can be run on most of desktop or laptop computers to compute small RNA signatures from files of sequencing read alignments. Moreover, we describe and illustrate step by step two different algorithms at the core of the software and which were previously used in a number of works.

Key words High-throughput sequencing, Small RNA signature, siRNA, piRNA

1 Introduction

PIWI-interacting RNAs (piRNAs) are a major class of gonad-specific, ~24–29 nt long small RNAs involved in transposons and retrotransposons silencing, both in vertebrates and invertebrates. Their biogenesis has long remained enigmatic, as, in contrast to siRNA biogenesis, it does not involve the activity of Dicer enzymes. In 2007, two seminal works revealed that the production of piRNAs in *Drosophila* germ cells requires an amplification mechanism called the "Ping-Pong cycle" [1, 2]. In the Ping-Pong cycle, an antisense piRNA guides an Argonaute protein of the PIWI subfamily (Aubergine or Piwi) to the cleavage of a complementary transposon sense transcript. One of the two cleavage products gives rise to a sense piRNA–Agonaute-3 complex which can in turn guide the cleavage of a complementary transposon antisense transcript, generating a new antisense piRNA. Because all three

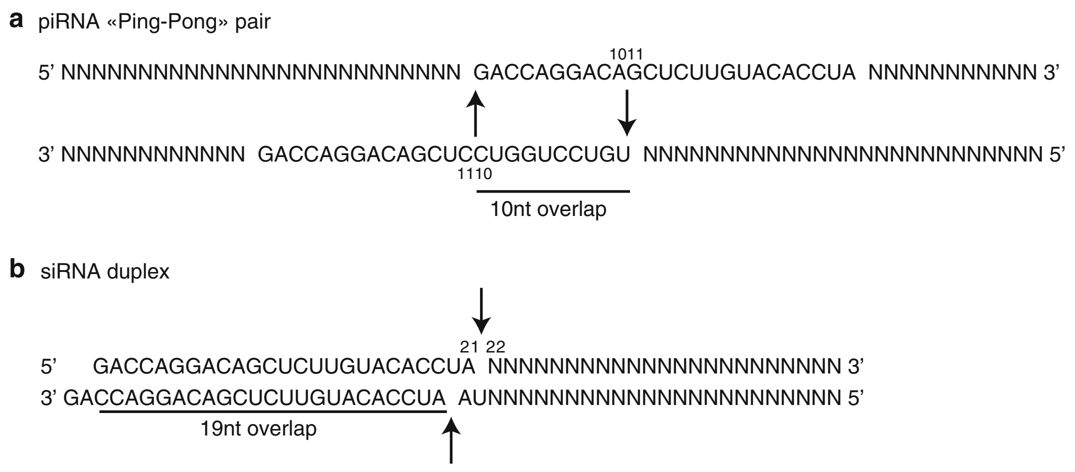

a piRNA «Ping-Pong» pair
1011
5' NNNNNNNNNNNNNNNNNNNNNNNNNN GACCAGGACAGCUCUUGUACACCUA NNNNNNNNNN 3'
3' NNNNNNNNNNNN GACCAGGACAGCUCCUGGUCCUGU NNNNNNNNNNNNNNNNNNNNNNNNN 5'
1110
10nt overlap
b siRNA duplex
21 22
5' GACCAGGACAGCUCUUGUACACCUA NNNNNNNNNNNNNNNNNNNNNNN 3'
3' GACCAGGACAGCUCUUGUACACCUA AUNNNNNNNNNNNNNNNNNNNNNNN 5'
19nt overlap

Fig. 1 piRNA "Ping-Pong" pairs and siRNA duplexes. (**a**) A typical pair of sense and antisense piRNAs overlapping by 10 nt. The 5′ end of piRNAs is determined by Argonaute-mediated cleavage (*arrows*) of a long piRNA precursor transcript (Ns) that occurs between positions 10 and 11 relative the 5′ end of the complementary guide piRNA. (**b**) An siRNA duplex with typical 3′ 2 nt overhang and 19 nt overlap between sense and antisense siRNAs is produced by cleavage of long double-stranded RNAs by the type III RNAse Dicer (*arrows*)

Drosophila PIWI Argonaute proteins cleave their target RNAs between nucleotides 10 and 11 relative to the 5′ end of their piRNA guides, iterations of the "Ping-Pong" mechanism lead to the accumulation of pairs of sense and antisense piRNAs which overlap exactly by ten nucleotides (Fig. 1a).

Figuratively speaking, high-throughput sequencing (HTS) technologies had an explosive impact on the field of piRNA biology. HTS of small RNAs allowed uncovering the specific features (sizes and nucleotide compositions) of piRNAs bound to Piwi, Aubergine, or Agonaute-3. Moreover, alignment software adapted to HTS datasets gave access to piRNA genome mapping at nucleotide level and allowed measuring, through computing approaches, their striking tendency to form 10 nt-overlapping pairs [2–8]. Interestingly, these simple but elegant statistical approaches to search for the so-called piRNA signature can also apply to siRNAs. Accordingly, if the Ping-Pong mechanism for piRNA production leaves a 10 nt-overlap signature in sequencing datasets, generation of siRNA duplexes by Dicer enzymes is expected to leave a 19 nt-overlap signature (Fig. 1b).

In this chapter, we describe two similar but distinct algorithms first used for the detection of the piRNA Ping-Pong signature [2, 4]. The algorithms were also able to detect 19 nt-overlap "Dicer" signatures in HTS data from siRNAs co-immunoprecipitated with the p19 viral suppressor of RNAi (*see* ref. [9], and Chapter 13 in this book) and, more recently, in sequencing data of viral siRNAs generated during *Drosophila* Nora virus infections [10]. We provide a python language script that implements these algorithms and can easily be used by biologists without advanced skills in bioinformatics to search for small RNA signatures with the help of any desktop computer equipped with a python interpreter.

2 Materials

A computer with at least 4 Gb of RAM.

Basic knowledge in UNIX management. Being able to open a UNIX terminal, to navigate in the file system and to launch simple command lines is sufficient.

Python interpreter installed. Python is pre-installed in Linux and Mac OS X standard configurations and can be easily installed on Window computers (http://www.python.org/getit/windows/).

Collections and *numpy* python modules installed. If this is not the case (unlikely), see how to install the modules at http://docs.python.org/.

An alignment file reporting small RNA sequence alignment to a reference genome. The script takes as input standard SAM files (http://samtools.sourceforge.net/SAM1.pdf). It can also take a more compact tabular format generated by bowtie (*see* ref. 11 and **Note 1**) with one reported match by line and (1) the read identifier in the first column, (2) the matched reference strand (+ or −) in the second column, (3) the name of the matched region in the reference genome (which corresponds to the sequence header when the reference genome is in a fasta format), (4) the 5′ coordinate of the match (0-based leftmost mapping position) always given relative to the sense strand (+) of the genome reference, and (5) the sequence of the matched read. It is important that the alignment file only reports matches to genomic targets relevant to the investigation. For example, searching for signatures of Nora virus derived siRNAs should not be performed using an input file that reports alignments to both *Drosophila* and Nora virus genomes (*see* **Note 2**).

2.1 Installation

1. Create a "test" directory.
2. Download the *signature.py* script file at http://drosophile.org/GEDlab/?page_id=730, or copy/paste the script in a text file named *signature.py* (*see* Appendix 1). In python language, code indentation is critical. If you copy/paste the signature.py code in a new file, make sure that the numbers of space characters at the beginning of lines have not been modified. Put a copy of *signature.py* in your test directory.

3. Make the *signature.py* script executable. This step is optional (*see* **Note 3**). Under UNIX compliant OS, it can be performed by navigating to your test directory (*cd ~/test*), typing *chmod 755 signature.py*, and pressing the enter key.

2.2 Running Signature.py

The script is run using the command line:

signature.py <input_file> <min_size> <max_size> <min_overlap> <max_overlap> <output_file>

where <*input_file*> is the path to the sequence alignment file (in a SAM or a bowtie format), <*min_size*> is the minimal size (integer) of small RNA reads to be analyzed, <*max_size*> is the maximal size (integer) of small RNA reads to be analyzed, <*min_overlap*> is the minimum overlap searched for small RNA pairs (an integer that expresses a 1-based overlap; a value of 1 means that the 5′ ends of small RNA pairs overlap with one nucleotide), <*max_overlap*> is the maximum overlap searched for small RNA pairs (an integer that expresses a 1-base overlap; a value of 21 will search for small RNA overlapping up to 21 nt, *see* **Note 3**), and <output_file> is the path to the output text file generated by the script.

Example: *signature.py bowtie_viral_matches.sam 21 21 1 21 myresults.txt* analyzes the overlapping tendencies of 21 nt reads matched in the *bowtie_viral_matches.sam* and outputs a result data frame in the *myresults.txt* file.

3 Methods

Here we describe in greater details the operations performed by the signature.py script. This section will help researchers with python expertise to adapt or extend the script to accommodate specific needs. However, we tried to simplify it as far as possible to help biologists to understand the two algorithms used for finding small RNA overlap signatures so that they are able to evaluate the biological significance of their analyses. Note that the terms "instance," "class," and "method" used below are typical python terms related to Object Oriented Programming.

3.1 Data Treatment by Signature.py Python Script

The signature.py script reads tabular bowtie or SAM alignment reports and implements as many instances of the *smRNAwindow* python class (*see* **Note 4**) as necessary. It is noteworthy that an *smRNAwindow* instance is generated for each distinct item found in the alignment file (by item, we mean genes, chromosomes, viral genomes or subgenomes, etc.). For example, if the alignment file contains only matches to a single molecule RNA virus genome, python will generate a single *smRNAwindow* instance. In contrast, if the alignment file contains matches to an RNA virus genome with two subgenomic A and B fragments, python will generate two *smRNAwindow* instances. Along the same line, if the alignment file reports matches to *Drosophila* genes, python will generate ~18,000 *smRNAwindow* instances.

3.2 The smRNAwindow Class

In addition to the standard *def __init__* python method which instantiates a python dictionary, the *smRNAwindow* class has four methods (*see* **Note 5**).

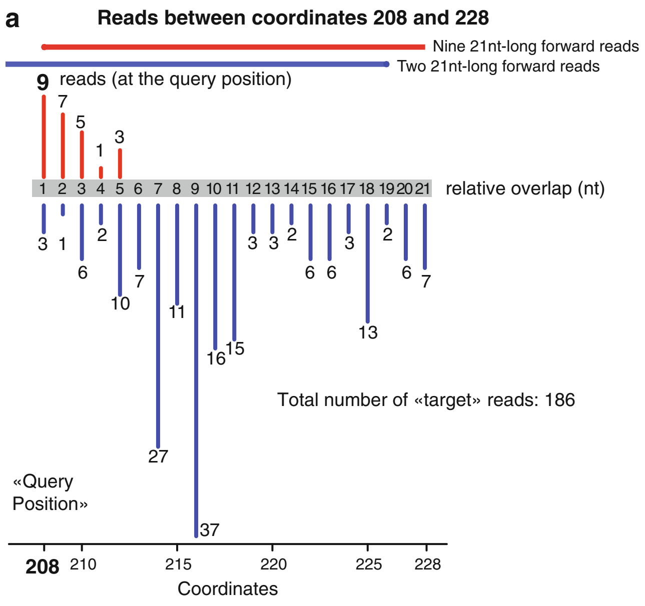
The *addread* method adds reads matched to an item to the corresponding *smRNAwindow* instance, which is indexed with this item in the *objDic* dictionary of *smRNAwindow* instances. Note that *smRNAwindow* instances are created when necessary on the fly during the initial data acquisition process.

The *readcount* method computes the number of reads of minimum and maximum sizes (given as method parameters) attributed to an *smRNAwindow* instance.

The *count_pairs* method computes the number of small RNA pairs found in an *smRNAwindow* instance, with an overlap of *minscope* to *maxscope* nucleotides (the “scope” of the signature). When *count_pairs* is called, each position of small RNAs with the appropriate size (defined by the minsize and maxsize parameters) is iteratively tested to find whether reads are referenced to an overlapping position of the opposite strand, within the *scope* range. When this happens, the lower number of reads between the query position and the “paired” position is added to a local histogram of number of pairs by class of overlap values. For instance (Fig 2a), if 9 reads at the query position *p* on the forward strand are found to overlap 2 reads at the position $p+19$ (19 nt overlap) on the reverse strand, the class 10 of the local histogram is incremented by 2 (only two pairs can be formed in this example). Along the iteration process, local histograms for each query position are summed. As all positions of both forward and reverse strands are examined, pairs of small RNA reads are counted twice. Therefore, the values in the summed histogram are divided by 2, before being returned by the *count_pairs* method, as a histogram of number of pairs by class of overlap values (Fig. 2b).

The *overlap_probability* method computes the probability, within a *smRNAwindow* instance, to find an antisense read overlapping with a sense read over *minscope* to *maxscope* nucleotides (the “scope” of the signature). The *overlap_probability* method of the *smRNAwindow* python class is inspired from the procedure first described by Brennecke et al. [2] and further formalized in ([3], *see* supporting online material of this reference).

a



overlap	number of pairs	number of target reads	probability	weighted probability (9/103373)
1	3	3	0.016129	1.4042E-06
2	1	1	0.005376	4.6805E-07
3	6	6	0.032258	2.8085E-06
4	2	2	0.010753	9.3619E-07
5	9	10	0.053763	4.6808E-06
6	7	7	0.037634	3.2765E-06
7	9	27	0.145161	1.2638E-05
8	9	11	0.05914	5.1489E-06
9	9	37	0.198925	1.7319E-05
10	9	16	0.086022	7.4894E-06
11	9	15	0.080645	7.0212E-06
12	3	3	0.016129	1.4042E-06
13	3	3	0.016129	1.4042E-06
14	2	2	0.010753	9.3619E-07
15	6	6	0.032258	2.8085E-06
16	6	6	0.032258	2.8085E-06
17	3	3	0.016129	1.4042E-06
18	9	13	0.069892	6.0850E-06
19	2	2	0.010753	9.3619E-07
20	6	6	0.032258	2.8085E-06
21	7	7	0.037634	3.2765E-06

b

overlap	pairs	z-score	overlap probability
1	14229	-0.2856	0.0529
2	14055	-0.6086	0.0486
3	14074	-0.5733	0.0485
4	14344	-0.0722	0.0480
5	14156	-0.4211	0.0449
6	13945	-0.8127	0.0433
7	14063	-0.5937	0.0441
8	14124	-0.4805	0.0448
9	14095	-0.5343	0.0449
10	14030	-0.6550	0.0428
11	14020	-0.6735	0.0432
12	14049	-0.6197	0.0424
13	13850	-0.9890	0.0426
14	14127	-0.4749	0.0438
15	14265	-0.2188	0.0452
16	14589	0.3825	0.0478
17	14696	0.5811	0.0493
18	15099	1.3290	0.0518
19	16096	3.1794	0.0588
20	15309	1.7188	0.0580
21	14826	0.8224	0.0542

c

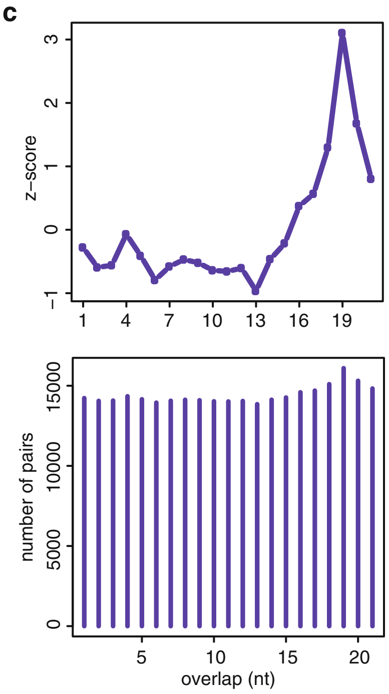


Fig. 2 Local and global histograms of number of pairs and of overlap probabilities. (**a**) An example of local histograms computed by signature.py at the query position 208 in the Nora virus genome (reference Nora). *Red* and *blue vertical bars* represent the number of reads found at the corresponding coordinates (5′ end of sequence reads) in sense and antisense orientation, respectively. An example of coverage by nine sense reads (*red*) at position 208 and two antisense reads (*blue*) at position 226 is given at the top; 5′ ends of the sequences are indicated with *bulges*. All 186 target reads (*blue*) are used to compute the local overlap probabilities.

When *overlap_probability* is called, each position (the *query* position) of small RNAs with the appropriate size (defined by the minsize and maxsize parameters) is iteratively tested to find reads that are referenced to an overlapping position of the opposite strand, within the *scope* range. For each *query* position (*see* Fig. 2a), a local histogram of probabilities by class of overlap values is computed by normalizing the number of read (target reads) at each overlap position to 100 % over the total number of target reads up to a distance of *scope* nucleotides. In a second step, the values of local probability histograms are weighted by normalizing the number of reads at the *query* position to 100 % over the total number of reads examined in queries (Fig 2a, column "weighted probability") and then summed in a global probability histogram for the *smRNAwindow* instance (Fig. 2b, column "overlap probability"), which is eventually returned by the *overlap_probability* method. This approach ensures that each local probability histogram contributes to the final probability histogram relative to the sequencing frequency of the small RNAs for which the local probability histogram was computed.

3.3 Signature.py Output

The script returns a data frame in a tabulated text output file. In this data frame, the "overlap" column refers to the overlap values. The "pairs" column refers to the numbers of pairs found during the analysis for *all* items referenced in the input sequence dataset. In fact, the histograms returned by the *count_pairs* method for each *smRNAwindow* instance are summed. The "*z*-score" column is a standardization of the "pairs" column. It is computed by the z_score function of the script using the formula $Z_i = (P_i - \text{mean}(P))/\text{standard deviation}(P)$, where P stands for the number of pairs found for an overlap of i nucleotides. Finally, the "overlap_prob" column refers to the overall probability to find small RNA pairs overlapping with each other over the indicated overlap value. Again, note that probabilities found for each *smRNAwindow*

Fig. 2 (continued) The data frame with the local histograms for the number of pairs, the number of target reads, the overlap probabilities, and the weighted overlap probabilities are shown at the right-hand side. As 103,373 21 nt reads were matched to the Nora Virus genome in the example, the weighting factor for the query position 208 is 9/103,373. (**b**) Summed histograms of small RNA pairs, corresponding *z*-scores, and overlap probabilities for 21 nt small RNAs matched to the Nora Virus genome. (**c**) Graphic representation of the numbers of 21 nt small RNA pairs (*bottom*) and associated *z*-scores (*top*) for the indicated nucleotide overlaps, as computed in (**b**) instances are merged in the final output, taking into account the relative weight of these instances.

Two cases of 21 nt siRNA signature analyses are given in Figs. 2 and 3. In Fig. 2, we detailed an analysis performed on Nora virus siRNAs from *Drosophila* adult thoraxes matched to a reconstituted Nora virus genome [10]. In Fig. 3, we detailed an analysis performed of endo-siRNAs immunoprecipitated with the viral protein P19 and matched to Drosophila transposon sequences ([10], and Chapter 13). In addition, we computed by hand the *z*-scores of the overlap probabilities using the formula $Zi = (p_i - \text{mean}\ (p))/\text{standard deviation}\ (p)$, where p stands for the probability of finding an overlap of i nucleotides. Note that these *z*-scores are similar but distinct from the *z*-scores found for the numbers of small RNA pairs. Indeed, the two algorithms described in this chapter can be viewed as two convergent mathematical functions whose behaviors differ at low read density and/or when the genomic distribution of the read matches varies (manuscript in preparation).

4 Notes

1. The command line to generate such a format is *bowtie -v 1 -M 1--est--trata--uppress 6,7,8 path/to/bowtie/index -f path/to/the/fasta/read/dataset > path/to/the/alignment/report/file*.
2. The signature.py code is built using *smRNAwindows* class instances. As the final output provides a weighted average of the values computed for each instance, values returned by instances corresponding to irrelevant items would minor values returned for relevant items. However the Class instantiation procedure used in the script allows to easily refactor python code for returning signature values for each *smRNAwindow* class instance.
3. If not made executable, the signature.py script may still be run using the syntax *python signature.py parameter-1 parameter-2 …*
4. For non-programmers, a python class may be viewed as the prototype or the template of records in a database, and an instance of this class as an actual record.
5. In addition to data (“attributes” in Object Oriented Programming), classes may contain internal functions that can transform data in the class instances or compute values from these data. These functions that are “internal” to the class instances (“private” in Objet Oriented Programming terminology) are called “methods.”

a

overlap	pairs	z-score	overlap probability
1	5584	-0.322885	0.023251
2	5769	-0.248787	0.023924
3	4157	-0.894443	0.015374
4	5864	-0.210736	0.054027
5	5720	-0.268413	0.026334
6	7299	0.364025	0.030530
7	5767	-0.249588	0.023819
8	4647	-0.698183	0.016514
9	6342	-0.019283	0.032864
10	6876	0.194601	0.029577
11	4965	-0.570814	0.019876
12	6265	-0.050124	0.025330
13	5979	-0.164675	0.025141
14	5988	-0.161071	0.033796
15	6515	0.050009	0.021287
16	6578	0.075243	0.030601
17	5873	-0.207132	0.018759
18	5317	-0.429827	0.017200
19	16959	4.233154	0.122629
20	4530	-0.745045	0.017574
21	7199	0.323972	0.040412

b

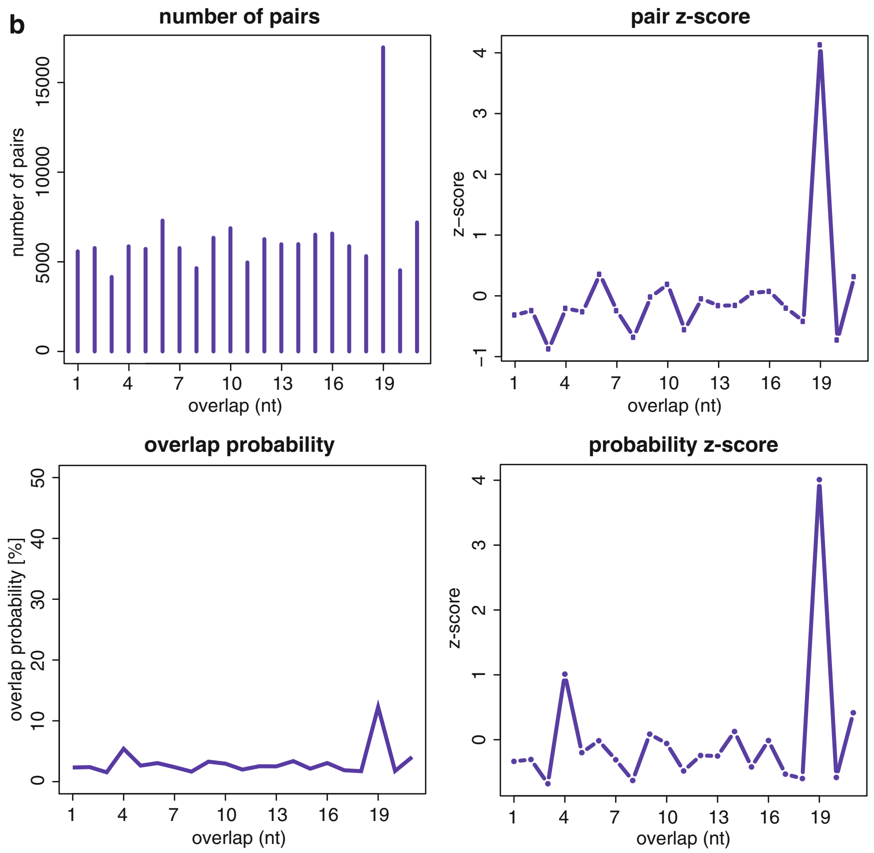


Fig. 3 Number of pairs, *z*-scores, and overlap probabilities. (**a**) An example of a data frame computed by signature.py from p19-bound siRNAs matched to Drosophila transposon sequences [9]. (**b**) Graphic representation of the number of pairs, the *z*-scores of the number of pairs (*top*), the probabilities of small RNA overlaps, and the *z*-scores of these probabilities (*bottom*) using R plot functions

Appendix: Signature.py Code

```
#!/usr/bin/env python
# computes overlap signatures from a bowtie or SAM input
# version Met Mol Biol 06-2-2012
# Usage signature.py <bowtie or sam input> <minsize> <maxsize> <minscope> <maxscope> <output>

import sys
from collections import defaultdict
from numpy import mean, std

class SmRNAwindow:
  def __init__(self, gene):
    self.readDict = defaultdict(list) # "dictionary of lists" structure {+/-offset:[size1, size2, ...], ...}
  def addread (self, polarity, offset, size):
    if polarity == "+":
      self.readDict[offset].append(size)
    else:
      self.readDict[-(offset + size -1)].append(size)
    return
  def readcount (self, lim_inf=0, lim_sup=1000):
    n=0
    for offset in self.readDict:
      licenced = [i for i in self.readDict[offset] if (i>=lim_inf and i<= lim_sup)]
      n+=len(licenced)
    return n
  def count_pairs (self, minsize, maxsize, scope):
    size_range = range (minsize, maxsize+1)
    Query_table = {}
    Target_table = {}
    frequency_table = dict ([(i, 0) for i in scope])
    for offset in self.readDict:
      for size in self.readDict[offset]:
        if size in size_range:
          Query_table[offset] = Query_table.get(offset, 0) + 1
          Target_table[offset] = Target_table.get(offset, 0) + 1
    for offset in Query_table:
      for i in scope:
        frequency_table[i] += min(Query_table[offset], Target_table.get(-offset -i +1, 0))
    for i in frequency_table:
      frequency_table[i] = frequency_table[i] / 2 # the number of smRNA PAIRS, not of the paired smRNA.
    return frequency_table
  def overlap_probability (self, minsize, maxsize, scope):
    size_range = range (minsize, maxsize+1)
    Query_table = {}
    Target_table = {}
    Total_Query_Numb = 0
    general_frequency_table = dict ([(i,0) for i in scope])
    for offset in self.readDict:
      for size in self.readDict[offset]:
        if size in size_range:
          Query_table[offset] = Query_table.get(offset, 0) + 1
          Target_table[offset] = Target_table.get(offset, 0) + 1
          Total_Query_Numb += 1
    for offset in Query_table:
      frequency_table = dict ([(i,0) for i in scope])
      number_of_targets = 0
      for i in scope:
        frequency_table[i] += Query_table[offset] *  Target_table.get(-offset -i +1, 0)
        number_of_targets += Target_table.get(-offset -i +1, 0)
      for i in scope:
        try:
          general_frequency_table[i] += (1. / number_of_targets / Total_Query_Numb) * frequency_table[i]
        except ZeroDivisionError :
          continue
    return general_frequency_table

def load_input (input_file):
  F=open(input_file)
  sampleline = F.readline()
  F.close()
  samplefields=sampleline.split()
  if len(samplefields) < 2 : #alignment format are tabulated with at least two fields.
    print "error: invalid input format"
    sys.exit()
  if samplefields[1] in ["+", "-"]: # standard tabular bowtie format detected
    F = open (input_file, "r")
    for line in F:
      fields = line.split()
      polarity = fields[1]
      gene = fields[2]
      offset = int(fields[3]) + 1 # to shift on 1-based coordinates
      size = len (fields[4])
      try:
        objDic[gene].addread (polarity, offset, size)
      except KeyError:
        objDic[gene] = SmRNAwindow(gene)
        objDic[gene].addread (polarity, offset, size)
    F.close()
  elif samplefields[0][0] == "@": # SAM format detected
    F = open (input_file, "r")
    for line in F:
      if line[0] == "@" : continue
      fields = line.split()
      if fields[2] == "*" : continue
      if fields[1] == "0": polarity = "+"
      else: polarity = "-"
      gene = fields[2]
      offset = int(fields[3])
      size = len (fields[9])
      try:
        objDic[gene].addread (polarity, offset, size)
      except KeyError:
        objDic[gene] = SmRNAwindow(gene)
        objDic[gene].addread (polarity, offset, size)
    F.close()
  else:
    print "error: invalid input format"
    sys.exit()

def z_score (table):
  value_list = [table[i] for i in sorted (table)]
  if std(value_list):
    meanlist = mean(value_list)
    stdlist = std(value_list)
    return dict ( zip ( sorted(table), [(i-meanlist)/stdlist for i in value_list] ) )
  else:
    return dict ( zip ( sorted(table), [0 for i in table]) )

objDic = {} # objDic is defined as a global variable

def __main__():
  if len(sys.argv) < 7:
    print "error: not enough parameters provided"
    sys.exit()
  load_input (sys.argv[1]) # feeds the global variable objDic (a dictionary of SmRNAwindow instances, keys=genes)
  minsize = int(sys.argv[2])
  maxsize = int(sys.argv[3])
  minscope = int(sys.argv[4])
  maxscope = int(sys.argv[5]) + 1
  general_pairs_table = dict ([(i,0) for i in range(minscope,maxscope)])
  general_prob_table = dict ([(i,0) for i in range(minscope,maxscope)])
  readcount_dic = {} # for normalized summing of local_percent_table(s)
  Total_read_in_objDic = 0
  for item in objDic:
    readcount_dic[item] = objDic[item].readcount(minsize, maxsize)
    Total_read_in_objDic += readcount_dic[item]
  OUT = open (sys.argv[-1], "w")
  for x in (objDic):
    local_pairs_table = objDic[x].count_pairs ( minsize, maxsize, range(minscope,maxscope) )
    local_prob_table = objDic[x].overlap_probability ( minsize, maxsize, range(minscope,maxscope) )
    try:
      for overlap in local_pairs_table.keys():
        general_pairs_table[overlap] = general_pairs_table.get(overlap, 0) + local_pairs_table[overlap]
    except:
      pass
    try:
      for overlap in local_prob_table.keys():
        general_prob_table[overlap]          =            general_prob_table.get(overlap,            0)           +
(1./Total_read_in_objDic*readcount_dic[x]*local_prob_table[overlap])
    except:
      pass
  z_table = z_score(general_pairs_table)
  print >> OUT, "overlap\tpairs\tz-score\toverlap_prob"
  for overlap in sorted(general_prob_table):
    print >> OUT, "%i\t%i\t%f\t%f" % (overlap, general_pairs_table[overlap], z_table[overlap], general_prob_table[overlap])
  OUT.close()

if __name__ == "__main__" : __main__()
```

References

1. Gunawardane LS, Saito K, Nishida KM, Miyoshi K, Kawamura Y, Nagami T, Siomi H, Siomi MC (2007) A slicer-mediated mechanism for repeat-associated siRNA 5' end formation in Drosophila. Science 315:1587–1590
2. Brennecke J, Aravin AA, Stark A, Dus M, Kellis M, Sachidanandam R, Hannon GJ (2007) Discrete small RNA-generating loci as master regulators of transposon activity in Drosophila. Cell 128:1089–1103
3. Brennecke J, Malone CD, Aravin AA, Sachidanandam R, Stark A, Hannon GJ (2008) An epigenetic role for maternally inherited piRNAs in transposon silencing. Science 322: 1387–1392
4. Klattenhoff C, Xi H, Li C, Lee S, Xu J, Khurana JS, Zhang F, Schultz N, Koppetsch BS, Nowosielska A et al (2009) The Drosophila HP1 homolog Rhino is required for transposon silencing and piRNA production by dual-strand clusters. Cell 138: 1137–1149
5. Li C, Vagin VV, Lee S, Xu J, Ma S, Xi H, Seitz H, Horwich MD, Syrzycka M, Honda BM et al (2009) Collapse of germline piRNAs in the absence of Argonaute3 reveals somatic piRNAs in flies. Cell 137:509–521
6. Malone CD, Brennecke J, Dus M, Stark A, McCombie WR, Sachidanandam R, Hannon GJ (2009) Specialized piRNA pathways act in

germline and somatic tissues of the Drosophila ovary. Cell 137:522–535

7. Lau NC, Robine N, Martin R, Chung WJ, Niki Y, Berezikov E, Lai EC (2009) Abundant primary piRNAs, endo-siRNAs, and microRNAs in a Drosophila ovary cell line. Genome Res 19:1776–1785
8. de Vanssay A, Bouge AL, Boivin A, Hermant C, Teysset L, Delmarre V, Antoniewski C, Ronsseray S (2012) Paramutation in Drosophila linked to emergence of a piRNA-producing locus. Nature 490:112–115
9. Fagegaltier D, Bouge AL, Berry B, Poisot E, Sismeiro O, Coppee JY, Theodore L, Voinnet O, Antoniewski C (2009) The endogenous siRNA pathway is involved in heterochromatin formation in Drosophila. Proc Natl Acad Sci U S A 106:21258–21263
10. van Mierlo JT, Bronkhorst AW, Overheul GJ, Sadanandan SA, Ekstrom JO, Heestermans M, Hultmark D, Antoniewski C, van Rij RP (2012) Convergent evolution of argonaute-2 slicer antagonism in two distinct insect RNA viruses. PLoS Pathog 8:e1002872
11. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25