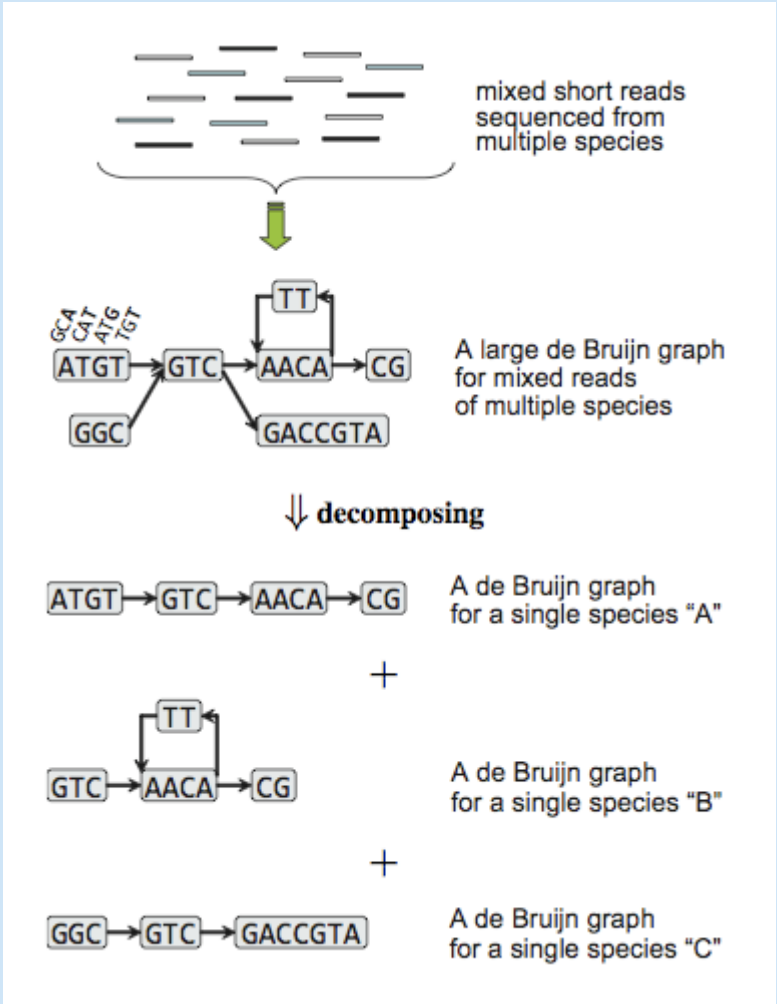
Installation and Getting Started

Afiahayati, Sato K, Namiki T, Hachiya T, Tanaka H, Sakakibara Y.

Abstract

Motivation: An important step of "metagenomics" analysis is the assembly of multiple genomes from mixed sequence reads of multiple species in a microbial community. Most conventional pipelines employ a single-genome assembler with carefully optimized parameters and post-process the resulting scaffolds to correct assembly errors. Limitations of the use of a single-genome assembler for *de novo* metagenome assembly are that highly conserved sequences shared between different species often causes chimera contigs, and sequences of highly abundant species are likely mis-identified as repeats in a single genome.

Methods: We modified and extended a single-genome and de Bruijn-graph based assembler, Velvet, for *de novo* metagenome assembly. Our fundamental ideas are first decomposing de Bruijn graph constructed from mixed short reads into individual sub-graphs and second building scaffolds based on every decomposed de Bruijn sub-graph as isolate species genome.



Citation: Afiahayati, Sato K, Namiki T, Hachiya T, Tanaka H, Sakakibara Y. Installation and Getting Started. **protocols.io**

dx.doi.org/10.17504/protocols.io.d5k84v
Published: 14 Jan 2016

Guidelines

What is improved?

Longer N50 size: On artificially generated simulation datasets of short sequence reads, it is proven that MetaVelvet assembled metegenomic read data with significantly longer N50 sizes than single-genome assemblers (Velvet and SOAPdenovo) and another meta-genome assembler (MetaIDBA) as shown in the table below.

Table 1. Accuracy comparison of assembly softwares (80 bp)

Metagenome dataset	Separate assembly	MetaGenomic assembly			
	Velvet	MetaVelvet	Velvet	SOAPdenovo	Meta-IDBA
Order20 (total genome size = 71,929,175bp; 93,423,332 reads)					
Num. scaffolds	685	813	924	5,998	2,678
Total scaffold length	71,009,045	71,053,228	48,450,203	70,296,665	70,312,381
N50 size (bp)	288,838	268,350	142,471	43,796	55,575
Chimeric scaffold length (%)	0.00	0.00	0.46	0.00	0.00
Cover rate (%)	98.38	98.25	67.48	95.67	96.98
Required CPU time (sec.)	4,994	8,685	7,076	11,564	7,375
Required memory (GB)	7.04	56.61	54.07	62.42	15.15
Family20 (total genome size = 84,552,832bp; 113,680,114 reads)					
Num. scaffolds	784	1,019	2,889	9,039	4,421
Total scaffold length	83,275,357	83,322,440	65,789,192	81,739,588	81,990,799
N50 size (bp)	313,454	257,853	76,239	27,510	39,961
Chimeric scaffold length (%)	0.00	0.45	0.02	0.03	0.00
Cover rate (%)	98.12	97.81	77.60	94.09	96.03
Required CPU time (sec.)	9,585	11,409	9,813	14,803	12,664
Required memory (GB)	13.15	72.06	68.98	62.48	23.11
Genus20 (total genome size = 88,595,850bp; 103,990,387 reads)					
Num. scaffolds	1,288	2,325	3,633	10,282	10,643
Total scaffold length	86,489,808	84,342,495	53,450,902	79,334,848	74,808,521
N50 size (bp)	279,359	239,061	74,182	16,194	12,773
Chimeric scaffold length (%)	0.00	1.56	0.00	0.08	0.00
Cover rate (%)	98.17	97.13	73.31	91.73	90.93
Required CPU time (sec.)	7,275	10,395	8,712	12,889	15,071
Required memory (GB)	11.12	63.22	60.43	62.45	16.75
Species20 (total genome size = 85,450,435bp; 98,817,303 reads)					
Num. scaffolds	818	3,447	2,403	9,317	6,657
Total scaffold length	83,865,679	80,628,784	40,619,181	70,762,160	64,880,992
N50 size (bp)	339,109	152,531	100,819	14,471	26,571
Chimeric scaffold length (%)	0.00	0.93	0.00	0.01	0.00
Cover rate (%)	97.79	94.56	60.29	84.62	82.50
Required CPU time (sec.)	7,618	12,001	8,775	12,858	20,755
Required memory (GB)	7.68	64.06	61.23	62.46	17.32

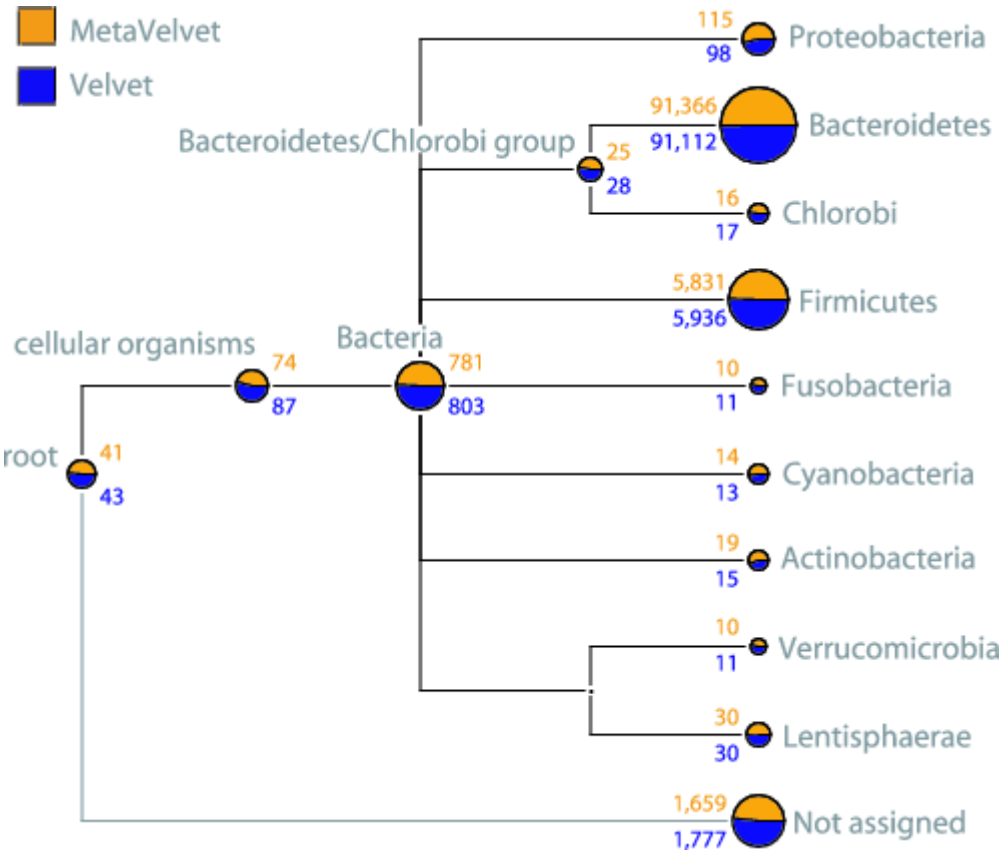
All computations were executed with Intel(R) Xeon(R) E5540 processors (2.53 GHz), with 48 GB physical memory except for a few cases. The figures in "Separate assembly" show the results of single genome assembly from pure sequence reads of each single isolate genome, which were not available in real-data analysis. MetaVelvet, Velvet, and SOAPdenovo were run with default parameters except for setting *k*-mer size at 51. Meta-IDBA was run with default parameters except for setting the maximum *k*-mer size at 50.

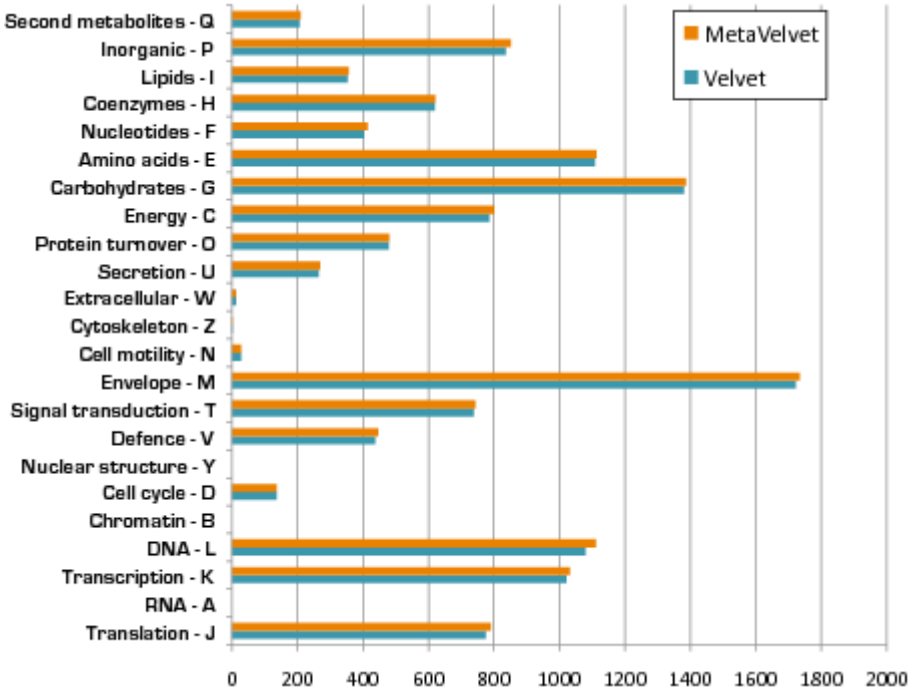
Improved gene prediction: We applied Velvet and MetaVelvet to real metagenomic dataset (SRR041654 and SRR041655; human gut microbiome data downloaded from NCBI sequence read archive). Then, MetaGeneAnnotator was used to predict a gene set of the microbiome. From the results, it is demonstrated that MetaVelvet can efficiently avoid chimeric scaffolds, resulting in longer scaffolds, increased number of predicted genes, and decreased fraction of partial genes.

	Velvet	MetaVelvet
Scaffolds[a]		
Num. scaffolds	6,697	5,757
Total scaffold length (bp)	58,794,997	64,395,790
Average scaffold length (bp)	8,779	11,186
N50 size (bp)	26,815	45,512
Protein-coding genes[b]		
Num. genes	50,349	55,416
Num. partial genes[c]	7,507 (14.9%)	6,871 (12.4%)
Average gene length (aa)	335 ± 251	334 ± 251

[a] Short scaffolds whose length is less than 1,000 bp are not included in this statistics. [b] Protein-coding genes were predicted only from scaffolds whose length is greater than or equal to 1,000 bp. [c] Partial genes denote genes whose 5' and/or 3' end is not included in a scaffold.

Widely applicable to metagenomic analyses: We also compared the results of metagenomic analyses, such as taxonomic content analysis, functional composition analysis, and pathway mapping analysis, between Velvet and MetaVelvet. The results demonstrate that taxonomic content can be estimated more clearly when based on the MetaVelvet scaffolds than when based on the Velvet scaffolds as shown in the figure below left (the number of genes that cannot be assigned to phylum-level taxonomy is substantially decreased). Moreover, the result of functional composition analysis also shows that MetaVelvet is more appropriate for metagenomics purpose than Velvet as shown in the figure below right (the number of genes assigned to any functional category is improved).





Source Code (Release Notes)

The following source codes is freely available under GNU General Public License.

- MetaVelvet-v1.2.01 (May.29,2012)
  - enable to use paired-end information in a graph decomposition step.
  - three options are newly added in order to specify how to use paired-end connections for graph decomposition: -valid_connections, -noise_connections, and -use_connections.
  - for details, please see Advanced topic 3: how to use paired-end information for graph decompositoin.

- MetaVelvet-v1.1.01 (Oct.19,2011)
  - this version is based upon Velvet 1.1.06.
  - fix a frequent bug: no peak detection error.
  - fix a frequent bug: infinite loop error.
  - modify the graph split algorithm: an exception for "knock-turn" loops.
  - modify the graph split algorithm: split priority is now based on peak IDs rather than node IDs.

- MetaVelvet-v0.3 (Jan.14,2011)
  - this version is based upon Velvet 0.7.62.

Source codes at these versions can be downloaded from here.

Requirements (>=1.1.01)

- Required libraries:
    - zlib library (>= 1.2.3) - zlib library is also included in the Velvet distribution.

- Required programs:
    - velveth (>= 1.1.06) - included in the Velvet distribution.
    - velvetg (>= 1.1.06) - included in the Velvet distribution.
    - g++ (>= 4.3.2) - for compile.

- Recommened environment:
    - OS : standard 64bit Linux (It can in theory function on a 32bit environment, but not recommended).
    - RAM: at least 12GB (more is no luxury).

Frequently Asked Questions

- **Q: meta-velveth** is not generated in the new version (>= 1.1.01). Is it problem?
  A: This is not problem. The usage of MetaVelvet is changed when the version 1.1.01 is released, and the new version does not include **meta-velveth**. Instead, please use **velveth**, **velvetg**, and **meta-velvetg**.

- **Q:** When only one coverage peak is detected (or manually input), is there any difference between MetaVelvet and Velvet algorithms?
  A: There is no substantial difference. In such cases, **meta-velvetg** moves to "single-peak mode" and graph splitting functions in **meta-velvetg** is not called. Instead of graph splitting functions, standard velvet functions are called in such cases.

- **Q:** What's the difference in working procedures between **velvetg**, **meta-velvetg** (<= 0.3.1), and **meta-velvetg**(>= 1.1.01)?

- **A:** The following is the working procedure of **velvetg**, **meta-velvetg** (<= 0.3.1), and **meta-velvetg** (>= 1.1.01):

  velvetg & meta-velvetg(<=0.3.1) :
  Load Sequences & Roadmaps file
  -> Generate PreGraph file
  -> Generate Graph or Graph2 file
  -> Generate contigs.fa and LastGraph

  meta-velvetg (>= 1.1.01):
  Load Sequences & Roadmaps & Graph2 file
  -> Generate meta-velvetg.contigs.fa and meta-velvetg.LastGraph

- **Q:** Is version compatibility between Velvet and MetaVelvet fully tested?
  A: Version compatibility between Velvet-1.0.06 and MetaVelvet-1.1.01 is fully tested.

Troubleshooting

Trouble: When drawing *k*-mer coverage histogram (as in the "Advanced topics 1" section), the following warning messages is appeared:

```
> weighted.hist(data$shot1_cov,data$lgth,breaks=seq(0,200,1))
Warning messages:
1: In min(x, na.rm = na.rm) :
no non-missing arguments to min; returning Inf
2: In max(x, na.rm = na.rm) :
no non-missing arguments to max; returning -Inf
3: In weighted.hist(data$shot1_cov, data$lgth, breaks = seq(0, 200, :
Areas will not relate to frequencies
```

Solution: This warning (error) is caused by "Inf" values in the Graph2 node stats. Accordingly, by removing "Inf" values from the Graph2 stats, the error is resolved:

```
$ head -n 1 meta-velvetg.Graph2-stats.txt \
      > meta-velvetg.Graph2-stats.rmInf.txt
$ perl -ne '{print $_ unless /Inf/;}' \
      meta-velvetg.Graph2-stats.txt \
      >> meta-velvetg.Graph2-stats.rmInf.txt
$ R
> library(plotrix)
> data = read.table("meta-velvetg.Graph2-stats.rmInf.txt", header=TRUE)
> weighted.hist(data$shot1_cov,data$lgth,breaks=seq(0,200,1))
```

Protocol

Install MetaVelvet

Step 1.

Download source code from here, and decompress the tar ball.

LINK:
http://metavelvet.dna.bio.keio.ac.jp/src/

cmd COMMAND

```
~$ tar zxvf MataVelvet-v1.1.01.tgz
```

decompress tar ball

Install MetaVelvet

Step 2.

Alternatively, up-to-date source code is available from github MetaVelvet project.

LINK:
https://github.com/hacchy/MetaVelvet

cmd COMMAND

```
~$ git clone git://github.com/hacchy/MetaVelvet.git
```
git clone MetaVelvet project

Install MetaVelvet

Step 3.

Change directory to the MetaVelvet directory.

cmd COMMAND
```
~$ cd MetaVelvet-v1.1.01
```
moving into MetaVelvet directory

Install MetaVelvet

Step 4.

Compile MetaVelvet execution files.

cmd COMMAND
```
~/MetaVelvet-v1.1.01$ make ['MAXKMERLENGTH = k'] ['CATEGORIES = cat']
```
k - the maximum k-mer size you like to use. cat - the maximum number of read categories (i.e., maximum number of libraries in different insert lengths)

Install MetaVelvet

Step 5.

Then, an executable file, **meta-velvetg** will be created (>= 1.1.01).

Install MetaVelvet

Step 6.

Copy the two executable files to **/usr/bin** or a directory you like to install.

cmd COMMAND
```
~/MetaVelvet-v1.1.01$ cp meta-velvetg /usr/bin/
```
copy executable files to /usr/bin

Getting started

Step 7.

MetaVelvet (>= 1.1.01) uses **velveth** and **velvetg** outputs for metagenomics assembly. Three files ("Sequences", "Roadmaps", and "Graph2") are needed for running **meta-velvetg**.

Getting started

Step 8.

Download a small dataset from here, and decompress the tar ball.

LINK:
http://metavelvet.dna.bio.keio.ac.jp/data/

cmd COMMAND
```
~$ tar zxvf HMP.small.tar.gz
```
decompress

Getting started

Step 9.

Execute **velveth** to import read sequences and construct *k*-mer hash table.

cmd COMMAND
```
~$ velveth out-dir 51 \
        -fasta -shortPaired \
        HMP.small/SRR041654_shuffled.fasta \
        HMP.small/SRR041655_shuffled.fasta
```

NOTES

Bonnie Hurwitz 16 Nov 2015

Please check that "*out-dir*/Sequences" and "*out-dir*/Roadmaps" files are created. Note that *k*-mer size has important effect on assembly results. If your short read data is longer than 65bp, we recommend *k*-mer longer than 51.

Getting started

Step 10.

Execute **velvetg** to construct an initial *de Bruijin* graph.

cmd COMMAND

```
~$ velvetg out-dir -exp_cov auto -ins_length 260
```

executing velvetg

NOTES

Bonnie Hurwitz 16 Nov 2015

Please check that "*out-dir*/Graph2" file is created. Please use **-exp_cov auto** option, otherwise **meta-velvetg** cannot be executed.

Getting started

Step 11.

Execute **meta-velvetg** to output scaffold sequences.

cmd COMMAND

```
~$ meta-velvetg out-dir [-ins_length 260] | tee logfile
```

Executing meta-velvetg

NOTES

Bonnie Hurwitz 16 Nov 2015

Please check that "*out-dir*/meta-velvetg.contigs.fa" file is created. The FASTA file is the main assembling results.