# vica Documentation

***Release 0.1.4***

**Adam Rivers**

**Nov 06, 2018**

# Contents:

CHAPTER 1

# Vica: Software to identify highly divergent DNA and RNA viruses and phages in microbiomes

## 1.1 Authors


- Adam R. Rivers, US Department of Agriculture, Agricultural Research Service
- Qingpeng Zhang, US Department of Energy, Joint Genome Institute
- Susannah Tringe, US Department of Energy, Joint Genome Institute


## 1.2 Introduction

Vica is designed to identify highly divergent viruses and phage representing new families or orders in assembled metagenomic and metatranscriptomic data. Vica does this by combining information from across the spectrum of composition to homology. The current version of Vica uses three feature sets (5-mers, codon usage in all three frames, and minhash sketches from long kmers (k=24,31). The classifier uses a jointly trained deep neural network and logistic model implemented in Tensorflow. The software is designed to identify both DNA and RNA viruses and phage in metagenomes and metatranscriptomes.

## 1.3 Models

The current leases does not include trained models but we will be adding them in the future to allow for the rapid identification of viruses without model training.

## 1.4 Usage

This package can classify assembled data and train new classification models. Most users will only use the classification functionality in Vica. We will provide trained models for classifying contigs in future releases. classification can be easily invoked with the command:

```
vica classify --infile contigs.fasta --out classifications.txt --modeldir modeldir
```

The package also has a suite of tools to prepare data, train and evaluate new classification models. Many of the workflows for doing this can be evoked with the same sub-command interface:

```
vica split
vica get_features
vica train
vica evaluate
```

For details see the Tutorial.

## 1.5 Requirements

The package relies on a number of python dependencies that are resolved when the package is installed with PIP.

The non-python dependencies are:

- Bbtools > v37.75- https://jgi.doe.gov/data-and-tools/bbtools/
- Prodigal > v2.6.3 - https://github.com/hyattpd/Prodigal
- GNU Coreutils - http://www.gnu.org/software/coreutils/coreutils.html

## 1.6 Documentation

Documentation for the package is at http://vica.readthedocs.io/en/latest/

## 1.7 Package availability

- PyPi: https://pypi.python.org/pypi/vica
- Github: https://github.com/USDA-ARS-GBRU/vica

CHAPTER 2

# Tutorial for using and training vica

## 2.1 Installation

Instructions for Genepool.nersc.gov (more to come):

```
module unload python
module load python/3.6-anaconda_4.3.0
module load bbtools
module load pigz
module load prodigal
conda create -n vicaenv python=3.6
source activate vicaenv
conda config --add channels bioconda
conda install pyfaidx
conda install khmer
git clone https://github.com/USDA-ARS-GBRU/vica.git
cd vica
python setup.py build
pip install -e .
# pip install vica
```

## 2.2 Classifying sequences with vica

To classify contigs using vica no additional data is needed except for the pre-trained model. minhash sketched are sent to minhash server to avoid having to download refseq minhash files:

```
vica classify [options]
```

## 2.3 Training models with vica

To train the model without evaluation we use all the available data.

- Download the NCBI taxonomy data and process the data for use by bbtools using the fetchTaxonomy script:

```
vica/pipelines/fetchTaxonomy.sh
```

- Download the Refseq data and process for use by bbtools using using the script:

```
vica/pipelines/fetchRefSeq.sh
```

- If you want to add additional fastas please format them with a header in the format:

```
tid|[taxid number]|[any identifier]| optional info...
```

- The taxid does not need to be at the species level but it must be equal to or lower than the classification group to be included. For example, If you have sequences you want to include in the group bacteria (taxid 2) you can use a family or genus level taxid.
- Shred the data without splitting using the command:

```
vica shred [options]
```

- Extract the features for each fasta using:

```
vica get_features [options]
```

- Feature selection is embarrassingly parallel. If you are working on an HPC system with a job scheduler you can split your files up and submit them in parallel. At the end, all of the TFrecord feature files can be combined into a single file with the command

```
cat *.tfrecord > all.tfrecord
```

- train the model with:

```
vica train [options]
```

- During training you can monitor the process using tensorflow's tensorboard:

```
tensorboard --logdir [directory]
```

## 2.4 Training models and and evaluating performance with vica

The process of training and evaluating a model is the most complex because data for evaluation (the test data) must be split from the data for training the model to get an accurate assessment of performance. In many Machine learning applications this is a trivial matter of random data selection, but because our data are taxonomically structured and some taxa are overrepresented, a simple random split is insufficient. This means that we need to split out test and training data at the level of taxonomic novelty we are hoping to discover.

For the most accurate assessment of the classifier performance we also must exclude test data from and databases used to create features. We currently use Three feature sets: Codon usage and 5mers (which do not contain external databases) and the phylum level taxonomic assignment from minhash sketches, which uses an external database. Codon usage and 5mer feature sets help the model to generalize well and are used in the deep neural network portion of the classifier. The purpose of the minhash feature set is quite different. It serves the purpose of improving classifier

precision by making sure that known non-viral taxa do not enter the viral bin. This feature set is incorporated using a logistic regression model.

All of this information from the logistic regression and the DNN are combined by jointly training both models. Then, "the wide component and deep component are combined using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training." (Cheng et al., 2015). This means that the predictive ability of the minhash features must be realistic or else too much weight could be given to these features. For that reason we leave the test taxa out of the minhash database used to generate minhash features.

The overall process for training from scratch using the Refseq dataset is this:

- Download the NCBI taxonomy data and process the data for use by bbtools using the script:

```
vica/pipelines/fetchTaxonomy.sh
```

- Download the Refseq data and process for use by bbtools using using the script:

```
vica/pipelines/fetchRefSeq.sh
```

- Split the test and train data, and fragment it into fragments of the selected length using the command:

```
vica split_and_shred [options]
```

  Currently we use 4 classes (Viruses, bacteria, archaea, eukaryotes). This will create a directory with test and train folders. Each folder will contain 1 fasta file per class, and a file contain the taxids to be excluded from RefSeq.

- Extract the features for each fasta using:

```
vica get_features [options]
```

  The feature selection is embarrassingly parallel if you are working on an HPC system with a job scheduler you can split your files up and submit them in parallel. at the end all of the TFrecords files can be combined into a single file with the command

```
cat *.tfrecord > all.tfrecord
```

- train the model with:

```
vica train [options]
```

  During training you can monitor the process using tensorflow's tensorboard:

```
tensorboard --logdir [directory]
```

- evaluate the model with:

```
vica evaluate [options]
```

CHAPTER 3

# The complete guide to training and evaluating a vica model

CHAPTER 4

# vica

## 4.1 vica package

### 4.1.1 Submodules

### 4.1.2 vica.get_features module

### 4.1.3 vica.khmer_features module

### 4.1.4 vica.minhash module

### 4.1.5 vica.prodigal module

### 4.1.6 vica.split_shred module

### 4.1.7 vica.tfrecord_maker module

### 4.1.8 vica.train_eval module

### 4.1.9 vica.vica_cli module

### 4.1.10 Module contents

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search