Setting up a virtual environment and installing Whatshap

Before we run any code, we want to create a virtual environment and ensure that we have all the necessary packages installed. To create a virtual environment, make sure you are in your home directory on Minerva and type the following:

module purge

module load python/3.5.0

virtualenv <name of environment>

To enter your virtual environment:

source <name of environment>/bin/activate

You should now see (name of environment) at the start of each terminal prompting line. For example:

(i\_venv) [seidea02@minerva4 ~]$

To leave your virtual environment, type

deactivate

We will need to load a few more modules from Minerva before installing the necessary packages so exit your virtual environment for now and load the following:

module purge

module load python/3.5.0 py\_packages/3.5 tabix

Now that we have the proper modules loaded, we can enter our virtual environment again and install Whatshap! In your home directory on minerva:

source <name of environment>/bin/activate

pip install --upgrade pip

pip install whatshap

Note: make sure you are not logged into an interactive node for this part, you will need access to the internet to install Whatshap. Once Whatshap is installed, you can log into an interactive node. If you'd like to work in an interactive node on Minerva after completing the above steps, you'll need to reload modules before getting into your virtual environment.

To double check that you have installed Whatshap correctly, you can type “whatshap” and the help menu should show up.

Congratulations! You have successfully installed Whatshap! Now, we will make sure our files are in the right format before inputting them into Whatshap.

Splitting VCF files and fixing problematic BAMs

To properly run the phasing programs, we need to split the VCFs into 22 VCFs by chromosome. To do this, you’ll need to edit the IDs and file locations of the VCFs in the program **chromosome\_split.py** in the **phasing** folder. The script will want to write the split VCF files to a directory specific to each ID so you’ll either have to create these directories or remove this aspect of the script. You’ll also have to module load tabix. Then run:

python3 chromosome\_split.py

It's possible that you will run into issues with your BAM files. The main issue we previously ran into was non-ASCII characters present in the BAM files. To remove them:

module load samtools/1.8

sed 's/[^\\x00-\\x7F]//g' original BAM file > ID\_edit.sam

grep -P -n --color=\"auto\" \"[^\\x00-\\x7F]\" ID\_edit.sam

Note: the last command will produce no output if the non-ASCII characters are removed

You'll then need to sort and index these files before running them through Whatshap. You can run this through **sort\_and\_index.py** in the **phasing** folder. You’ll have to edit the IDs in this script. As long as you name the SAM files ID\_edit.sam and run the script in the same directory as the SAM file, there should be no issues with running this.

Running Whatshap with Pacbio data

You're ready to run Whatshap! Woohoo!

To run Whatshap with the Pacbio data, modify **pacbio\_whatshap.py** in the **phasing** folder to reflect the proper IDs and file locations. It will write the phased VCFs to a folder named “ID\_no\_indels” so you will either need to create these directories or adjust the script.

python3 pacbio\_whatshap.py

You will also want to generate GTF files through Whatshap for later steps. To do this, modify **get\_gtf.py** in the **phasing** folder to reflect the proper IDs. Again, make sure you either create the “ID\_no\_indels” directories or change this aspect of the script.

python3 get\_gtf.py

It is important to make note of what you name your files. If you keep them consistent to what I've listed, you won't have to ensure that the file names match in the code I've written.

Running Whatshap with Illumina data

Now we will go through the steps of running Whatshap with Illumina data. This is very similar to running Whatshap with Pacbio data but with the addition of the --indels flag.

To run Whatshap with the Illumina data, modify i**llumina\_whatshap\_int1.py** and **illumina\_whatshap\_int2.py** in the **phasing** folderto reflect the correct IDs and file locations. I split the IDs into 2 here to run on both interactive 1 and interactive 2 simultaneously just because the Illumina data takes longer than Pacbio.

python3 illumina\_whatshap\_int1.py

python3 illumina\_whatshap\_int2.py

Parent assignment

Now that you have the phased VCFs from Whatshap, it’s time to assign the *de novo* variants to the parents! Yay!

The first thing you’ll have to do is go into **PhasedData.py** in the **phasing** folder and modify a few things.

1. If the mother’s and father’s IDs are not formatted as child\_ID-01 and child\_ID-02, then you’ll most likely need to pass in the parental IDs as parameters into the constructor and modify self.mom and self.dad to reflect that.
2. Adjust self.bed to read in the correct file in the constructor.
3. In create\_vcf\_dictionary, change self.vcf\_dfs to read in the VCF files from the Illumina WGS trios and self.gtf\_dfs to read in the GTF files from the Illumina WGS trios
4. In create\_vcf\_no\_indels, change self.vcf\_dfs to read in the VCF files from the Pacbio data from the Netherlands and self.gtf\_dfs to read in the GTF files from the Pacbio data from the Netherlands
5. In convert\_to\_dataframe, the last line writes the dataframe to a text file. If you’d like to change the file’s location or name, you will have to do that here.

Next, you’ll have to modify the IDs in **get\_ID\_dataframes.py**. You will also have to comment out either the Pacbio command or the Illumina command when you run it to make sure Minerva doesn’t explode. I think you should only run one at a time.

Getting one big huge dataframe

The code for getting one big huge dataframe (which I called phasing\_analysis\_df.txt but you can name whatever you want obviously) is in **analysis.py** in the **phasing\_analysis** folder. These are the few things you’ll have to change:

1. At the very top, you’ll have to change the IDs again. You’ll also have to change the locations of the BED files and the dataframe text files that were outputted from **get\_ID\_dataframes.py**
2. To deal with getting the CpG regions, I went in and manually changed every start and end value in the 10 BED files I was working with because I wanted some mindless work to do at the time. Since you won’t want to do that by hand, you’ll have to write a script to change the all the values in the start column to be “start-1” and all the values in the end column to be “end+1”
   1. Once you get your modified bed files, you’ll have to run getfasta (bedtools getfasta -fi <fasta> -bed <modified\_bed> -fo <output\_filename> -tab)
   2. The location of these getfasta files should then replace mine on line 89.
3. Lastly, you’ll have to adjust the location of the BED files on line 120 to be where yours are

Hooray!! You have one big huge (happy) dataframe with all the information you need!