**TAXONOMIC CLASSIFICATION OF TJ RIVER SAMPLE USING VEBA 2.0**

**AIM:**

To identify and classify taxonomy for the TJ River Sample.

**INTRODUCTION:**

The Viral Eukaryotic Bacterial Archaeal (VEBA) is a freely available open-source software suite (<https://github.com/jolespin/veba>) used for end-to-end metagenomics/transcriptomics methodologies beyond the prokaryome to support the eukaryome and virome. Since the metagenomes are obtained from TJ River, I wanted to focus on the prokaryotic taxonomy classification.

**SAMPLE SOURCE:**

I am using **Illumina short reads**: ERR11181250\_1.fastq and ERR11181250\_2.fastq

**Link to the file**: [https://www.ncbi.nlm.nih.gov/sra/?term=ERR11181250](file:///Users/nive/Documents/BIOL-668_AdvBiolDataAnal/PULIADI_SUBRAMANIAN_668_FINAL_PROJECT/PULIADI_SUBRAMANIAN_668_FINAL_PROJECT_VEBA2.0.docx)

**DOCKER INSTALLATION:**

<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>

I performed taxonomic classification using VEBA Docker. I did Docker Installation on BLARNEY.

**STEPS:**

Install using the apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

**1.Set up Docker's apt repository.**

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

**2.Install the Docker packages.**

To install the latest version, run:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

**3.Verify that the Docker Engine installation is successful by running the hello-world image.**

sudo docker run hello-world

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine.

**VEBA DATABASES DOWNLOAD:**

**VIA CONDA:**

### I did VEBA 2.0 installation via Conda.

Please download [miniconda distribution](https://docs.conda.io/projects/miniconda/en/latest/) (what I use).

Currently, **Conda environments for VEBA are ONLY configured for Linux** and, due to the large databases, this software was designed to be used via HPC.

**There are 3 steps to install VEBA:**

* Download repository from GitHub
* Install conda environments
* Download/configure databases

**0. Clean up your conda installation [Optional, but highly recommended]**

The VEBA installation is going to configure some conda environments for you and some of them have quite a bit of packages. To minimize the likelihood of [weird errors](https://forum.qiime2.org/t/valueerror-unsupported-format-character-t-0x54-at-index-3312-when-creating-environment-from-environment-file/25237), it's recommended to do the following:

* Use this as your [~/.condarc](https://conda.io/projects/conda/en/latest/user-guide/configuration/use-condarc.html). If you're not familiar with the .condarc file, then you probably don't have one configured. You can use an editor like [nano](https://anaconda.org/conda-forge/nano) (which is what I use), [vim](https://anaconda.org/conda-forge/vim), or [emacs](https://anaconda.org/conda-forge/emacs) to copy/paste the following into [~/.condarc](https://github.com/jolespin/veba/blob/main/install/condarc).

channel\_priority: flexible

channels:

- conda-forge

- bioconda

- jolespin

- defaults

- qiime2

report\_errors: true

Make sure your conda is [initialized](https://docs.conda.io/projects/conda/en/latest/commands/init.html). I use bash for my initialization.

conda init bash

Clean up your conda environment with the following command.

conda clean --all -y

Update your conda.

conda update -n base --all -y

Install and update [mamba](https://mamba.readthedocs.io/en/latest/installation.html).

conda install -c conda-forge mamba -y

conda update mamba -y

**1. Download repository**

**VERSION="2.1.0"**

**wget https://github.com/jolespin/veba/archive/refs/tags/v${VERSION}.tar.gz**

**tar -xvf v${VERSION}.tar.gz && mv veba-${VERSION} veba**

**# Update the permissions**

**chmod 775 veba/bin/\*.py**

**chmod 775 veba/bin/scripts/\***

**chmod 775 veba/install/\*.sh**

**# Go into the install directory**

**cd veba/install**

**2. Install VEBA environments**

**Recommended resource allocatation:** 4 hours with 16 GB memory (include extra time for variable I/O speed for various hosts)

The update from CheckM1 -> CheckM2 and installation of antiSMASH require more memory and may require grid access if head node is limited.

bash install.sh

**3. Activate the database conda environment, download, and configure databases**

**Recommended resource allocation:** 48 GB memory (time is dependent on I/O of database repositories)

The databases are installed with a series scripts to allow for custom installations and database builds. The easiest way to download and configure the databases is to just run main: veba/install/download\_databases.sh which calls the 4 database scripts and the environment variable script.

**Total size is ~270 GB** but if you have certain databases installed already then you can just symlink them so the VEBA\_DATABASE path has the correct structure.

conda activate VEBA-database\_env

bash download\_databases.sh /path/to/veba\_database/

Now, you should have the following environments:

VEBA-annotate\_env

VEBA-assembly\_env

VEBA-binning-eukaryotic\_env

VEBA-binning-prokaryotic\_env

VEBA-binning-viral\_env

VEBA-biosynthetic\_env

VEBA-classify\_env

VEBA-cluster\_env

VEBA-database\_env

VEBA-mapping\_env

VEBA-phylogeny\_env

VEBA-preprocess\_env

VEBA-profile\_env

**DOCKER RUN:**

I followed the following link to Install Docker, Pull Docker Images, Run Docker Container, Run VEBA modules within Docker Container, Mount working directory to an internal volume within Docker container, and Mount separate input and output directories to internal volumes within the Docker container to get the results.

[**https://github.com/jolespin/veba/blob/main/walkthroughs/docs/adapting\_commands\_for\_docker.md**](https://github.com/jolespin/veba/blob/main/walkthroughs/docs/adapting_commands_for_docker.md)

**METHODS:**

**MY COMMANDS FOR EVERY MODULE:**

**Preprocess:**

* **Decontaminate Metagenomic Reads**

# Set a variable for the docker image with the correct version

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_preprocess:${VERSION}"

# Pull the image from the official repository

docker image pull "${DOCKER\_IMAGE}"

# Check to make sure you can pull up the help docs

docker run --name veba-preprocess --rm -it "${DOCKER\_IMAGE}" preprocess.py -h

# Local Directories

LOCAL\_WORKING\_DIRECTORY=~/volumes #make a dir called volumes

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/volumes/container\_output # make a dir called container output in volumes

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

# Container Directories

CONTAINER\_INPUT\_DIRECTORY=/volumes/input

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output

# Parameters

ID=ERR11181250

R1=Fastq/${ID}\_1.fastq.gz #make a dir called Fastq and Read1 files are inside this dir

R2=Fastq/${ID}\_2.fastq.gz #Read2 files are inside this Fastq dir

NAME=veba-preprocess\_\_${ID}

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/preprocess/ #create a dir called veba\_output

CMD="preprocess.py -1 ${CONTAINER\_INPUT\_DIRECTORY}/${R1} -2 ${CONTAINER\_INPUT\_DIRECTORY}/${R2} -n ${ID} -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} -p=-1"

#Docker run

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

"${DOCKER\_IMAGE}" \

${CMD}

**# The word “volumes” is confusing in terms of Docker. So, let’s make a small change.**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_preprocess:${VERSION}"

# Change the directory name

mv ~/volumes/ ~/veba\_docker\_analysis/

# Also, let’s change the the ~/veba\_docker\_analysis/input/ directory name as well to minimize confusion between volume mount points and local file system directories

mv ~/veba\_docker\_analysis/input/ ~/veba\_docker\_analysis/sequencing\_files/

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/sequencing\_files/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Make sure your fastq files are in there and are not empty files

ls -lh ${LOCAL\_WORKING\_DIRECTORY}/Fastq/

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

# Make sure the permissions are set so docker can write to it (will probably require password)

sudo chmod -R 777 ~/veba\_docker\_analysis/

# Set the container mount points (these only exist on the container, but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

# Specify the args

ID="ERR11181250"

R1=Fastq/${ID}\_1.fastq.gz

R2=Fastq/${ID}\_2.fastq.gz

NAME=veba-preprocess\_\_${ID}

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/preprocess/

# Build the command

CMD="preprocess.py -1 ${CONTAINER\_INPUT\_DIRECTORY}/${R1} -2 ${CONTAINER\_INPUT\_DIRECTORY}/${R2} -n ${ID} -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} -p=-1"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

"${DOCKER\_IMAGE}" \

${CMD}

In the end, these commands should have mounted the Fastq/ directory in `~/veba\_docker\_analysis/sequencing\_files/` with the corresponding container directory here: `/volumes/input/` then it mounts the local output directory `~/veba\_docker\_analysis/container\_output/` to the container directory here: /volumes/output/ where it writes the results to a directory called `veba\_output/preprocess/`. So on the container, it’s writing to `/volumes/output/veba\_output/preprocess/` but it’s mirrored in `~/veba\_docker\_analysis/container\_output/veba\_output/preprocess/` and will be saved there when container is finished.

**#Assembly:**

* **Assemble reads, map reads to assembly, calculate assembly statistics, and index the assembly**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_assembly:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/preprocess/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

# Specify the args

ID="ERR11181250"

R1=${ID}/output/trimmed\_1.fastq.gz

R2=${ID}/output/trimmed\_2.fastq.gz

NAME=veba-assembly\_\_${ID}

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/assembly/

# Build the command

CMD="assembly.py -1 ${CONTAINER\_INPUT\_DIRECTORY}/${R1} -2 ${CONTAINER\_INPUT\_DIRECTORY}/${R2} -n ${ID} -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} --program metaspades.py -p=4"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

"${DOCKER\_IMAGE}" \

${CMD}

**# Recover viruses from metagenomic assemblies:**

* **Removing viral contigs first should make the prokaryotic binning more refined since we are removing noise**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_binning-viral:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/assembly/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

LOCAL\_DATABASE\_DIRECTORY=/dev/shm/veba/veba\_database-v6/ #${VEBA\_DATABASE} # /path/to/VEBA\_DATABASE/

LOCAL\_DATABASE\_DIRECTORY=$(realpath -m ${LOCAL\_DATABASE\_DIRECTORY})

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

CONTAINER\_DATABASE\_DIRECTORY=/volumes/database/

# Specify the args

ID="ERR11181250"

FASTA=${ID}/output/scaffolds.fasta

BAM=${ID}/output/mapped.sorted.bam

NAME=veba\_binning-viral\_\_${ID}

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/binning/viral/

# Build the command

CMD="binning-viral.py -f ${CONTAINER\_INPUT\_DIRECTORY}/${FASTA} -b ${CONTAINER\_INPUT\_DIRECTORY}/${BAM} -n ${ID} -p=64 -m 1500 -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} --veba\_database ${CONTAINER\_DATABASE\_DIRECTORY}"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:ro" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_DATABASE\_DIRECTORY}:${CONTAINER\_DATABASE\_DIRECTORY}:ro" \

"${DOCKER\_IMAGE}" \

${CMD}

**# Recover prokaryotic from metagenomic assemblies:**

* **We are going to perform iterative prokaryotic binning**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_binning-prokaryotic:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

LOCAL\_DATABASE\_DIRECTORY=/dev/shm/veba/veba\_database-v6/ #${VEBA\_DATABASE} # /path/to/VEBA\_DATABASE/

LOCAL\_DATABASE\_DIRECTORY=$(realpath -m ${LOCAL\_DATABASE\_DIRECTORY})

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

CONTAINER\_DATABASE\_DIRECTORY=/volumes/database/

# Specify the args

ID="ERR11181250"

FASTA=binning/viral/${ID}/output/unbinned.fasta

BAM=assembly/${ID}/output/mapped.sorted.bam

NAME=veba\_binning-prokaryotic\_\_${ID}

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/binning/prokaryotic/

# Build the command

CMD="binning-prokaryotic.py -f ${CONTAINER\_INPUT\_DIRECTORY}/${FASTA} -b ${CONTAINER\_INPUT\_DIRECTORY}/${BAM} -n ${ID} -p=64 -m 1500 -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} --veba\_database ${CONTAINER\_DATABASE\_DIRECTORY}"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:ro" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_DATABASE\_DIRECTORY}:${CONTAINER\_DATABASE\_DIRECTORY}:ro" \

"${DOCKER\_IMAGE}" \

${CMD}

**# Cluster**

**# We need to build the genomes\_table.tsv file and it has to be done interactively:**

# IF YOU WANT TO RUN INTERACTIVELY ------

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_cluster:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/input/

docker run \

--name "${NAME}" \

--rm \

-it \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

"${DOCKER\_IMAGE}" \

bash

**# Once interactive**

mkdir -p /volumes/output/misc/

compile\_genomes\_table.py -i /volumes/input/binning/ > /volumes/input/misc/genomes\_table.tsv

**# Then exit out of docker container w/ `exit`**

**#Cluster:**

* **We are going to generate some counts tables and we want a single set of features to compare across all samples. To achieve this, we are going to cluster the genomes into species-level clusters (SLC).**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_cluster:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

# Specify the args

GENOMES\_TABLE=misc/genomes\_table.tsv

NAME=veba\_cluster

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/cluster/

# Build the command

CMD="cluster.py -i ${CONTAINER\_INPUT\_DIRECTORY}/misc/genomes\_table.tsv -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} -p=-1"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:ro" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

"${DOCKER\_IMAGE}" \

${CMD}

**# Classify Viral Genomes**

* **Viral classification is performed using geNomad. Classification can be performed using the intermediate binning results which is much quicker.**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_classify:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

LOCAL\_DATABASE\_DIRECTORY=/dev/shm/veba/veba\_database-v6/ #${VEBA\_DATABASE} # /path/to/VEBA\_DATABASE/

LOCAL\_DATABASE\_DIRECTORY=$(realpath -m ${LOCAL\_DATABASE\_DIRECTORY})

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

CONTAINER\_DATABASE\_DIRECTORY=/volumes/database/

# Specify the args

BINNING\_DIRECTORY=binning/viral/

CLUSTERS=cluster/output/global/mags\_to\_slcs.tsv

NAME=veba\_classify

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/classify/viral/

# Build the command

CMD="classify-viral.py -i ${CONTAINER\_INPUT\_DIRECTORY}/${BINNING\_DIRECTORY} -c ${CONTAINER\_INPUT\_DIRECTORY}/${CLUSTERS} -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} --veba\_database ${CONTAINER\_DATABASE\_DIRECTORY}"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:ro" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_DATABASE\_DIRECTORY}:${CONTAINER\_DATABASE\_DIRECTORY}:ro" \

"${DOCKER\_IMAGE}" \

${CMD}

**# Classify Prokaryotic**

* **Prokaryotic classification is performed using GTDB-Tk. Classification can be performed using the intermediate binning results which is easier.**

# Docker image

VERSION="2.0.0"

DOCKER\_IMAGE="jolespin/veba\_classify:${VERSION}"

LOCAL\_WORKING\_DIRECTORY=~/veba\_docker\_analysis/container\_output/veba\_output/

LOCAL\_WORKING\_DIRECTORY=$(realpath -m "${LOCAL\_WORKING\_DIRECTORY}")

# Output directory

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=~/veba\_docker\_analysis/container\_output/

LOCAL\_OUTPUT\_PARENT\_DIRECTORY=$(realpath -m "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}")

LOCAL\_DATABASE\_DIRECTORY=/dev/shm/veba/veba\_database-v6/ #${VEBA\_DATABASE} # /path/to/VEBA\_DATABASE/

LOCAL\_DATABASE\_DIRECTORY=$(realpath -m ${LOCAL\_DATABASE\_DIRECTORY})

# Set the container mount points (these only exist on the container but you are mounting into these specific directories from your local file system)

CONTAINER\_INPUT\_DIRECTORY=/volumes/input/

CONTAINER\_OUTPUT\_DIRECTORY=/volumes/output/

CONTAINER\_DATABASE\_DIRECTORY=/volumes/database/

# Specify the args

BINNING\_DIRECTORY=binning/prokaryotic/

CLUSTERS=cluster/output/global/mags\_to\_slcs.tsv

NAME=veba\_classify-prokaryotic

RELATIVE\_OUTPUT\_DIRECTORY=veba\_output/classify/prokaryotic/

# Build the command

CMD="classify-prokaryotic.py -i ${CONTAINER\_INPUT\_DIRECTORY}/${BINNING\_DIRECTORY} -c ${CONTAINER\_INPUT\_DIRECTORY}/${CLUSTERS} -o ${CONTAINER\_OUTPUT\_DIRECTORY}/${RELATIVE\_OUTPUT\_DIRECTORY} -p=-1 --veba\_database ${CONTAINER\_DATABASE\_DIRECTORY}"

# Run docker

docker run \

--name "${NAME}" \

--rm \

--volume "${LOCAL\_WORKING\_DIRECTORY}:${CONTAINER\_INPUT\_DIRECTORY}:ro" \

--volume "${LOCAL\_OUTPUT\_PARENT\_DIRECTORY}:${CONTAINER\_OUTPUT\_DIRECTORY}:rw" \

--volume "${LOCAL\_DATABASE\_DIRECTORY}:${CONTAINER\_DATABASE\_DIRECTORY}:ro" \

"${DOCKER\_IMAGE}" \

${CMD}

**RESULTS:**

**Taxonomy Classification from Prokaryotes:**

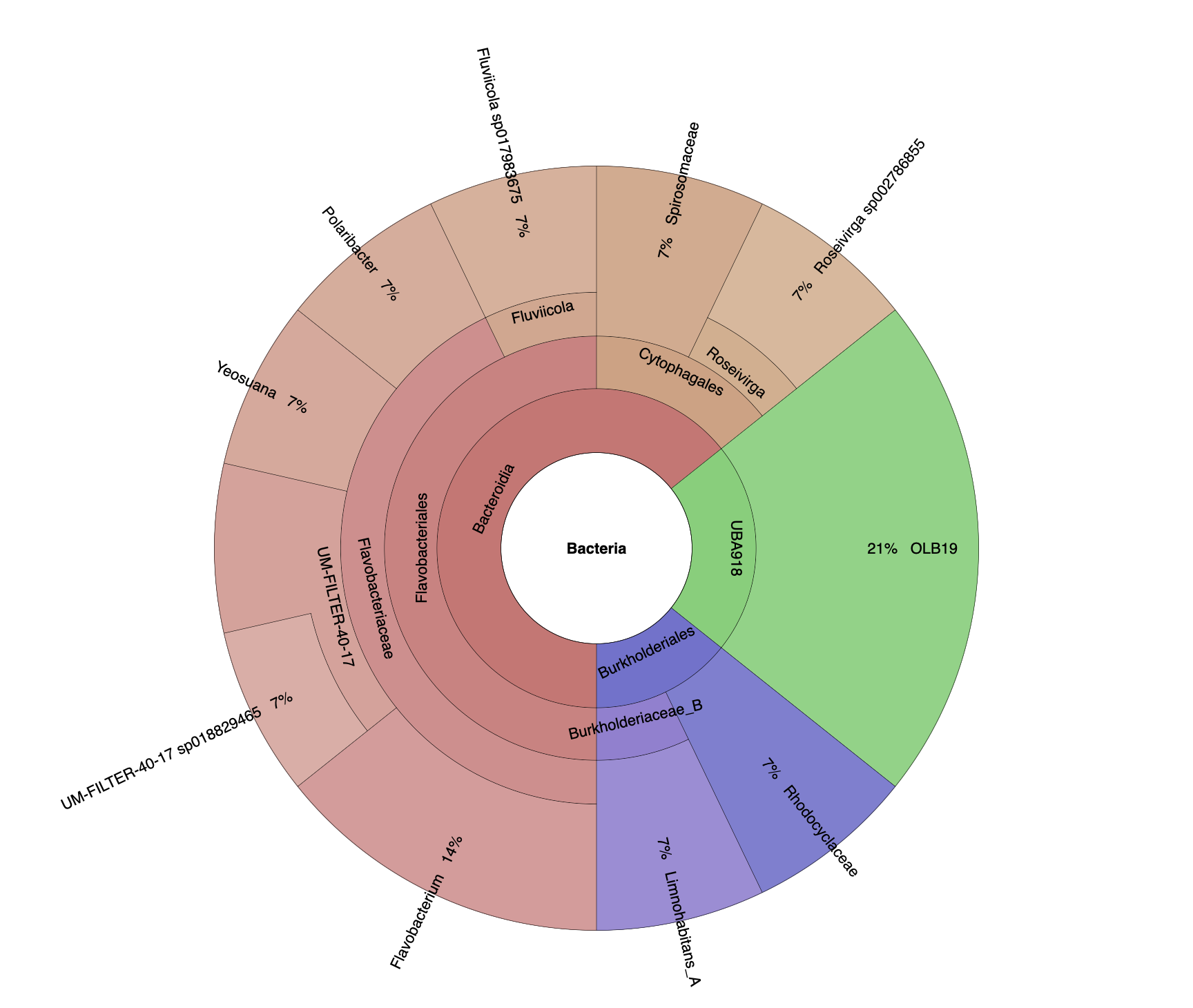
****

**Taxonomy Classification from Virus:**

****

**KRONA: A VISUALIZATION TOOL**

**FOR PROKARYOTES:**

****

**DISCUSSION:**

We got 14 bins for prokaryotic metagenomes obtained from TJ River sample. Most of them are classified at the genus level except few are classified at the species level. For viral metagenomes, we got 10 bins. Also, we can visualize the metagenomic classification using a tool called KRONA. We could visualize the taxonomic composition of the metagenomes obtained from TJ River sample. OLB19 has 21% composition, the highest one in this sample. This could be studied by analyzing the genomic content or conduct further investigations to understand the function or ecological role of organisms classified within this taxonomic group. Additionally, consulting relevant literature or performing comparative genomic analyses with closely related taxa could provide insights into its potential function.