Visualizing the bins with anvi'o

NOTE: Anvi'o has a *lot* of tutorials that are more comprehensive than this one and you can find them at the anvi'o website. I encourage you to work through those tutorials.

NOTE: anvi'o is installed on the AWS images (version > 8) in a virtual environment. To run it, you need to enter the command:

```
source /usr/local/virtual-envs/anvi-5.2/bin/activate
```

this will activate the anvi'o system, and you should be good to go. If at anytime you want to leave the anvi'o system, you can use

```
deactivate
```

to return to your previous settings.

To test your anvi'o installation, you can use

```
anvi-self-test --suite mini
```

which will create a test dataset and start the server listening.

Accessing the AWS instance

By default the AWS instance does not allow access via `http` or `https`, the two protocols that we use on websites.

You need to create a new set of security rules and then apply them to your image. For more details, see how to create security rules

---

Getting started with anvi'o

This is a visualization platform to see how the bins look. This platform can be installed only on Mac and LINUX environment. They don't have a version for windows, however everyone seems to get it to work using Linux under windows. Follow the instructions on the anvi'o installation website to install it.

We're following the Metagenomics workflow that is described in the v2 tutorial for these steps.

To start the workflow you will need the following two input files

- **contigs.fasta**: The assembled contigs from our spades.py assembly folder.

- **Several BAM files** of our alignments of our reads to those contigs. See the section below on making the `.bam` files.

Making a `.bam` file from our bowtie2 mapping.

For anvi'o we want to have one alignment for every reads file, and so we are going to repeat the `bowtie` mapping that we did earlier. However, previously we did it with a single reads file, the *AllReads*, but this time we are going to do it with each reads file independently.

First, we start by mapping our reads:

```
bowtie2 -f -x AlgaeBowtie -U Algae_11.renum.fna > Algae_11.renum.sam
bowtie2 -f -x AlgaeBowtie -U Algae_12.renum.fna > Algae_12.renum.sam
bowtie2 -f -x AlgaeBowtie -U Algae_13.renum.fna > Algae_13.renum.sam
bowtie2 -f -x AlgaeBowtie -U Algae_14.renum.fna > Algae_14.renum.sam
```

This creates a series of files that ends with `.sam`. This is a plain text format of a file which we can easily read, but computers find hard to read. Computers find it easier to read `.bam` files, which are a binary format, and so we can convert those `.sam` files to `.bam` files. Note that here, the `samtools` program has a weird glitch, which is that it will automatically add `.bam` to the end of the file name. Therefore, we don't have to do that! If you get a file that ends `.bam.bam`, you can always use the `mv` command to move it to a file that ends just `.bam`

```
samtools view -bS  Algae_11.renum.sam  | samtools sort - Algae_11.renum
samtools view -bS  Algae_12.renum.sam  | samtools sort - Algae_12.renum
samtools view -bS  Algae_13.renum.sam  | samtools sort - Algae_13.renum
samtools view -bS  Algae_14.renum.sam  | samtools sort - Algae_14.renum
```

Note that this creates four new files called `Algae_11.renum.bam`, `Algae_12.renum.bam`, `Algae_13.renum.bam`, `Algae_14.renum.bam`.

We need to index that file, which means we make a separate file that tells us where the reads are in the binary file. We can either do that using samtools with this command:

```
samtools index AlgaeAllReads.bowtie.bam AlgaeAllReads.bowtie.bam.bai
```

or we can do this using anvi'o (see below). (Some of the samtools implementations crash when you try and do this step. If that is the case, just use anvi'o, the result is the same.)

Creating the anvi'o contigs database:

First, we start by creating a database of the contigs for our data, and the open reading frames for that data:

```
anvi-gen-contigs-database -f AlgaeAssembly/contigs.fasta -o algae.db
```

When we run this command, anvi'o does several things:

- Computes the k-mer frequencies for each contig (the default is k=4)
- Soft-split contigs longer than 20,000 kbp into smaller ones
- Identifies open reading frames using Prodigal,

Run HMM models against the ORFs we have predicted

Anvi'o will run hidden Markov models against the open reading frames that prodigal predicts. anvi'o uses single copy bacterial genes to identify things on the contigs.

```
anvi-run-hmms -c algae.db
```

Index the `.bam` files

If you did not index the `.bam` file that you created above, anvi'o can do that for you. If you have a `.bai` file for every `.bam` file, you can skip this step.

```
anvi-init-bam Algae_11.renum.bam
anvi-init-bam Algae_12.renum.bam
anvi-init-bam Algae_13.renum.bam
anvi-init-bam Algae_14.renum.bam
```

Profile the data.

Finally, we make a profile of the `.bam` file using anvi'o. This reads the bam file and stores information about each of the contigs and their presence in the different samples.

```
anvi-profile -i Algae_11.renum.bam-sorted.bam -c algae.db
anvi-profile -i Algae_12.renum.bam-sorted.bam -c algae.db
anvi-profile -i Algae_13.renum.bam-sorted.bam -c algae.db
anvi-profile -i Algae_14.renum.bam-sorted.bam -c algae.db
```

Merge the anvi'o outputs into a single location

We are going to use the anvi-merge command with each of our Algae data sets to create a single set of merged data. This also uses the bash ? that we saw earlier which replaces a single character

```
anvi-merge Algae_1?.renum.bam-sorted.bam-ANVIO_PROFILE/PROFILE.db -o AllAlgae -c algae.db
```

Open the anvi'o results

Finally, we open and view the anvi'o results in our web-browser

```
anvi-interactive -p AllAlgae/PROFILE.db -c algae.db
```

You can now open a web browser on your laptop and point it at the IP address of your virtual machine. (See page to see how to get the IP address for your machine). You need to add :8080 to the IP address. In the example above, my IP address is 149.165.157.55 and so the URL that I will open is:

```
http://149.165.157.55:8080
```

Note that anvi'o recommends using Google Chrome for visualization, although other browsers also work.