Fitness Effects of Horizontal Gene Transfer from *Wolbachia* to *Drosophila* and Pipeline development and implementation of phylogeny analysis software

Developers: Olga Better, Kaitlin Klotz, and John Patterson

**Background/Previous Work/Literature Review:** Trending efforts to understand host-infection relationships to better address pathogenic outbreak and containment has lead researchers into the domain of bioinformatics. Using a model such as *Drosophila* infection from the bacterial parasite *Wolbachia,* which causes direct phenotype manipulation through horizontal gene transfer*,* we can develop a basis by which to analyze the host-infection relationship. *Wolbachia* is a parasite that invades *Drosophila* and causes feminization of the offspring, manipulating cellular processes to favor its own transmission (McGraw and O’Neill, 2003). We expect this phenotype to exert either a positive or negative effect on *Drosophila* fitness.

In addition to male feminization, infection with *Wolbachia* causes a phenomenon called cytoplasmic incompatibility, which results in gametes in both sexes of *Drosophila* being unable to produce viable offspring between subspecies (Zabalou et. al., 2004). While the mechanisms of cytoplasmic incompatibility (CI) are not well understood, the Marshall lab found that mitochondrial elements can be inserted into host species by parasites such as *Wolbachia* (Marshall, 2004). These parasitically-induced phenotypes may lead to a decline in successful reproduction between subspecies of arthropods and an overall decline in speciation in a phenomenon known as reproductive isolation (Marshall, 2004).

Previously, researchers in the Rogers Lab at UNCC identified DNA that was transferred from the parasite *Wolbachia* to *Drosophila* fruit flies by isolating sequenced read-pairs that showed one read on the *Wolbachia* chromosome, and one read on the *Drosophila* chromosome. The Rogers research group created genome sequence files through Illumina sequencing. Genuine mutations were supported by multiple pairs of abnormally mapping read-pairs. Portions of the genome containing tandem gene duplications (copies of the same gene next to each other in the genome) carried an elevated probability for further mutation, i.e. insertion of the *Wolbachia* DNA, to create novel genes (Rogers et. al, 2014). The procedural programming scripts that evaluated the *Wolbachia* and *Drosophila* genomes for their similarity to each other output a list of *Drosophila* genes as entries in the FlyBase *Drosophila* genome database (FlyBase, 2018) that could be of interest to the investigator. New genes formed via Horizontal Gene Transfer (HGT) and were detected by Reverse Transcriptase-Polymerase Chain Reaction (RT-PCR), RNA Seq, and fluorescent in situ hybridization (FISH) after consideration of the gene list produced by the scripts. Careful evaluation of the gene expression patterns of newly transferred *Wolbachia* DNA was implemented to identify cases of new gene formation. A minimum 20 strain subset of the Rogers Lab’s 150 different *Drosophila* strains were used to study why some strains of *Drosophila* thrived after *Wolbachia* HGT and others did not. Phenotypic assays were performed on stressed (temperature shocked, starved, and exposed to chemical mutagens) *Drosophila* to test the effects of *Wolbachia* to *Drosophila* HGT and new gene formation. Finally, the Rogers lab explored the evolutionary dynamics of these novel genetic mutations by examining the phylogeny of HGT segments in *Drosophila*.

**Significance:** The phenotypic changes conferred to *Drosophila* by HGT of *Wolbachia* genes may not apply solely to *Drosophila*. In addition to fruit flies, *Wolbachia* infects mosquitoes (particularly *Aedes aegypti*). Once infected with *Wolbachia*, mosquitoes mate and infected females transmit the parasite to their offspring. As the parasite feminizes infected males, it causes sterility, however, females cannot tell which males are infected. Unknowingly mating with sterile, infected males, the the disease is transmitted to healthy females. While no offspring are produced from these encounters, the parasite lives within the female until she successfully reproduces and passes *Wolbachia* along to her progeny. Both the male and female progeny are born infected with *Wolbachia*. The male progeny are feminized and rendered sterile while the female progeny transmit the disease to uninfected males in their mating encounters.

While this may not seem significant from the provided information, it provides humans with a potential method of mosquito population control. If *Wolbachia*-infected males are sterile, then any union between an uninfected female and an infected male results in no offspring-only *Wolbachia* infection that will perpetuate in all future mates and offspring. This causes an exponential increase in the *Wolbachia* infected population, which cannot produce offspring due to sterility (Bost, 2016). While human interventions such as screened doors and windows, bed netting, air conditioning and electricity have contributed greatly to the decline of mosquito-borne disease in the developed world, that is not the case everywhere. There are still some regions of the developing world where diseases such as erwinia, malaria, dengue fever, chikungunya and brugia are endemic and public health officials and scientists must exploit nature to reduce the prevalence of mosquito-borne disease (Bost, 2016).

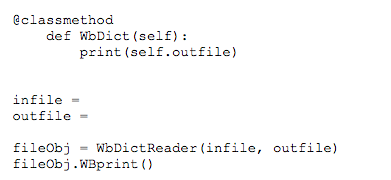
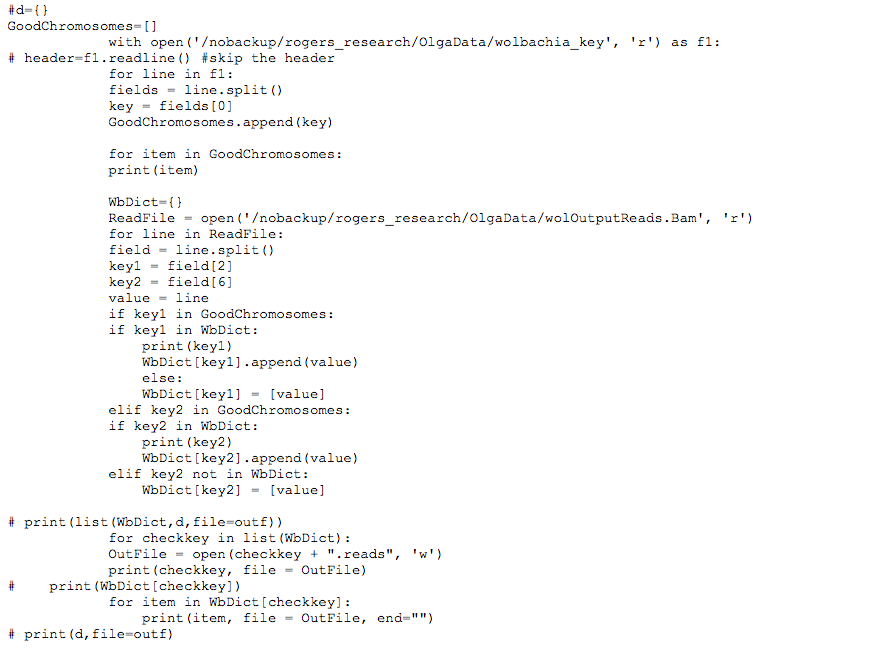
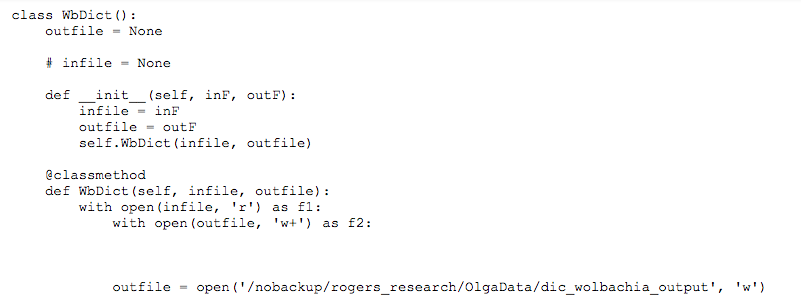
While *Wolbachia* infection of mosquitoes occurs often in nature, scientists can facilitate the large scale infection of male mosquitoes (who will then be sterile) and then release them. The sterile males will mate and infect females without the production of more mosquitoes. When the infected females mate with healthy males, they will further transmit the disease leading to a larger sterile proportion of the population and hence, an eventual decrease in the total number of mosquitoes (Bost, 2016). An understanding of exactly how insect hosts of *Wolbachia* become feminized may be applied to mosquitoes to enhance efficiency of the sterilization process. Intuitively, if there are fewer total mosquitoes, there are fewer opportunities for mosquito-borne disease to be transmitted to humans and livestock.

**Novelty and Contributions:** Previously, the code used to distinguish novel mutations in the *Drosophila* genome as a result of *Wolbachia* HGT existed in four separate Python files developed by Olga Better during her rotation in the Rogers Lab during the Fall 2017 semester. At the beginning of the Spring 2018 semester, permission was obtained from Dr. Rogers to further develop the existing code into a package for use by non-programmers. While the original scripts are relatively simple to understand for users with a basic background in Python programming, they would not mean much to a non-programmer such as a general biologist. The improvements we are making to the existing project are the universalization of the code for application to genera of host/pathogen outside of *Drosophila*/*Wolbachia* and the implementation of a GUI. The GUI will provide an easy-to-use version of the current scripts after conversion to object oriented programming in which biologists can input their genome Illumina sequence files obtained through their own sequencing assays and receive a gene list, which is list of genes that were likely mutated (using FlyBase references). PANTHER (Protein ANalysis THrough Evolutionary Relationships) could then be used to determine what (if any) evolutionary relationships exist between the mutated genes in the output gene list so that they can then focus their studies to target for phenotypic contribution (Mi et al., 2016). These additions/improvements to the current code serve to make genomic techniques more accessible to a wider range of researchers. Biologists will then be able to use their results, a list of novel mutations introduced by HGT, to develop targeted gene knockout experiments and phenotypic assays.

**Methods:** To begin our project, our first task was converting the original inline command scripts from to object oriented code. While the final product of this project will take all four procedural runtime scripts and convert them to object oriented code, for the midterm checkpoint, we have successfully converted the first script shown in Figures 1 and 2. Translation of the new classes created from the previous scripts is performed one at a time rather than all at once, allowing time for debugging and implementation with the GUI code. So, the first script that was converted was wolbachia\_dic script, which can be seen in Figure 1 below.



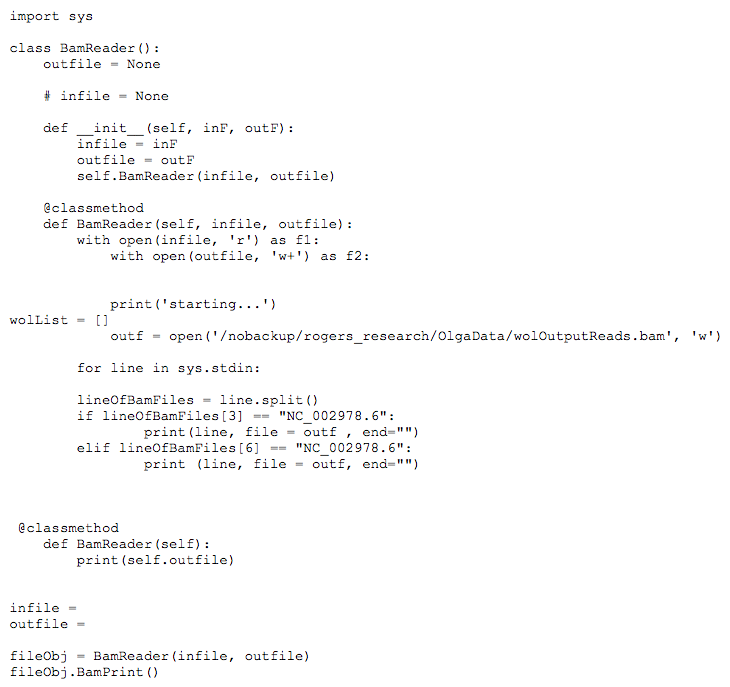
**Figure 1.** The original wolbachia\_dic.py runtime script that established a dictionary of *Wolbachia* genes.



**Figure 2.** The wolbachia\_dic.py runtime script converted into OOP. The class WbDict, which holds WbDict class method achieves the same result as the original script. However, WbDict does so in a way that is more organized.

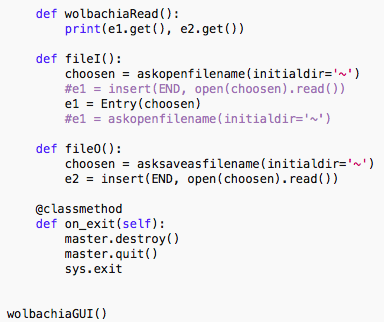
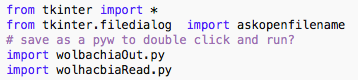
The object oriented code establishes WbDict as the class and initializes the method with an infile and outfile as well as calls the class method WbDict that will be called inside the init method in order to produce the dictionary. The list GoodChromosomes was populated by field [0] of a data file from the Rogers Lab. The dictionary WbDict was then composed of entries that have two possible keys and a value. The keys were taken from fields [2] and [6] of a Roger’s Lab BAM file and the value was taken from the lines of the BAM file. If Key 1 was in both the WbDict and GoodChromosomes list, the key was printed out then appended to its value. If Key 1 was not in both the WbDict and Good Chromosomes list, the key was not appended to a value but the key did return its value. The same method of searching for the key in both the Key list and the GoodChromosomes list applied for the Key 2 list. Finally, the class method defined inside the class was called outside the class to print the WbDict to an outfile. Once fully incorporated into the GUI package, WbDict will provide the package with a method of finding the *Wolbachia* genes that may have inserted in the *Drosophila* genome through HGT.

Next, a BAM file parser was created to take a BAM file as input and parse the file for only lines containing the NCBI reference genome ID NC\_002978.6 (*Wolbachia* endosymbiont of *Drosophila melanogaster*, complete genome) (Wu et al., 2004). Once the lines containing NC\_002978.6 were parsed from the original BAM file, they were written to an outfile. Therefore, a list of *Drosophila* genes in which *Wolbachia* genetic information had been detected could be produced inside the outfile.



**Figure 3.** OOP code used to write the BAM file parser.

After developing some of the new back end code, we focused on the production of the GUI using TKinter. TKinter is the a inherently distributed Python GUI building module and while other options of GUI builders exist, TKinter is the most readily available to the largest number of python users (Python 3.6.4 Documentation). TKinter is compatible with object oriented code design, making it an obvious choice for incorporating our new back end code for this project(Python 3.6.4 Documentation). While setting up the actual GUI is a static event more or less, having access to the GUI before all the back in code will allows for iteratively adding function to the whole system, or use a waterfall software development method. Moreso, the methodology of the button and entry fields in Tkinter allows for compartmentalization of methods that we will need to use for this project, but will also allow for easy modification at a later time, due to a partition between the two classes.

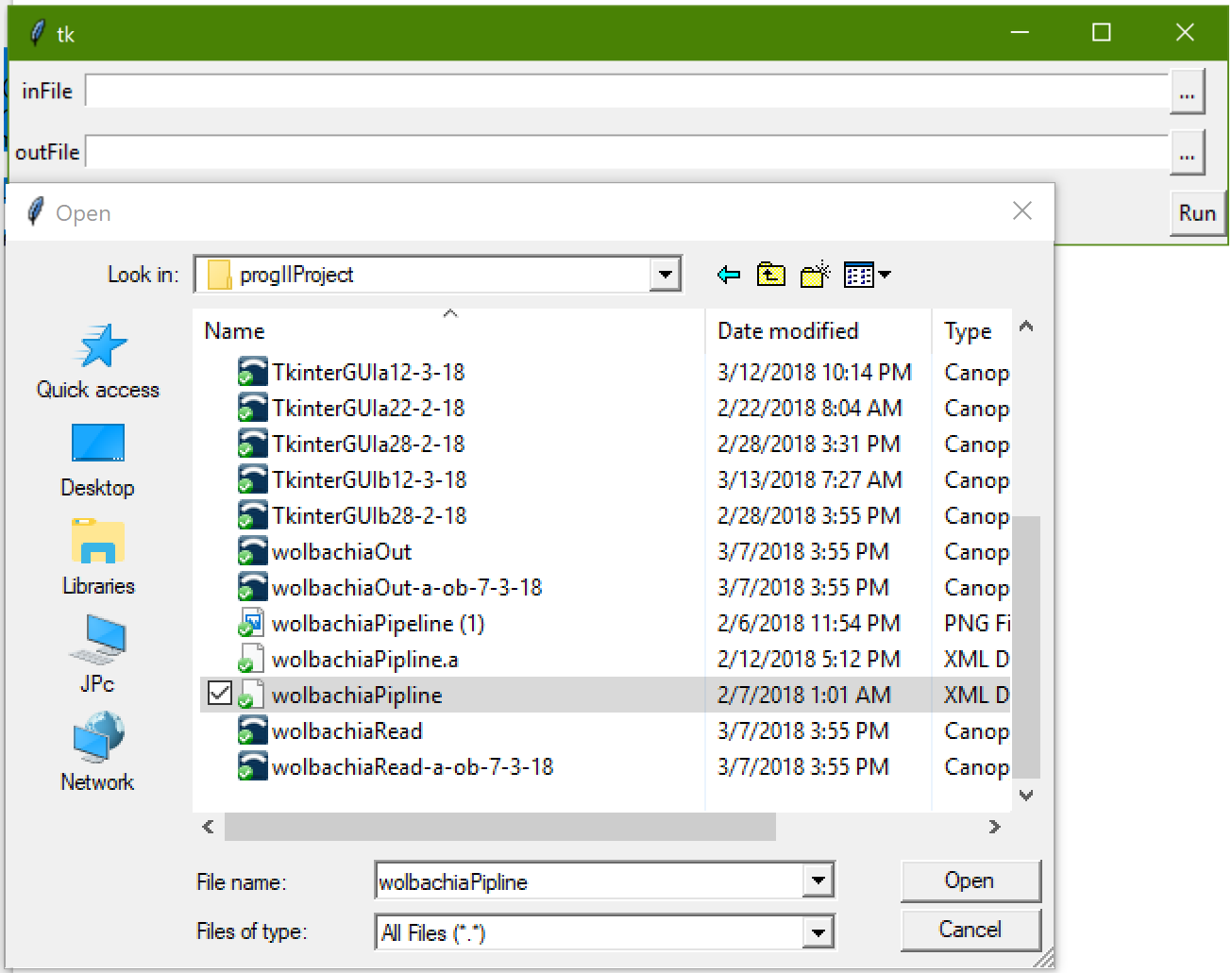


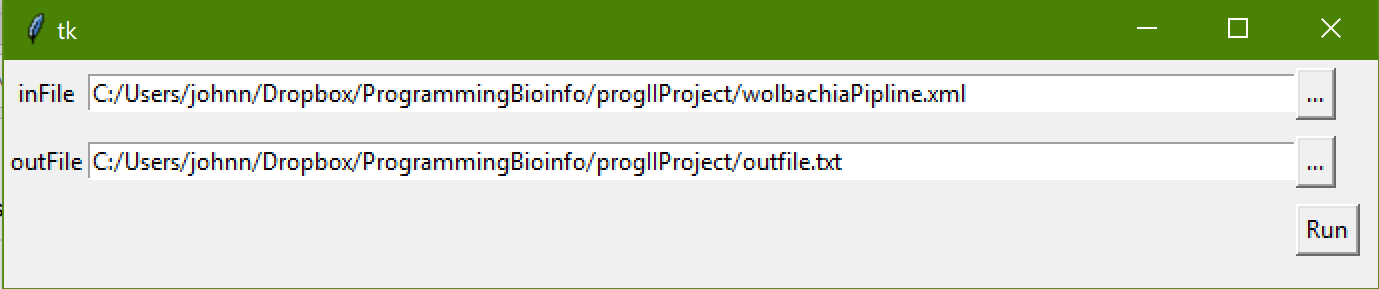
**Figure 4.** Code written to generate the functional GUI.

The GUI module was imported along with two modules that allow it to run the backend analysis desired in our project. First in Figure 4, we established the wolbachiaGUI class which reads and creates files, and calls the rest of the pipeline. Next, we set parameters for the input and output windows and created and named the GUI’s buttons. Then methods for file input/output and added them to the functionality of the GUI by nesting them into the button actions themselves, making a partition between the actual functional methods of BamReader and the GUI. At this point the GUI could open the native file browser for utility. A method to build a “safety net” was built in to prevent the end-user from closing the GUI without closing the program, which experimentally was found to cause kernel loop issues causing python to restart. Finally, we called the class method to actually run the program at the bottom of the code. Having the GUI instantiated without any other classes or methods needed for the actual analysis will keep debugging more manageable and allow for function expansion.

**Results:** Currently, we have a functional GUI. With rational design in mind, the GUI shown in Figure 5 can find a input file and create a location and name for a output file. The GUI takes the arguments defined by the user and passes it to the BamReader class allowing the class to parse files for a particular genome ID for use in analysis. In addition to our functional GUI, we have converted wolbachia reading, writing, and dic classes for incorporation into the OOP package that we are in the process of developing. These pieces of the entire wolbachia match methodology previously created will be debugged and nested into the currently functioning GUI.







**Figure 5.** Visual of the GUI (top image), the use of the GUI to select an input and an output file from the utility browser (middle image) and, the GUI with an input file named with an outfile (with location) specified (bottom image).

**Future Improvements:** Once everything is in working order, a “gene list” of likely HGT mutated *Drosophila* genes will be produced from the GUI and its input file. The output list will be a listing of genes that can be searched in FlyBase, the fly genome database (FlyBase, 2018). To test our GUI and converted runtime scripts, we will be obtaining a pseudo BAM (exemplary data) file from the Wolbachia reads BAM file (through subsetting the BAM file) to run through the GUI to see that it functions. This was not a debugging step (that will occur with actual BAM files-not pseudo BAM files). A possible extension of this project would be to produce a gene list that can be incorporated into multiple evolutionary gene/protein relationship databases. For example, we could enter the gene list into PANTHER to determine any evolutionary relationships between potential mutants and their protein products (Mi et al., 2016). As the project progresses, we will be debugging each of the converted runtime scripts as they are incorporated into the GUI to ensure that each component of the GUI works before adding more. We are considering the addition of an error handling system that will facilitate easier handling of the package by non-programmers.

**References:**

McGraw, Elizabeth A, and Scott L O'Neill. “Wolbachia Pipientis: Intracellular Infection and

Pathogenesis in Drosophila.” *Current Opinion in Microbiology*, Elsevier Current Trends, 29 Dec. 2003, [www.sciencedirect.com/science/article/pii/S1369527403001681](http://www.sciencedirect.com/science/article/pii/S1369527403001681).

Zabalou, Sofia, et al. “Natural Wolbachia Infections in the Drosophila Yakuba Species

Complex Do Not Induce Cytoplasmic Incompatibility but Fully Rescue the WRi Modification.” *Genetics*, Genetics, 1 June 2004, [www.genetics.org/content/167/2/827](http://www.genetics.org/content/167/2/827).

Marshall, Jeremy L. “Article Access.” *BioOne*, Nov. 2004,

[www.bioone.org/doi/pdf/10.1554/04-283](http://www.bioone.org/doi/pdf/10.1554/04-283).

Rogers, Rebekah L., et al. “Landscape of Standing Variation for Tandem Duplications in

Drosophila Yakuba and Drosophila Simulans | Molecular Biology and Evolution | Oxford Academic.” *OUP Academic*, Oxford University Press, 7 Apr. 2014, academic.oup.com/mbe/article-abstract/31/7/1750/2925730/Landscape-of-Standing-Variation-for-Tandem.

FlyBase. “FlyBase Homepage.” *FlyBase Homepage*, U.S. National Institutes of Health, 1 Mar.

2018, flybase.org/.

Bost, Kenneth. “Lecture 9 Dengue-2.” Epidemics and Plagues . Epidemics and Plagues Class

Huaiyu Mi, Xiaosong Huang, Anushya Muruganujan, Haiming Tang, Caitlin Mills, Diane Kang,

and Paul D. Thomas, *Nucl. Acids Res. (2016) doi: 10.1093/nar/gkw1138*

Wu, M, et al. “Wolbachia Endosymbiont of Drosophila Melanogaster, Complete Genome - Nucleotide -

NCBI.” *National Center for Biotechnology Information*, U.S. National Library of Medicine, 2004, www.ncbi.nlm.nih.gov/nuccore/NC\_002978.6.

“25.1. Tkinter - Python Interface to Tcl/Tk¶.” *25.1. Tkinter - Python Interface to Tcl/Tk - Python*

*3.6.4 Documentation*, Python Software Foundation, docs.python.org/3/library/tkinter.html.