Prost! output description (March 2019)

Input parameters:

1. Before importing data into *Prost!*, the sequencing reads need to be pre-processed by trimming the adapters and filtering for quality reads. Because several tools already exist for these steps, this function was not implemented into *Prost!*.

We usually use fastx_clipper and fastx_trimmer to perform the adapter trimming. And because we are personally interested in high quality reads to be able to study post-transcriptional modifications including nucleotide edition, we almost exclusively pre-process our Illumina data with 'Q30-27' as the minimum Q-score before running Prost!.

2. Then, *Prost!* starts by filtering reads based on their length. The size range of reads to retain is user-defined by a providing a minimum acceptable length and maximum acceptable length.

In order to focus specifically on miRNAs, we usually filter for lengths superior or equal to 17 nucleotides and smaller or equal to 25 nucleotides.

3. *Prost!* then collapses identical sequencing reads into sets of unique sequences and counts how many reads are combined into a single sequence. Depending on the "minCount" the user choses, *Prost!* filters out the sequences that have too few counts.

Note that the 'minCount' parameter sums counts across all samples. For example, a sequence with 20 reads in each sample of a set of 3 samples will have a total count of 60 and would be kept if the 'minCount' is set to 50, but would be discarded if the 'minCount' is set to 100.

For a de novo miRNA annotation of a species, we tend to use a low value such as 5 or 10 to capture as many mature miRNAs as possible, even the potentially lowly expressed secondary strands. For a differential expression analysis, we tend to use 30 or 50 to only focus on reads that show significant level of expression. However, these numbers are highly dependent on the depth of sequencing.

4. Finally, *Prost!* aligns reads on a genomic reference (e.g. genome assembly, genomic data) and can suggest repeats which would have many potential origins in the genomic reference. The maximum number of location a read can have on the genomic assembly is let to the user to define.

In our case, we usually allow a maximum of 10 locations before a read is classified as likely originating from a repeated sequence in the genome.

The *Prost!* Output file:

To understand the *Prost!* output file, it's important to understand how *Prost!* works. As a difference from most other miRNA software, *Prost!* was designed to have all information and intermediate steps accessible to the user so that one can go back to each step and analyze in detail each result.

The main *Prost!* result file is an .xlsx file made of 7 tabs: 'isomiRs', 'by_genomic_location', 'by_annotation', 'by_seed', 'candidate_mirror-mirs', 'candidate_arm_switching', and 'no_genomic_hits'.

The coming sections go through each tab: its different parts, how it was generated, and the logic behind them. But to understand how each tab relates to the others and therefore how to extract the complete information, we will first describe the way sequences are compared to each other and how they are step-wise combined with each other to obtain manageable data.

The genomic location "binning" process.

Prost! groups, or bins, isomiR sequences in various ways: by their genomic location(s), by their seed sequence, and by their annotation.

The "by_genomic_location" bins are the most complex and numerous and are also the ones in which most of the information can be found to dig into the diversity of a specific miRNA. Bins in that tab are called 'genomic location bins' and are given an index "glXXX".

Genomic location bins are initiated by taking the set of sequences that align perfectly (i.e. no mismatches) to the genomic reference across the full length of the sequence, and sorting those sequences so that the sequences with the highest normalized count across all samples come first. From there, the isomiR with the highest normalized count, perfectly matching the genome sequence, starts the first bin and is given a designation of "1" (and is also called a "BinStarter" sequence). Then the sequence with the next highest count perfectly matching the genome is considered. If this sequence is "within nucleotide wiggle" (see definition below) of the BinStarter sequence (i.e. have almost the same genomic location(s)), then it is added to that bin and designated as a "2" (Perfect match to the genome but not the most abundant isomiR for that genomic location bin). If that sequence was not "within nucleotide wiggle" of the first sequence, it then starts its own bin and is designated as a type "1" sequence. This continues until we run out of sequences that perfectly match the genome.

For sequences to be "within nucleotide wiggle", they must be within (by default) 5 nt of the start AND stop of the sequence that started a bin, be on the same strand of DNA, and both of these previous requirements must hold true for ALL genomic location hits if either sequence aligns to more than one location in the genome (that is the case of miRNA that belong to large gene families and can potentially be generated by more than one loci – e.g. let-7, mir199, etc). Therefore, for example, if a sequence that started a bin had two genomic locations, then a sequence with only one genomic location can never join its bin. That way the user can know that while some isomiRs for a given miRNA can be coming from two loci (maybe two paralogous miRNA genes), some isomiRs were coming from only one of the two loci and therefore this information is conserved and can be tracked back. The number of nucleotides used to determine "within nucleotide wiggle" can be configured by *Prost!* parameter "within_wiggle".

Next, sequences that don't perfectly match the genome are considered for bin joining. If a non-perfect alignment sequence aligns to the genome within nucleotide wiggle of an existing bin, it is added to that bin; if it instead aligns where no bin exists, then that sequence can also start a bin. Sequences that have a next best hit to the genome are designated as "3" or "3.3p". A “3” or “3.3p” sequence is a sequence that has mismatches to the genome, but only on its 3'-end. These non-templated additions are biologically common. We usually tend to allow a maximum of 3 mismatches on the 3'-end before the alignment would be considered bad and discarded (NB: discarded alignments are listed in the ‘no_genomic_hits’ tab) (the number of 3' mismatches allowed is configurable by setting ‘max_3p_mismatches’ in prost.config). All 3's are considered before analyzing the next set of sequences, and once again, the 3's with the highest counts summed across all samples are considered first.

The next designation category is made of sequences that have at least one mismatch to the genome that is not at the 3' end (configurable by setting ‘max_non_3p_mismatches’ in prost.config). A sequence with a single mismatch is designated "4.0". A sequence with a single internal mismatch and up to three mismatches on the 3' end is designated a "4.3p". A sequence with a single mismatch on the 5'-end is designated a "4.5p". And finally a sequence with a single mismatch on the 5'-end and up to three mismatches on the 3'-end is designated a "4.5p3p". All sequences with a single non-3' mismatch are considered before moving on to sequences with more mismatches.

Next, sequences with two mismatches anywhere other than the 3'-end are considered with the same rules as above, and are designated as 5's (5.0, 5.3p, etc). We personally usually run data allowing a maximum of two non-3' mismatches, so the highest designation we usually see is a 5, but this can be expanded. If more mismatches are allowed, the designation continues to increase by 1 for each additional mismatch that is allowed and for example a sequence with 3 internal mismatches will receive a designation 6.

Because there are no biological evidence of miRNA processing that produces indels, we usually disallow indels in alignments through post-filtering of the BBMap alignments. It is however possible to allow indels using BBMap. Nevertheless, indels are reported in the *Prost!* output by the "Gapped?" column in most tabs. If indels are disallowed, it will always be false.

The “isomiRs” tab

This tab has a row for each unique sequence that 1) passes quality filtering, 2) is within the specified size range, 3) meets the ‘minCount’ requirement, and 4) aligns to the genomic reference with at least one hit that is good enough to not be filtered out (parameters adjustable in BBMap).

A - The first column is the sequence itself.

B - The second is the sequence's seed (only relevant when studying miRNAs).

C,D,E - These three columns are indices to navigate between tabs and retrace the grouping. The genomic location index “Loc_idx” (coded as ‘glXXX’), the seed index “Seed_idx” (coded as ‘sXXX’), and the annotation index “Anno_idx” (coded as ‘aXXX’) refers to groups of sequences that are in nucleotide wiggle of each other, have same seed sequence, and have the same annotation, respectively. All sequences that are in nucleotide wiggle of each other will receive the same gl#. Similarly, sequences that have the same seed or the same annotation will receive the same s# or a#, respectively.

F - The “Locations” column lists the position(s) of only the best hit(s) on the genomic reference. Secondary (worse) hits are not listed. When there are multiple alignments of equal quality, they are separated by a ";" unless there are too many alignment locations (TML), and then the cell will simply report "TML:XX" with XX the number of locations it aligns to in the genome. This allows to filter out small RNA that originated form repeats.

G - This column reports the CIGAR string of each alignment oriented from 5' to 3' with respect to the sequence in the first column. BBMap uses the "extended" CIGAR string format. For more on CIGAR strings refer to the official SAM specifications.

H - The designation column informs whether in its genomic location bin the sequence is a #1, 2, 3, 3.3p, 4, etc.

I - The BinStarter column reports the Designation 1 sequence that initiated the genomic location bin.

- Past column I are two sets of columns, each of equal number to the number of samples:
    - the raw counts of each unique sequence (with one column per sample)
    - and the normalized counts of each sequence, which are in Reads Per Million (RPM) with respect to each sample. Normalization is therefore done after filtering for incorrect length reads, low count, and removing reads that don't align to the genome. Each normalized column will add up to 1 million.

- The next set of 8 columns are annotation columns and should be for most of them self-explanatory: ‘[species]_miRNA’, ‘other_species_miRNA’, ‘[species]_miRNA_rev’, ‘other_species_miRNA_rev’, ‘[species]_hairpin’, ‘other_species_hairpin’, ‘other_ncRNA’, and ‘ncRNA_biotype’. In the output, [species] will be the three letter code used for annotating miRNAs in a given species (e.g. ‘hsa’ for human *Homo sapiens*, ‘dre’ for zebrafish *Danio rerio*, etc). The first 4 columns refer to the annotation of mature miRNAs, the two following columns refer to the annotation to hairpins or precursor sequences (pre-miRNAs), and the two last columns refer to annotation to other types of non-coding RNAs and their biotype categorizations.

For a sequence to receive an annotation, *Prost!* requires that the sequence aligns perfectly across its full length to a sequence in the user-provided annotation file. If a given sequence is longer than the sequence present in the annotation dataset, but if the sequence in the annotation dataset aligns perfectly across its full length to the given sequence, the given sequence will then receive an annotation in the ‘_rev’ category (reverse annotation).

- The final set of 8 columns describe the variations of each sequence compared to the main sequence it relates to (the Designation 1 sequence).
    - ‘iso_5p’ reports by how many nucleotides the 5’-end of the sequence differs from the #1 sequence
    - ‘iso_3p’ reports by how many nucleotides the 3’-end of the sequence differs from the #1 sequence
    - ‘iso_add’ reports how many non-templated nucleotides are present at the 3’-end of the sequence
    - ‘iso_snp_seed’ reports whether the sequence displays a nucleotide modification in the seed (nt2-8) compared to the #1 sequence

- ‘iso_snp_central_offset’ reports whether the sequence displays nucleotide modifications at the central offset nucleotide (nt8) compared to the #1 sequence
- ‘iso_snp_central’ reports whether the sequence displays nucleotide modifications in the central region of the miRNA (nt9-12) compared to the #1 sequence
- ‘iso_snp_supp’ reports whether the sequence displays nucleotide modifications in the supplementary region (nt13-17) compared to the #1 sequence
- ‘iso_snp’ reports whether the sequence displays nucleotide modifications anywhere else (nt18-end) compared to the #1 sequence

“by_genomic_location” tab

In this tab *Prost!* bins sequences from the isomiRs tab together based on where they align to the genome as previously described.

A- The first column is the index that links the isomiRs tab to the by_genomic_location tab.

B- This is the sequence of the most expressed isomiR present in the bin and is by definition the BinStarter sequence.

C,D,E- The location and CIGAR string of the main isomiR and its designation (1, 2, 3.3p, etc)

F to M- Annotations at the mature miRNA and hairpin levels, plus the biotype provided by Ensembl. A genomic location bin is annotated with the unique set of all annotations of all its 1's and 2's, joined together. If a bin doesn't have 1's and 2's, or does have 1's and 2's but those 1's and 2's don't have any annotations, the bin is then annotated with the set of all annotations that any of its members have.

N and O- If the sequence location is ambiguous. The value is "True" if there are some individual sequences in the bin that don't have exactly the same mature miRNA or hairpin annotations as the one assigned to that bin. This is quite rare and usually points at annotation problems.

The next set of columns are the sum of the counts of all sequences that contribute to a bin, and then re-normalized counts (renormalization is required because reads that don’t align to the genomic references are not carried over in the by_genomic_location tab).

The next set of columns is where *Prost!* reports for each sample the frequency of selected isomiRs types composing each genomic location bin. The comparison is always made with the BinStarter sequence as reference (and not the sequence present in the annotation database if they are different):

- “%seed_shifted”: The percentage of isomiRs in that genomic location bin that display a seed-shift compared to the BinStarter sequence (different 5’ start which therefore modifies the seed sequence)
- “%seed_edited”: The percentage of isomiRs in that genomic location bin that have a mismatch in the seed compared to the BinStarter sequence (different seed suggests different set of targets)
- “%3'-supplementary_edited”: The percentage of isomiRs in that genomic location bin that have a mismatch in the 3’-supplementary region compared to the BinStarter sequence (This region corresponds to the nucleotides 13 to 17 and are thought to participate in the base pairing stability and thus potentially influence the function)

- “%3'-alternatively_cut”: The percentage of isomiRs in that genomic location bin that are matching the genomic location without mismatch but display 3’ size variations compared to the BinStarter sequence (i.e. they are alternatively cut)
- “%3'-mismatch”: The percentage of isomiRs in that genomic location bin that have additional mismatches at the 3’-end compared to the genomic location(s) they align to (Bases added or edited at the 3’-end)
- “%other_edited”: The percentage of isomiRs in that genomic location bin that have mismatches in other areas than the seed and/or the 3’-supplementary region.

Overall, this tab is particularly useful for tracking these isoforms. The cells are color coded to indicate the percentage of the bin with each specific event (the brighter, the greater the percentage is). It has to be kept in mind however that bins that have low total expression often tend to have bright colors, but this is just noise in the dataset.

“by_annotation” tab

Annotation bins are created by combining genomic locations bins together when they have exactly the same set of annotated miRNAs in the "species_miRNA" column. The first column (“Anno_idx”) is the index that links the isomiRs sequences in the “isomiRs” tab to this annotation tab. The “MainSequence” is the highest count, lowest designation BinStarter of all the genomic location bins that contributed to this annotation bin.

The next column, called “MainSeqMatchesAnnotationFile”, reports “yes” if the BinStarter sequence (i.e. the most expressed isomiR of the annotation bin) corresponds to the sequence that is annotated in that species. “No” means that the main isomiR of the bin is different from the annotated one, and “Multiple” refers to cases where a bin receives two or more annotations.

Prost! let the user decide what to do with annotation bins that have multiple annotations. In our case, we usually split the reads of a doubled annotated bin among the respective singled annotated bins. For ex, if a bin is annotated ‘miRX-5p;miRY-5p’, its reads would be distributed among the ‘miRX-5p’ and the ‘miRY-5p’ bins proportionally to each individual bin expression level. But other users may want to proceed differently so *Prost!* doesn’t pre-compute this step.

The counts and normalized counts are once again summed and re-normalized. Re-normalization is necessary again here because genomic location bins that don’t have an annotation are not carried over in the by_annotation tab. The counts in this tab are generally what is used to look for differential expression with statistical packages such as DESeq2 or EdgeR.

“by_seed” tab

Seed bins are created from sequences in the isomiRs tab that belong to an annotation bin (i.e. have an Anno_idx). All of the sequences that have an annotation and the same seed are binned together even if the Anno_idx differs. All the following columns function similarly to the by_annotation_tab. The annotation columns are built from the set of annotated bins that make up the seed bin. Seed bins are attempting to give a more functional view on expression if we make the assumption that seed is the only factor that matters in miRNA:mRNAs recognition and that all of these isomiRs have potentially the same function. This is a fallacy if taken for granted but can still be useful in many cases.

“candidate_mirror-miRNA” tab

This tab is made up of genomic location bins that are from opposite strands and in genomically close proximity. Precisely, it contains sequences that overlap by at least 5 nt on opposite strands. If users find this tab useful, we'll probably work on adding an option to change this parameter in the future. Mirror-miRNAs are still quite mis-understood but in our tests it revealed each time several known pairs. This tab contains a lot of false-positives so one must be careful when interpreting the data because some reads may come from sequences that are actually not miRNAs.

"candidate_arm_switching" tab

It is known that for some miRNAs one arm can be more expressed than the other one and that this ratio may not always be stable depending on the tissue, the developmental stage, etc and can even be inverted. The goal of this tab is to find when there is a switch from 5p to 3p expression or vice versa between samples. This tab is naive to experimental design and doesn't know anything about replicates, so it just compares within a single sample for every pair of 5p/3p miRNAs that have both 5p and 3p strands annotated and expressed. The ratio of expression 5p/3p are color coded for faster visual inspection of the direction of the change. If all samples for a given line are of the same color it means that all samples expressed mostly the same strand. In contrast, if the color changes between red and blue, then it means that the most expressed arm changes between samples. Similarly to the isomiR variations, it has to be kept in mind that bins that have low total expression often tend to have bright colors and may change color more frequently, but this is just noise in the dataset.

"no_genomic_hits" tab

This final tab lists the sequences that made it through all initial filtering steps but didn't align to the genome with a good enough hit. *Prost!* attempts to annotate these sequences the same way as all the sequences in the isomiRs tab because they may correspond to real miRNAs that are truly present in the genome of the species but absent in the genomic reference used. A location and a CIGAR column inform where and how well a sequence would have aligned if *Prost!* hadn't filtered it out as a bad hit.

Conclusions

Prost! was designed with the objective of letting the user have access to all the data and steps of analysis to fully have the ability to understand how the results were obtained.

Please let us know if there are questions, comments, or things that appear not clear. Any feedback is welcome.

Thanks for using *Prost!*