REPdenovo: A software pipeline to de novo reconstruction and analysis of repetitive genomic sequences from sequence data

User Manual

Version 1.0.0

March 2, 2015

Chong Chu and Yufeng Wu
CSE Department, University of Connecticut Storrs, CT 06269, U.S.A.
Email: chong.chu,ywu@engr.uconn.edu

 A manuscript for this software is under preparation. If you use this program in a publication, please cite the following reference:
Chong Chu, Rasmus Nielsen and Yufeng Wu, De novo reconstruction and analysis of repetitive genomic sequences from sequence reads, 2015. Please note this is a tentative title. Please check back this site for more up-to-date information on citing this software.

1 Getting Started with REPdenovo

1.1 Program availability

REPdenovo is mainly written in Python and C++. Executables for popular platforms such as Linux 32 bits or 64 bits and MacOS are downloadable from this site. Files can be downloaded using "Save Link/Target As..." After downloading the softwares, you may need to change file access permissions (e.g. chmod u+x stells-linux). In case that you want to compile the code yourself, source code is also available for download at the above URL. To compile the code, first put the gzip file in the directory youd like and unzip it: use gunzip and tar commands such as:
▷ gunzip ¡stells-src.tar.gz¿
▷ tar -xvf ¡stells-src.tar¿
Then type:
▷ make at the prompt. This creates an executable called stells, which can be run by typing
▷ stells at the prompt. You will need to specify some input options - see below.

1.2 What is REPdenovo?

The main objective of REPdenovo is, given raw sequence reads from an organism, reconstruct consensus repeat sequences that are of low divergence rate. REPdenovo also provides analysis of repeats. This includes determining the amount of sequence reads that can be mapped to these consensus repeats, and also determining the basic types of repeats: are they tandem repeats or interspersed repeats?

1.3 How does REPdenovo work?

The basic idea of REPdenovo is that k-mers (the length-k segments of reads) in repeats are likely to have much higher frequencies than k-mers coming from non-repetitive regions. When the divergence rate of copies of a repeat is relatively low, a k-mer in a copy of a repeat has high probability of matching the corresponding k-mer in the consensus repeat. So, one may assemble consensus repeats by assembling frequent k-mers. This process is shown in Figure **??**. There is a program called REPARK (Nucleotide Acid Research) that performs assembly of frequent k-mers, although it does not point out repeats of low divergence are likely to be assembled.

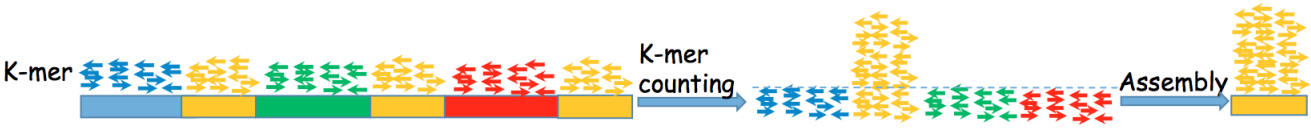

Figure 1: *Illustration of repeat assembly from short sequence reads. Counting of k-mers shows that k-mers from repetitive regions (yellow) are more frequent than expected. Then we can assemble the consensus repeats by assembling these frequent k-mers.*

In reality, repeats are more complicated than what is shown here. For example, there may be two different repeats sharing common segments. Also, there may exist regions in a consensus repeats that are more variable than the other parts of the consensus, and thus only parts of consensus can be assembled directly from frequent k-mers. REPdenovo is designed to address these and other issues in repeat assembly. REPdenovo also provides statistics for the amount of sequence reads that are mapped to constructed repeats.

2 Functionalities and Usage of REPdenovo

Based on the simple idea of frequent k-mer assembly, REPdenovo provides much more functionalities than REPARK. REPdenovo supports the following main functionalities.

1. Assembly. This step performs k-mer counting. Then we find frequent k-mers whose frequencies are over certain threshold. We then assemble these frequent k-mers into consensus repeats (in the form of contigs).

2. Analysis. We map the sequence reads to the repeat contigs. We provide statistics on how many reads are mapped to the repeats. We also improve the quality of repeat contigs based on the mapping coverage information (e.g. trimming repeat contigs if

coverage is too low). We also classify the repeat contigs into two basic types: interspersed and tandem repeats.

3. Scaffolding. We use paired-end reads to connect repeat contigs into scaffolds.

2.1 Dependencies

REPdenovo needs the following tools to be installed in the machine you are working on.

1. A k-mer counting tool. REPdenovo uses Jellyfish program for performing k-mer counting. Jellyfish can be downloaded from `https://github.com/gmarcais/Jellyfish`.

2. An reads assembler. REPdenovo uses Velvet at this point. In the future, we may support different assembler. Velvet can be downloaded from: `https://www.ebi.ac.uk/~zerbino/velvet/`. Caution: if you want to assemble k-mers that are longer than 30 bp, you need to recompile Velvet to let it work with longer sequence length. For example: make MAXKMERLENGTH=60. This makes Velvet work for k-mer length up to 60.

3. Reads mapping. REPdenovo uses BWA. BWA can be downloaded from `http://bio-bwa.sourceforge.net/`.

4. Sequence processing utilities. These include the commonly used SAMtools. Our code also uses BAMtools (`https://github.com/pezmaster31/bamtools`).

2.2 Preparing inputs

REPdenovo takes sequence reads in the FASTQ format (uncompressed or compressed). REPdenovo needs a configuration file, which tells REPdenovo the basic settings. The following shows a typical settings file.

Here, we give an explanation on the parameters. In general, you should have all the entries shown in the figure. For some parameters, the values shown in the example are perhaps those that you should use (especially those are said to not change below).

1. MIN_REPEAT_FREQ. This is the cutoff of k-mers that are considered to be frequent for assembly. Note that this is the relative to the average coverage of the sequence reads. The average coverage of the sequence reads is calculated by the number of reads, reads length and the genome size.

2. RANGE_ASM_FREQ_DEC and RANGE_ASM_FREQ_GAP: these are used for assembly. Usually you don't need to change these.

3. K_MIN, K_MAX and K_INC: the smallest value, maximum value and increment of K. REPdenovo can use different K. In the example shown in the figure, three K values will be used: 30, 40 and 50.

4. K_DFT: default value of K value. This is equivalent of setting K_MIN = K_MAX = K_DFT.

```
MIN_REPEAT_FREQ 100
RANGE_ASM_FREQ_DEC 2
RANGE_ASM_FREQ_GAP 0.8
K_MIN 30
K_MAX 50
K_INC 10
K_DFT 30
READ_LENGTH 100
JELLYFISH_TREADS 3
ASM_NODE_LENGTH_OFFSET -1
MIN_CONTIG_LENGTH 100
COV_DIFF_CUTOFF 0.5
MIN_SUPPORT_PAIRS 20
MIN_FULLY_MAP_RATIO 0.2
TR_SIMILARITY 0.85
JELLYFISH_PATH /scratch2/chongchu/jellyfish-2.1.4/bin/
VELVET_PATH /scratch2/chongchu/velvet-master/
BWA_THREADS 5
OUTPUT_FOLDER ./human_HG01886_keep_used_no_upper_bound/
VERBOSE 1
```

Figure 2: *Settings of REPdenovo.*

5. READ_LENGTH: length of reads.

6. JELLYFISH_THREADS: how many threads to use to run Jellyfish.

7. ASM_NODE_LENGTH_OFFSET: if set to -1, then require each k-mer in the repeat be frequent. That is, all k-mers in a repeat is considered to be frequent.

8. COV_DIFF_CUTOFF, MIN_SUPPORT_PAIRS, MIN_FULLY_MAP_RATIO, : used by REPdenovo in improving quality of assembled repeats. You don't usually need to change these.

9. TR_SIMILARITY: REPdenovo merges two assembled repeats if their similarity is over this threshold.

10. JELLYFISH_PATH: set to the path of Jellyfish executable.

11. VELVET_PATH: set to the path of Velvet executable.

12. BWA_THREADS: the number of threads BWA is set to use.

13. OUTPUT_FOLDER : where to output the results. It is relative to the installation folder of REPdenovo.

14. VERBOSE. If set to be 1, output more information about the current running states of REPdenovo.

2.3 Basic usage

There are two basic operation modes in STELLS. In either case, you must provide a gene tree file using the -g option (with the gene tree name following the -g). First, you can specify -s option (followed by the species tree name) to let STELLS focus on this particular species tree stored in the species tree file. That is,

▷ ./stells-linux -g <gene-tree-file-name> -s <species-tree-file-name>

The format of these two files are described above. In this mode, STELLS calculates the coalescent likelihood of the gene trees on the given species tree (i.e. the sum of log-likelihood of each gene tree on the species tree). Then, STELLS will try to optimize the branch length of the species tree in order to find higher likelihood for the given species tree, and the species tree with optimal branch length will be output.. Note: optimizing branch length will be slower. If you do not want to optimize branch length, you can use the -B option as follows to omit the branch length optimization:

▷ ./stells-linux -B -g <gene-tree-file-name> -s <species-tree-file-name>

2.4 Command line options

For ease of reference, I now provide the list of command line options.

The following list is mandatory.

1. -g <filename>. This species the file that contains one or more gene trees. Each line species a gene tree in the Newick format. Do not include anything else in this file. Note that branch lengths can be included but will be ignored by STELLS. As explained before, the taxon name of gene trees should match the taxon name you have for the species under study. You may have or or more gene alleles for the same species and you should use the same name (i.e. there can be duplicate taxon names in the gene trees). It is important to note that you need to have the same set of taxa in each of the gene trees. That is, each gene tree must have at least one gene allele for each species/population appeared in the gene tree file.

The following lists options that are optional.

1. -s <filename>. This species the species tree to evaluate. If -s is given, STELLS only searches for optimal branch lengths of the given species tree but will not search for optimal species tree topologies.

3 Credits

REPdenovo is developed by Chong Chu and Yufeng Wu. Rasmus Nielsen (UC Berkeley) provided many insights to the project.