CVTree 3.0
User's Manual

(Standalone Version)

Guanghong Zuo
ghzuo@fudan.edu.cn

October 20, 2020

Contents

1 Introduction

CVTree stands for **Composition Vector Tree** which is the implementation of a cluster alignment-free algorithms to generate dissimilarity matrices from comparatively large collection of DNA or Amino Acid sequences, preferably genome data, for phylogenetic studies. It is first developed to infer evolutionary relatedness of Bacteria and Archaea (Qi *et al.*, 2004*b*), and then successfully applied to viruses, chloroplasts, and fungi.

There are two available ways to use the algorithms:

1. **CVTree Web Server** which has been published twice in the Web Server Issues of *Nucleic Acids Research*, (Qi *et al.*, 2004*a*) and (Xu and Hao, 2009). The latest released CVTree Web Server, CVTree3 Web Server (Zuo and Hao, 2015), have two identical but independent installations:

   - CVTree3 Web Server on Fudan University, Shanghai:
     `http://tlife.fudan.edu.cn/cvtree/`
   - CVTree3 Web Server on Beijing Institute of Genomics, Beijing:
     `http://bigd.big.ac.cn/cvtree`

   There are 3000+ inbuilt whole genomes in the CVTree3 web servers which covered 1700+ prokaryotic species. A much more powerful web server, CVTree4 Web Server, which included 100000+ genomes and covered 8000+ prokaryotic species was in testing on Aliyun.

   - Testing Web Server on Aliyun:
     `http://cvtree.online`

2. **CVTree Standalone Version** which is provided to those who are interested in the intermediate results, e.g., the collection of all CVs, or deal with extremely huge datasets of their own, as well as bioinformatic developers. We provide also a few options and tools that were not available in the Web Server versions.

This manual is for the CVTree Standalone Version.

2 The Installation and Testing

CVTree is distributed via source code. After download the source codes. There are two ways to compile the source codes of CVTree: the classical way, compiled and performed the in normal like other Unix-like programs; the docker way.

2.1 Normal Unix-like Mode

All the programs was implemented in C++, some compile tools and library is required, and other libraries optional for special purposes.

2.1.1 Preparation

- cmake $\geq$ 3.0
- g++ $\geq$ 4.8 or other compiler supporting C++11 standard
- require library: libz
- compiler with support OpenMP for parallel (*option*)
- Library (*option*): netcdf, netcdf_cpp, libhdf5 for c++ [1]

2.1.2 Compile by Cmake

1. unzip the package file and change into it
2. mkdir build and change into it
3. cmake .. and some options you wanted
4. make
5. make install (*option*)

2.1.3 Testing with Example

If this is the first time you use CVTree package, please go to the "example" folder. Edit "list" to include the genome names, and run the cvtree command to get the phylogeny tree by:

```
../build/cvtree -G faa
```

More detail of the command usage can be obtain by '-h' option or read the follow sections.

2.2 Run CVTree in Container

The containers allows users run programs on both Windows and Linux/MacOS, and transfer the programs easily. To employ the container with CVTree, you should install `docker` at first. You can download docker free and reference from `https://docs.docker.com/install/` to how to install it. After install docker, basic usages for CVTree in container are shown blow:

1. Obtain image: You can build the cvtree docker image based on `Dockerfile` in the source code by command

   ```
   docker build -t="cvtree" .
   ```

[1] It seem that the hdf5 libraries in Anaconda was not working in our testing

Here option '-t' set the image name. After build image, you can delete the dangling images for build by `docker image prune`. This will save much hard disk space. You can also download prebuilt cvtree image from internet by command:

```
docker pull ghzuo/cvtree .
```

In this step, an image with cvtree programs will obtained.

2. Start container from image: run the follow command in the CVTree directory, i.e. the directory which include the 'example' directory of the CVTree

   ```
   docker run --rm -it -v $PWD/example:/root/data cvtree
   ```

   In this step, you will enter the cvtree container, and the "example" folder of this project will be find in the "data" folder. Change path to the data folder, and run

   ```
   cvtree -G faa
   ```

   You will get the result for eight genomes in the `list` file. You can change the path `'$PWD/example'` to your own data directory.

3. Exit and stop container: `exit` in docker terminal.

4. Run cvtree in a temporary container by one command without enter the container:

   ```
   docker run --rm -v $PWD:/data -w /data cvtree cvtree -G faa
   ```

5. More usage for `docker` can reference `https://docs.docker.com/`.

3 Programs and Command-Line Options

3.1 Scheme of CVTree

Figure 1 shows the scheme of CVTree. In CVTree, the genome sequences were cut into many small segments with k length. The composition vector (CV) of the genome was obtained based on the amount of these k-mers with or without statistical models. Here the genome sequence can be recorded in protein, RNA and DNA. In this way, we converted the genome sequence into a vector, which possess many good attributions in mathematics, e.g. alignment of vectors is independent with the comparing order. Then a dissimilarity matrix was obtained based on these composition vectors. And the phylogenic tree was inferred based on the dissimilarity matrix. To sum up, there are three steps for CVTree:

1. Convert a genome sequence to a composition vector. And two method, Hao and Counted, are provided in CVTree.

2. Calculate the dissimilarity matrix based on the composition vectors. And Three methods, Cosine, InterList, and InterSet are provide in CVTree.

3. Infer the phylogenic tree by dissimilarity matrix. And the Neighbor-Joint is provided in CVTree.

The classical CVTree algorithm was based on the Markov model and first released by Prof. Bailin Hao in 2004 (Qi *et al.*, 2004*b*). Recently we added some new algorithms into CVTree software. The detail of algorithms are described in the subsequent section. We noted that the CVTree is an open frame for the alignment-free phylogeny, and new methods will added in feature.

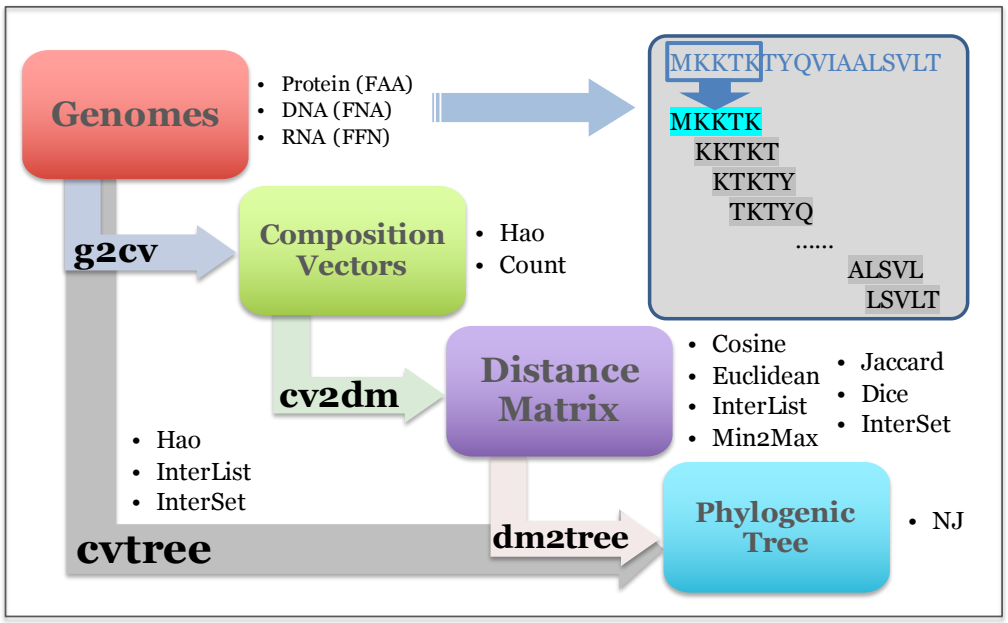

Figure 1: The Scheme of CVTree

In the standalone version CVTree, there are two ways to implement the CVTree algorithms to obtain the phylogenic tree from the fasta files:

- all in one command: cvtree
- step by step commands: g2cv, cv2dm, dm2tree

We suggest you use the 'all in one command' like the way in testing..

IMPORTANT: You need prepare the fasta files of genomes, one file for one genome, and a list file, which includes the genome names you want calculating.

3.2 All in One Command

3.2.1 The Workflow of cvtree command

The `cvtree` command is the main command for this software. It inbuilt an automatic work flow to obtain the phylogenic tree from the fasta files directly. Figure 2 shows the flowchart of the

`cvtree` command. It builds the dissimilarity matrix based on the reference dissimilarity matrices, and fills the missing dissimilarity between two genomes automatically. All you need are prepare the fasta files of genomes, a genome list, and the options according the requirements of your project. We noted that in method selection of `cvtree`, you can set the selection for cv method, distance method, and tree method by a string split by colon, e.g. `cvtree -m Counted:Cosine:NJ` ···. And we provided three markers (see figure 1): `Hao` for "Hao:Cosine:NJ"; `InterList` for "Count:InterList:NJ"; and `InterSet` for "Counted:InterSet:NJ".

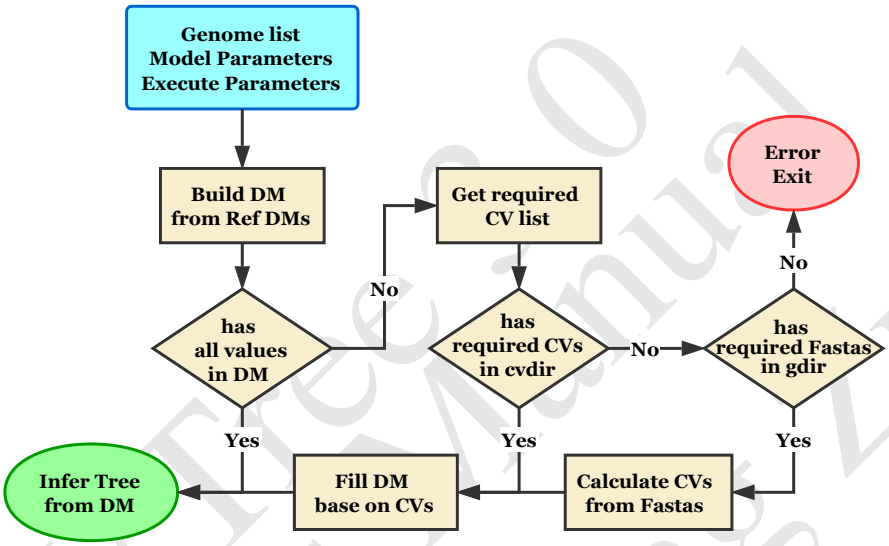

Figure 2: The Flowchart of the cvtree command

3.2.2 The command usage

- cvtree – Generate Newick tree from input genomes

```
cvtree
 [ -d <dm> ]       Output distance matrix name,
                   default: <Method><Suffix><K>
 [ -t <nwk> ]      Output newick file name,
                   default: <Method><Suffix><K>.nwk
 [ -G <gdir> ]     Super directory of Input genome file,
                   default: <current directory>
 [ -g faa ]        the type of genome file,
                   default: faa
 [ -V <cvdir> ]    Super directory of cv files
 [ -i list ]       Genome list for distance matrix,
                   default: list
 [ -k '5 6 7' ]    values of k,
```

```
                  default: K = 5 6 7
  [ -r <matrix> ] Reference distance matrices, split with ','
  [ -M <N> ]      Running memory size as G roughly,
                  default 80% of physical memory
  [ -m Hao ]      Method for cvtree Hao/InterList/InterSet,
                  default: Hao
  [ -R ]          Refer the output distance matrix
  [ -q ]          Run command in quiet mode
  [ -h ]          Display this information
```

3.3 Step by Step Commands

You can also implement your project step by step by three programs, i.e. `g2cv`, `cv2dm`, `dm2tree`. The function and usage of these three programs are show blow:

- g2cv – Generate CV files from input data

```
g2cv
  [ -G <gdir> ]     input genome file directory
  [ -V <cvdir> ]    output cv directory
  [ -i list ]       input species list, default: list
  [ -f <Fasta> ]    get cv for only one fasta
  [ -k '5 6 7' ]    values of k, default: K = 5 6 7
  [ -g faa ]        the type of genome file, default: faa
  [ -m Hao/Count ]  the method for cvtree, default: Hao
  [ -q ]            Run command in quiet mode
  [ -h ]            Display this information
```

- cv2dm – Generate distance matrix based on CV files

```
cv2dm
  [ -o <dm> ]       Output distance matrix,
                    default: <Method><Suffix>.h5
  [ -V <cvdir> ]    Super directory of extend cv files
  [ -i list ]       Genome list for distance matrix,
                    default: list
  [ -s <Suffix> ]   Suffix of the cvfile, default: .faa.cv6
  [ -r <matrix> ]   Reference distance matrices, split with ','
  [ -M <N> ]        Running memory size as G roughly,
                    default 80% of physical memory
  [ -m Cosine ]     Method for distance:
                    Cosine/Euclidean based on vector;
```

```
                    InterList/Min2Max based on count of kmers;
                    InterSet/Jaccard/Dice based set of kmers.
                    default: Cosine
  [ -q ]            Run command in quiet mode
  [ -h ]            Display this information
```

- dm2tree – Generate Newick tree from distance matrix by using the Neighbor-Joint method

```
dm2tree
  [ -d infile ]     Input distance matrix,
                    default: dist.matrix
  [ -o Tree.nwk ]   Output Newick tree,
                    default: Tree.nwk
  [ -i <list> ]     index list of selection genomes
                    of the distance matrix, if no defined,
                    whole distance matrix are used
  [ -C ]            Use the netcdf input format,
                    default false
  [ -q ]            Run command in quiet mode
  [ -h ]            Display this information
```

3.4 Tools

In cvtree software, the final output phylogenic tree is in Newick format. And in the implement of cvtree, the intermediate results are also saved to avoid the repeat calculation in the next time. These intermediate results are in binary format in default, and can not be examined directly. We prepared some tools to examined these binary files. The function and usage of these tools are show blow:

- cvdump – convect the binary cv file to acsii file

```
cvdump  -i <cvfile>  input file name
  [ -g faa|ffn ]     the type of genome file, default: faa
  [ -n ]             output the number code,
                     default: the letters
  [ -h ]             display this information
```

- getdist – get the distance from distance matrix

```
getdist
  [ -d infile ]     distance matrix files list separate by ","
                    default: infile
  [ -i selfile ]    the select genomes, default: none
  [ -H ]            whether output html table, default: false
```

```
[ -l ]              get the list of species of distance matrix
[ -h ]              display this information
```

- diffmtx – whether two matrixes is equal

```
diffmtx <dm1> <dm2>   input the two distance matrices
```

4 Algorithm

4.1 The Classical CVTree Method: Hao Method

As the description above, there are three steps to obtain the phylogenic tree based on the genomes by CVTree, i.e. calculating composition vector form genome, building dissimilarity matrix from composition vectors, and inferring phylogenic tree from dissimilarity matrix. Here we describe the classical algorithm of CVTree briefly. For more detailed description of the algorithm please consult reference (Qi *et al.*, 2004*b*).

1. Fix a string length k ($k \in [3, 14]$ for Amino Acid sequences and $k \in [3, 30]$ for nucleotide sequences). Read in the sequence collection of each species separately. Count the number of all k, $k-1$ and $k-2$ tuples for a species. And the counts of the K-tuples are recorded as $f(a_1a_2\cdots a_k)$ with a_i indicates the letter of the residue of protein or nucleic acid. Thus the probability of a k-tuple in the genome sequence is:

$$p(a_1a_2\cdots a_k) = \frac{f(a_1a_2\cdots a_k)}{N_k}, \tag{1}$$

here N_k is the total number of k-tuples. It is obvious that a k-tuple $a_1a_2\cdots a_k$ can be obtained by adding a a_k after $a_1a_2\cdots a_{k-1}$. As the Markov model, the probability of a k-tuple $a_1a_2\cdots a_k$ can be predicted by the probability of its subsequence $a_1a_2\cdots a_{k-1}$ by a conditional probability, i.e.:

$$\tilde{p}(a_1a_2\cdots a_k) = p(a_k|a_1a_2\cdots a_{k-1})p(a_1a_2\cdots a_{k-1}) \tag{2}$$

we supported that the conditional probability is independent with the first letter, then the conditional probability can be calculated by the probability of $k-1$-tuple and $k-2$-tuple, i.e.:

$$\begin{aligned} \tilde{p}(a_1a_2\cdots a_k) &\approx p(a_k|a_2\cdots a_{k-1})p(a_1a_2\cdots a_{k-1}) \\ &= \frac{p(a_2\cdots a_{k-1}a_k)p(a_1a_2\cdots a_{k-1})}{p(a_2\cdots a_{k-1})} \end{aligned} \tag{3}$$

Then we define the composition vector $\vec{V}$ as the relative difference between the predict value and the real value

$$v_i(a_1a_2\cdots a_k) \equiv \frac{p(a_1a_2\cdots a_k) - \tilde{p}(a_1a_2\cdots a_k)}{\tilde{p}(a_1a_2\cdots a_k)}, \tag{4}$$

And $v_i(a_1a_2\cdots a_k) = 0$ if $\tilde{p}(a_1a_2\cdots a_k) = 0$.

2. For a project with N genomes, the dissimilarity matrix is a $N \times N$ matrix with its diagonal equal 0. The non-diagonal items is calculated by the cosine of two composition vectors in the classical CVTree algorithm, i.e.

$$d_{ij} = \frac{1}{2}\left(1 - \frac{\vec{V_i} \cdot \vec{V_j}}{|\vec{V_i}||\vec{V_j}|}\right) \tag{5}$$

3. Then infer the phylogenetic tree (Newick Format) based on this dissimilarity matrix by Neighbor-Joint method.

4.2 Two Typical New CVTree Methods

In this version of CVTree, we added two new methods, named by `InterSet` and `InterList`. The `InterSet` method was proposed by Qiang Li. It is one of methods described in the PhD. thesis of Qiang Li. The `InterList` method proposed by Guanghong Zuo. All these methods have the same scheme. That is, there are three steps, calculating composition vector, building dissimilarity matrix, and inferring phylogenic tree. In the follow sections, we describe these two methods briefly.

4.2.1 The InterSet Method

In the `InterSet` method, the genome is represented by the k-mers set in theory. And the dissimilarity between genomes was defined based the intersection of two represented sets.

1. In practice, the composition vector is sparse for most of k, and only the existed k-mers will be recorded in cvfile as key-value items. Thus the set of all keys of k-mers counts is the k-mers set. Considering of reusing of data, we use the counts of the k-mers as the composition vector of `InterSet` method. That is, the value of the composition vector for every dimension (k-mers) is:

$$v_i = f(a_1 a_2 \cdots a_k) \tag{6}$$

2. The dissimilarity between two genomes is calculated by

$$d_{ij} = 1 - \frac{|Set(V_i) \cap Set(V_j)|}{\sqrt{|Set(V_i)| \cdot |Set(V_j)|}}, \tag{7}$$

   Here $Set(\cdot)$ obtains all keys whose value is non-zero in the composition vector of every genome.

3. The phylogenetic tree (Newick Format) is also inferred by Neighbor Joint method.

4.2.2 The InterList Method

In the `InterList` method, the genome is represented by the vector of the histogram of k-mers. And the dissimilarity between two vectors is defined based on the number of overlapped k-mers of two genomes. The method is inspired by the `InterSet` method. It may be helpful to study the relationship between dissimilarity of CVTree and molecular clock.

1. The counts of the k-mers is used as the composition vectors in `InterList` method. That is, the value of the composition vector for every dimension (k-mers) is:

$$v_i = f(a_1 a_2 \cdots a_k) \tag{8}$$

2. The dissimilarity between two genomes is calculated by

$$d_{jl} = 1.0 - \frac{2.0 \times \sum_i \min(v_{ji}, v_{li})}{\sum_i v_{ji} + \sum_i v_{li}} \tag{9}$$

3. The phylogenetic tree (Newick Format) is also inferred by Neighbor Joint method.

4.3 Available Methods in Detail

The standalone CVTree is an open workflow for the alignment-free phylogenic algorithm. It is very easy to add new method by C++ coding (see detail for how to add method in the subsequent section). Thus there are many optional methods available in the standalone CVTree in every step. And we will add other method in the program in feature.

4.3.1 Available Composition Vector Methods

There are two composition method in standalone CVTree:

- The Hao vector based on the difference between the counted probability and the predicted probability of Markov model (see detail in section 4.1).
- The number of counted. We noted that the count vector is also used as the set of the k-mers in practice, because the composition vector is sparse for most of k, and only the non-zero dimension will be recorded in cvfile.

4.3.2 Available Dissimilarity Methods

According to the major factor of these method, they can be divided in to three groups:

- Based on modeling vector of k-mers
  - Cosine:
$$d_{ij} = \frac{1}{2}\left(1 - \frac{\vec{V_i} \cdot \vec{V_j}}{|\vec{V_i}||\vec{V_j}|}\right)$$
  - Euclidean:
$$d_{jl} = \sqrt{\sum_i (v_{ji} - v_{li})^2}$$
- Based on the number of overlaped k-mers

  - InterList:
$$d_{jl} = 1.0 - \frac{2.0 \times \sum_i \min(v_{ji}, v_{li})}{\sum_i v_{ji} + \sum_i v_{li}}$$
  - Min2Max:
$$d_{jl} = 1.0 - \frac{\sum_i \min(v_{ji}, v_{li})}{\sum_i \max(v_{ji}, v_{li})}$$

- based on the intersection of the sets of k-mers
  - InterSet:
$$d_{ij} = 1.0 - \frac{|Set(V_i) \cap Set(V_j)|}{\sqrt{|Set(V_i)| \cdot |Set(V_j)|}}$$
  - Jaccard:
$$d_{ij} = 1.0 - \frac{|Set(V_i) \cap Set(V_j)|}{|Set(V_i) \cup Set(V_j)|}$$
  - Dice:
$$d_{ij} = 1.0 - \frac{2 \times |Set(V_i) \cap Set(V_j)|}{|Set(V_i)| + |Set(V_j)|}$$

4.3.3 Available Tree Methods

Presently, only the neighbor-joint (NJ) tree method is included in the CVTree standalone package.

5 Design Pattern of CVTree

The standalone CVTree is coded by C++ language, and designed in an object-oriented model. The main programs of this version is parallel, implemented by OpenMP. Figure 3 shows the basic design pattern of CVTree. There are three steps to obtain the phylogenic tree based on the genomes by CVTree, i.e. calculating composition vector form genome, building dissimilarity matrix from composition vectors, and inferring phylogenic tree from dissimilarity matrix. Thus we designed three interfaces (C++ virtual classes), i.e. `CVmeth`, `DistMeth`, and `TreeMath`, to implement these three steps. Despite member functions which perform basic events, there are a factory function, named `create(···)`, and virtual function, named `cv(···)`, `dist(···)`, and `tree(···)` respectively in these three classes. To add a new method in a step:

- derive a class from the base class, and implement the calculating algorithm in the derived class to override the virtual function in the base class.
- add items in the factory function, `create(···)`, of the base class to provide the selection of methods.

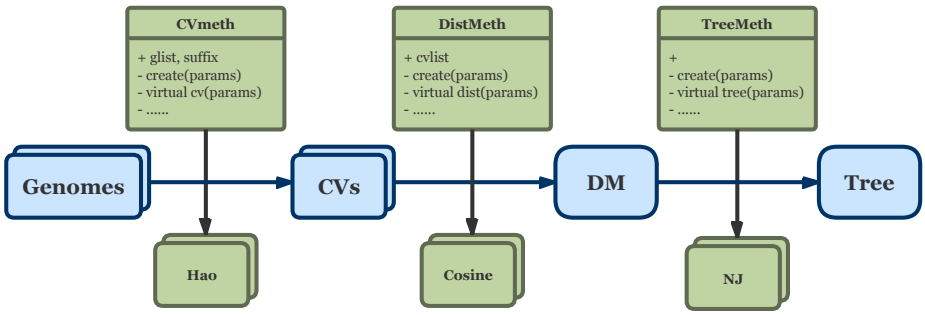

Figure 3: The Design Pattern of CVTree3

6 Version History and Contributors

Since the publication of paper (Qi *et al.*, 2004*b*), many groups had implemented the classical CVTree algorithms. Here we list the major versions which implemented by our group, and the version numbers of the Standalone CVTree were reseted by the version number of the Web Server CVTree as the standalone CVTree have never published:

- Most 0.x Standalone CVTree was written by Lei Gao; Ver. 0.9.6 was written by Ji Qi.
- Web Server CVTree 1.0 was written by Ji Qi, Hong Luo and Bailin Hao
- Standalone CVTree 1.x was written by Zhao Xu
- Web Server CVTree 2.0 was written by Zhao Xu and Bailin Hao
- Standalone CVTree 2.x was written by Guanghong Zuo
- Web Server CVTree 3.0 was written by Guanghong Zuo and Bailin Hao
- Standalone CVTree 3.x was written by Guanghong Zuo
- Web Server CVTree 4.x was written by Guanghong Zuo and Bailin Hao

7 Citing CVTree in a Publication

Please cite:

1. Guanghong Zuo (2020) CVTree: An Alignment-free Phylogeny and Taxonomy Tool Kit based on Composition Vectors of Genomes. *Genomics Proteomics & Bioinformatics,* `in submission`.
2. Guanghong Zuo and Bailin Hao (2015) CVTree3 Web Server for Whole-genome-based and Alignment-free Prokaryotic Phylogeny and Taxonomy. *Genomics Proteomics & Bioinformatics,* **13**: 321–331.
3. Ji Qi, Bin Wang, Bailin Hao (2004), Whole proteome prokaryote phylogeny without sequence alignment: a K-string composition approach. *Journal of Molecular Evolution*, **58**: 1–11.

References

Qi, J., Luo, H. and Hao, B. (2004*a*) Cvtree: a phylogenetic tree reconstruction tool based on whole genomes. *Nucleic acids research,* **32**, W45–7. PMID: 15215347.

Qi, J., Wang, B. and Hao, B. (2004*b*) Whole proteome prokaryote phylogeny without sequence alignment: a k-string composition approach. *Journal of Molecular Evolution,* **58**, 1–11.

Xu, Z. and Hao, B. (2009) Cvtree update: a newly designed phylogenetic study platform using composition vectors and whole genomes. *Nucleic Acids Research,* **37 Web Server Issue**, W174–W178.

Zuo, G. and Hao, B. (2015) CVTree3 Web Server for Whole-genome-based and Alignment-free Prokaryotic Phylogeny and Taxonomy. *Genomics Proteomics & Bioinformatics,* **13**, 321–331.

Zuo, G. (2020) CVTree: An Alignment-free Phylogeny and Taxonomy Tool kit based on Composition Vectors of Genomes. *Genomics Proteomics & Bioinformatics,* `in submission`.