Cloud Based Plasmid Annotation Tool

Leslie Smith & Dr. Mike Leih

Point Loma Nazarene University

Table of Contents

Abstract

Plasmid genome analysis is a primary way biologists can study and infer characteristics about antibiotic-resistant bacteria. However, analysis of plasmid genomes has been hindered by inconsistent gene naming, various incomplete databases, and the need for many different tools to identify and analyze plasmid features. We aim to alleviate some of these challenges by providing a single, cloud-based web tool that can link the plasmid genome annotation workflow. With use of the genome analysis tool Prokka, AWS, and RShiny, we are able to provide a web tool for user-friendly plasmid genome upload, annotation, and visualization.

Introduction

Why

As a result of the rise in prescribed antibiotics for the treatment of bacterial infections, antibiotic-resistant bacteria have quickly become a prominent public health threat (Ventola, 2015). The CDC estimates that each year at least 35,000 people die from an antibiotic-resistant bacterial infection, up from 23,000 deaths per year in 2013 (CDC, 2020). This widespread threat affects people in every industry, from healthcare to agriculture, and causes immense financial strain on the U.S. healthcare system (Ventola, 2015). A primary way bacteria acquire resistance to antibiotics efficiently is through the exchange of plasmids. Plasmids are small, circular genetic elements that are able to replicate independently of the bacterial chromosome and can transfer between bacterial genomes, allowing for new antibiotic resistant combinations. The analysis of plasmid genomes can allow biologists to infer important plasmid characteristics in order to identify which plasmids are likely to have antibiotic resistances, how likely these genes are to transfer, and how likely new antibiotic resistant combinations are to occur (Shintani, 2015). However, the information needed for this kind of analysis is in multiple databases with varying levels of quality. Additionally, the installation and use of command-line software often used for genome analysis is a hindrance for biological research, as software installations are time-consuming and most biologists are unfamiliar with command-line operations. These complexities have deterred the analysis and utilization of plasmids in the fight against antibiotic-resistant bacteria.

What

This research aims to provide a single cloud-based web application that links all of the tools and databases necessary for consistent and reliable plasmid genome annotation and analysis. The web application automates the plasmid sequencing, annotation, and analysis workflow developed by Point Loma Nazarene University's (PLNU) Microbiology Lab, led by Dr. David Cummings, Dr. Dawne Page, and Dr. Ryan Botts. In this workflow, a file containing a plasmid sequence is passed into the annotation tool Prokka via the command line with arguments to specify how the genome should be annotated. Prokka is a command-line software used to identify and label prokaryotic genomes (Seemann, 2020). When the

genome annotation is complete, Prokka creates a folder containing 12 output files that are downloaded automatically to the host computer. From this folder, the .tbl file containing identified genes from the sequence is passed into a Python script which cleans the gene data and puts it into a csv file. This csv file can then be used in the programming language R to visualize annotation results.

While this workflow reliably analyzes plasmid genomes, its execution is limited to those who 1) are able to install Prokka correctly on their local computer, 2) have access to a reliable database of plasmid genes, 3) have Python scripts to correctly pull out useful information from Prokka's output files, and 4) know how to utilize R for useful visualization of genomic data. The goal of this research is to automate this workflow through a web application in order to mitigate these barriers and allow the workflow to be utilized more broadly.

How

The web tool is hosted on Amazon Web Services' Elastic Computing (AWS EC2) instance. R, RStudio, RShiny, and Prokka are installed on the created instance. The web application is developed in RShiny, the application workflow is as follows:

1) The application accepts a file input from the user and calls Prokka on it.

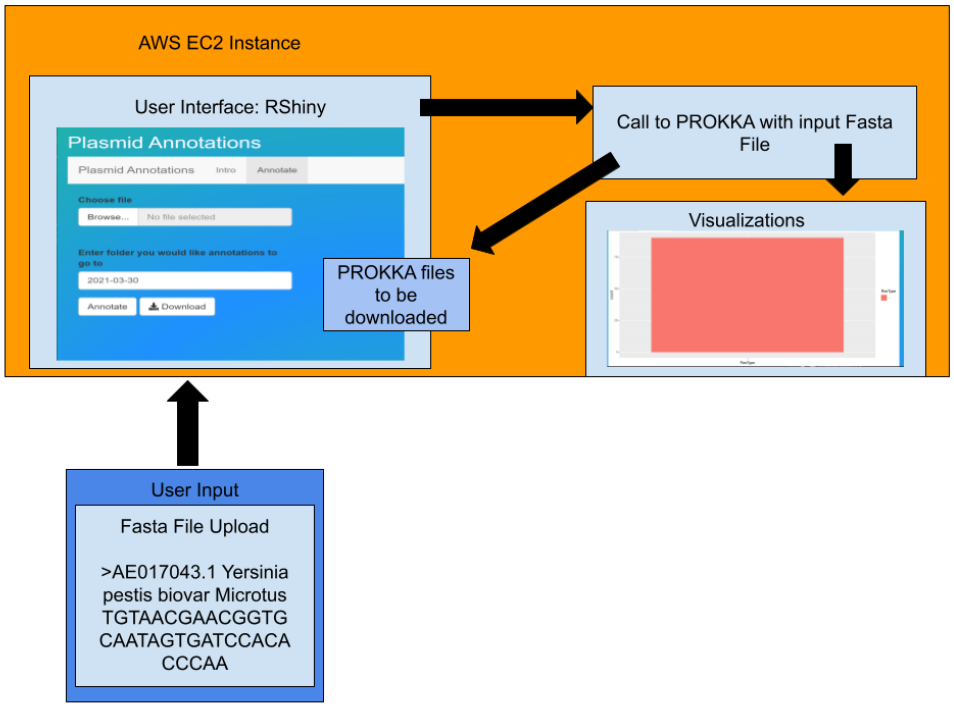
2) The output file from Prokka is passed into a Python script.

3) The Python script cleans Prokka's output file for easy visualization in R.

4) The visualization of antibiotic resistance genes contained in the input file is output to the user.

5) Prokka output files are made available for download.

6) The application is pushed to the shiny-server to be accessed via the internet.



Our desired project outcomes are:

1. Create a cloud based web tool to act as a user interface for the plasmid annotation tool Prokka with the ability to download Prokka output files.
2. Automation of PLNU's plasmid annotation workflow through the cloud based web tool. This includes the ability to call Prokka on a user uploaded file, run Prokka output through our Python script, and provide an insightful visualization of genes from the annotation.
3. Push the web tool to shiny-server to be accessed by the public.

Methodology

AWS EC2 creation:

The first step in creating the cloud based web tool was to choose a web platform. For this tool, an Amazon Web Service (AWS) EC2 instance was selected. Amazon Web Services (AWS) is a cloud computing platform that offers many different cloud computing services. The Elastic Computing (EC2) instance creates a computer environment, based on user specifications, which can be developed to the specific needs of an application. AWS was selected due to its versatility and educational resources. For example, an AWS EC2 instance allows the user to select memory volume, computational power, an operating system, and security measures based specifically on an application's needs. Additionally, AWS gives an allowance of $50 for research, which enables students to test out applications and tools up to $50 at no cost. In this process, a free tier t2.micro Ubuntu 20.04 Amazon Machine Image was created. The t2.micro has one central processing unit (CPU) and one GiB (gibibyte) of memory. The t2.micro was chosen because it had the computational power needed and can be run for free. The Ubuntu operating system was chosen because it is a light weight, stable system that can easily run and support RShiny, it was also one of the supported operating systems defined by the RShiny installation guide.

During instance set up, configurations, storage, and tags were left in their default settings. A new security group was created to define inbound and outbound rules specific to the application. The security groups in AWS allow users to define which ports an application can access and be accessed from. A port is a "place" in an operating system that helps computers organize network traffic. Different ports are used to run different internet protocols and services. The web tool needed to be accessed on the internet via hypertext transfer protocol (HTTP), which runs on port 80, so port 80 was opened in the security group's inbound rules. Similarly the port 8787 was opened in order to run RStudio, port 3838 and 3939 were opened to run RShiny, port 22 was opened to allow the application to be accessed using secure shell, and port 443 was opened for hypertext transfer protocol secure(HTTPS). After the security group was finalized, the instance was launched, the key pair file can now be used to connect to the EC2 instance.

Program Development Selection:

Following AWS EC2 instance creation, the next step was to select the programming language and application development environment. R, RStudio, RShiny, and a shiny server were selected. R is an open-source programming language specialized for data management and analysis. RStudio is R's integrated development environment, meaning that it provides an interface to write, run, and debug R code(Rstudio, n.d.). RShiny is an R package that allows users to build web applications from R and the shiny server is a web server designed specifically for RShiny applications (RShiny, n.d.). The success of hosting a genomic command line tool on RShiny and the shiny server was shown in Meharji Arumilli's paper on her web tool "WebGQT" (Arumilli, 2020). RShiny has many tools and libraries that can be easily integrated into an application. When looking at other languages and environments, like PHP, most of the back-end tools, such as a database and user interface tools, would have to be built separately. RShiny already has these tools robustly built in. For example, RShiny provides libraries like CSS, which allows for unique application customization, ggplot2, which allows easy graphing of genome annotation results, and the reticulate library, which allows for the utilisation of previously written Python scripts in the web tool's workflow.

Prokka Software:

After installation of all R tools, installation of the annotation software Prokka was initiated. While Prokka is most easily downloaded through packages like Conda and the Bioconda channel, these packages could not be successfully installed on the EC2 instance in this project. Alternatively, Prokka was downloaded with the commands for an Ubuntu machine (specified below in Appendix A). If possible, it is recommended to utilize installation methods through Conda and the Bioconda channel as this method of installation is likely to have the least complications.

Application Development:

With all necessary installations completed, coding of the RShiny app in RStudio on the AWS EC2 instance was begun. In the RShiny application, there are two primary files: server.R and ui.R. The server.R file contains all server logic, i.e. logic for what to do when a button is pressed, file deletion, the system call to Prokka, and a trigger to create the graph after Prokka is done running. The ui.R file holds all the objects which need to be presented to the user in the application's user interface, however all functionality for these objects resides in the server.R files. An additional file of importance is the FunctionTest.py file. This contains the scripts written by Point Loma Nazarene University's Microbiology student lab worker Mariele Lensink (Lensink, 2019). This script takes in the .tbl file from Prokka and outputs four different files, one from each step in the scripting process, with the final file optimized for compatibility with R. To ensure file paths are passed into the FuntionTest.py script correctly, a string variable for the file name is created and reused throughout the script. Different endings are then added to this base file name in order to have each output file correspond to a step in the scripting process. Using the same name for every run allows for easy file path parameter passing and helps conserve memory on the instance. At the beginning of each run, the server deletes all output files from the previous run which allows the file names to be reused with no issues. The final output from this script is then plugged into the plotting function to be graphed.

Application launch:

With the workflow built and running successfully in RStudio, the needed application files were copied over to the shiny server in the directory /srv/shiny-server/. This directory is where the shiny server looks for applications by default. From here multiple applications can be hosted, as long as each application has its own directory containing the necessary server.R and ui.R files. For the web tool presented, necessary files which need to be in the /srv/shiny-server directory are the server.R and ui.R files, the FunctionTest.py Python script, the database used for Prokka annotation, and the directory www

containing pictures used in the application. www is another default directory used by shiny to find the pictures, if any, used in an application.

Plasmid Annotation Workflow Details:

Database Construction:

A primary problem in plasmid genome annotation is the inconsistency of plasmid gene names and function. As genome sequencing became cheap and fast, increasingly more plasmid genes were added to well known databases such as NCBI's GenBank Database with no consensus on naming conventions (Frost, 2014). Because of the initial inconsistencies in plasmid naming conventions, many of these databases hold the same gene under different names or different genes under the same name. This creates a clear problem when trying to identify gene functions and relationships. To account for this problem, a database containing only genes with reliable names was compiled by PLNU's Microbiology lab. Reliability was determined by pulling genes only from function specific databases such as TAfinder, a database containing genes solely for toxin-antitoxin systems, and ISfinder, a database solely for bacterial insertion sequences. Each database had its own unique, and often messy, format. To address this, scripts were written for each database using various combinations of regular expressions to sort through each gene and put it into fasta format. Fasta format is the most common format for nucleotide sequences and is the format that Prokka expects its databases to be in. Fasta format is identified by having a carot at the start of each new gene, followed by a single line of the gene's description and the gene's sequence.

```
>ML12345 ~~~~~~groupIncHI1B(R27)_1~~~
IPENRSL*AGPAPF*PWHSGYRGVMTVGVVHEADRIRCGKRCSAAQPDTGTGPVRTPYGP
VFAMSSGRVGFYRRRDTAY*ASNLA***FFPWNCRNTSIPPGACLSLSDHLYRVFS**MK
CWSGLYASAKHQSALSVVYQCAWYQTHKILQTPENSAPADLLNQAAG*TTHKRSHTMI*N
```

In addition to reformatting the genome sequences from each database, a tag is added to genes from our reliable database so they are easily found in the Prokka output. The database is named "Summer19CompleteDatabase.faa", an example of its tag in the Prokka output is shown below.

```
3160    3414    CDS
                inference   ab initio prediction:Prodigal:2.6
                inference   similar to AA sequence:Summer19CompleteDatabase.faa:ML12345
                locus_tag   EGPMOCAK_00005
                product Rep
```

Prokka Parameters:

Prokka is the genome annotation tool we use to annotate genomes in our web tool . Through the command line, users are able to pass in a file of gene sequences in fasta format to be annotated. The gene file is then run through multiple databases where the actual annotation occurs. The program outputs 12 data files, each holding information from the annotation (specifics on the contents of each file can be found at https://github.com/tseemann/prokka). Prokka utilizes databases such as the NCBI Bacterial Antimicrobial Resistance Reference Gene Database: a database from NCBI holding antimicrobial resistance genes, ISfinder: a database of insertion sequences, and UniProtKB (SwissProt): a database holding curated protein sequences. While Prokka will pull from these databases by default, it allows the user to pass in their own database as an argument to be annotated from first, before it checks its default databases. An example of the call to Prokka for the web tool is below:

```
irect("console", {¬
e("prokka --outdir ~/Documents/HonorsProject/outDir/ourDir1 --force --proteins ~/SummerResearch19/Summer19CompleteDatabase.faa --evalue 0.001 --addgenes ", example_file.fasta))
```

1. prokka: Tell the operating system user is making a call to the Prokka software
2. -- outdir ~/Documents/HonorsProject/ourDir/ourDir1: the output directory where Prokka should put its annotated output files
3. --force: tell Prokka to overwrite the output folder if it already exists
4. --proteins ~/SummerResearch19/Summer19CompleteDatabase.faa: tell Prokka to annotate from the reliable database first, and direction to where the reliable database resides
5. --evalue 0.001: similarity cutoff; evalue is how the significance of alignment found between genes is evaluated
6. --addgenes: file containing the genome sequences for Prokka to annotate

Python Script:

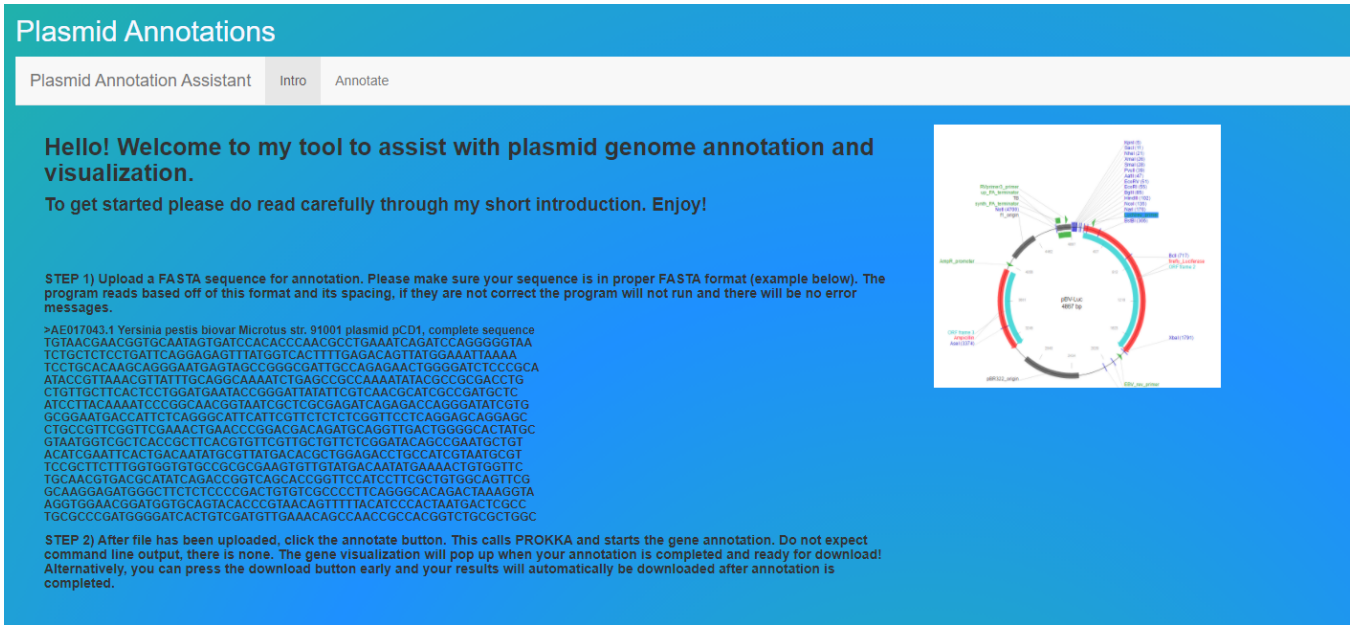
The Python script used to sort the Prokka output on the server was written by Mariele Lensink and consists of four primary functions:

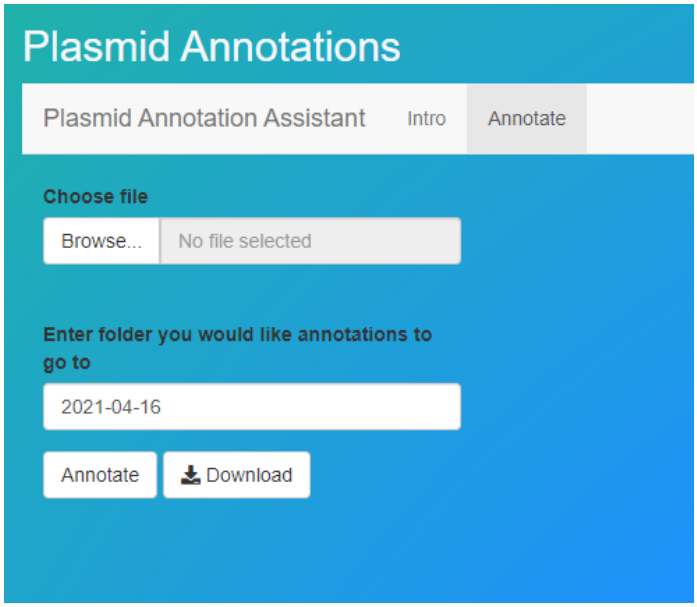
1. The first function, rowsFromProkka, takes in the Prokka .tbl output file and creates a list of each gene's start location, stop location, resistance name, resistance type, gene name, incompatibility group, plasmid name, and whether we want to keep the gene to include in our analysis. Only genes from the reliable database are kept. These genes are then written to an output file ending in "TableOutput.csv" which labels the file as the first step in the scripting process. This file will be the input file for the second step.
2. The second function, findNeighbors, takes in the output file from the first function and finds the nearest backbone genes both upstream and downstream of every antibiotic resistance gene. The output file is a csv file containing antibiotic resistance genes from the sequence, their upstream and downstream backbone genes, incompatibility group, and plasmid name. The file ends in "Neighbors.csv."
3. The third function, geneTableEditor, takes in the output file from step one, rowsFromProkka, and works similarly. This function cleans the genes to make them more compatible with R. For example checks are run to ensure all genes have start and stop positions as well as incompatibility groups. The output file is a csv file containing a list of genes and ends in "EditedGeneTable.csv."
4. The final function, resGeneEdit, takes in the output file from step three, geneTableEditor, and completes additional checks for blank lines, out of bounds errors, and removes subgroups. The output file is a csv containing a list of genes and ends in "RCompatibleTable.csv."

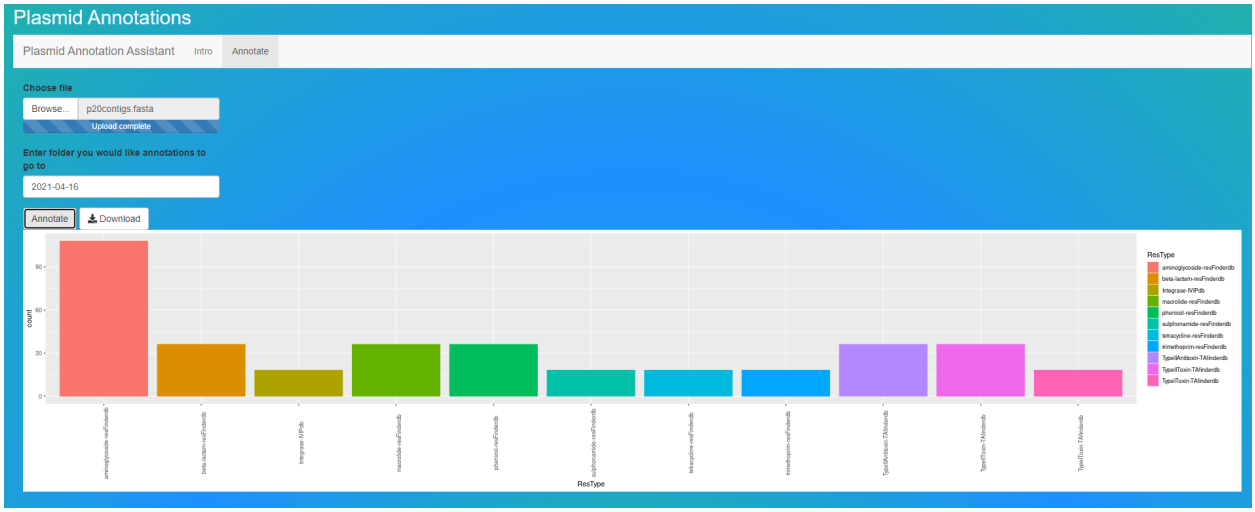
Results

The web tool presented in this paper successfully provides a user interface for the plasmid annotation tool Prokka using RShiny. It successfully automates the plasmid sequencing, annotation, and

analysis workflow developed by PLNU's Microbiology Lab in RShiny. It was launched to the EC2 instance successfully and can be accessed from the AWS EC2 instance's IP address. Screenshots of the final product are below.







Issues:

During the installation of Prokka through Ubuntu installation commands, the last command "--setupdb" did not execute correctly. The error "The sequence databases have not been indexed. Please run 'Prokka --setupdb' first" was encountered repeatedly, even after multiple runs of the "--setupdb" command. After debugging, it was discovered that the "--setupdb" command did not copy databases over to the directory where Prokka was searching for them as it was supposed to. After manually copying the databases over to the correct directory, Prokka ran with no further issues.

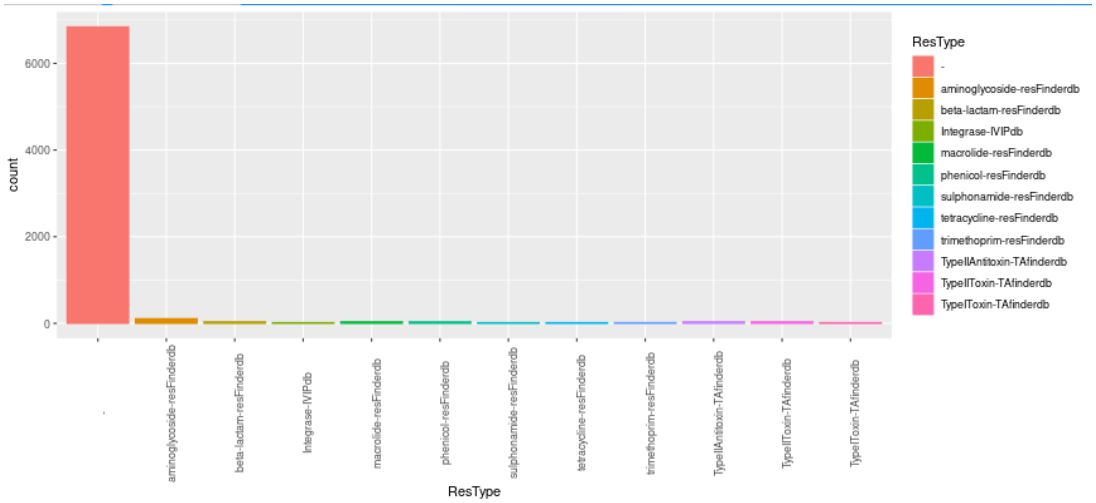
Another issue encountered was to ensure that the Python script FunctionTest.py would only be run after Prokka was finished annotating. To ensure this, initially an additional button for visualizations was added. This approach relied on a responsible user who would not push the visualization button until they were informed annotations were complete by the web tool; prematurely pushing this button caused the program to crash. To prevent any crashes, it was attempted to program the button so that it only appeared after the annotations were complete, however there were various issues with this and ultimately it could not be done successfully. The solution was to place the visualization function in the same code block as the Prokka call, which allowed the visualization plot to be created automatically after Prokka was done running and the output file from FuntionTest.py was created.

Significant difficulty was encountered when attempting to run the application correctly on the shiny server. The shiny server does not have the same default libraries installed as RStudio does, which caused the application to disconnect from the server when launched. In addition, by default the shiny server deletes log files, log files hold information such as error messages from each run of the shiny-server. With these logs automatically deleted, there was no way to see what exactly was causing the application to be disconnected. To solve this, several changes were made to the shiny-server.conf file found in /etc/shiny-server. The first change was to add "preserve_logs true;" to the configuration file, which tells shiny server not to delete the log files. Secondly, the command "run_as ubuntu" was changed to "run_as rstudio" which allowed the application to use file directories from the rstudio user and ensured the application would have access to files created during each application run. After log files were accessible, they showed the application would crash when trying to use the "reticulate" and "shinyWidget" libraries. After these packages were manually installed, the application ran a little longer before crashing again. Referring to the log files again revealed that some of the code which worked in RStudioto delete files from previous runs did not work the same way when running on the shiny server. These issues were resolved with minor changes in the R code.

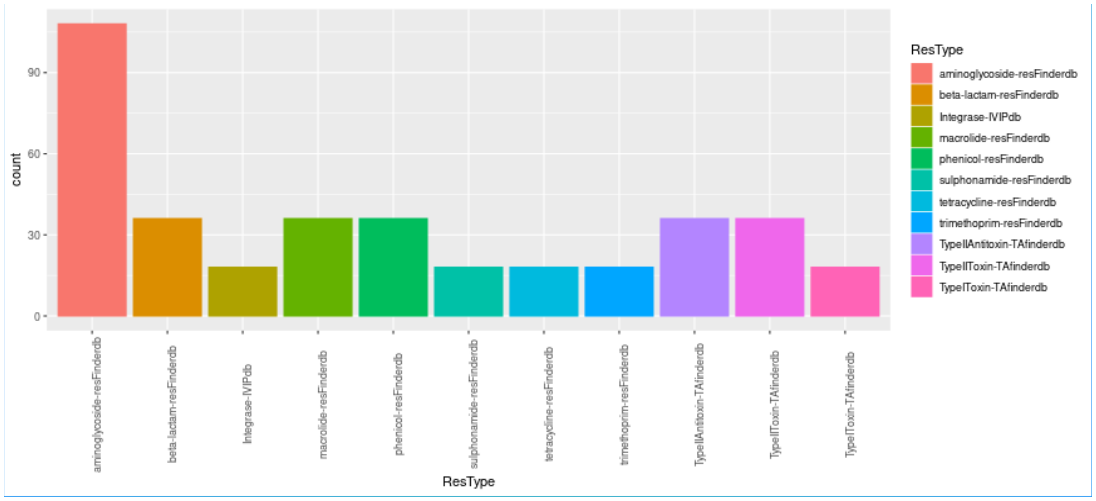
Evaluation:

The web tool was evaluated by biologist Dr. Dawn Page. It was recommended that a column in the output graph be removed. This column, labeled "-", was a place filler for any genes that were not antibiotic resistance genes. As most genes in a plasmid genome are not antibiotic resistance genes this column was overwhelmingly larger than other columns. After this column was removed it was much easier to extract information from the remaining columns. Examples from before and after this change are below.

Before:

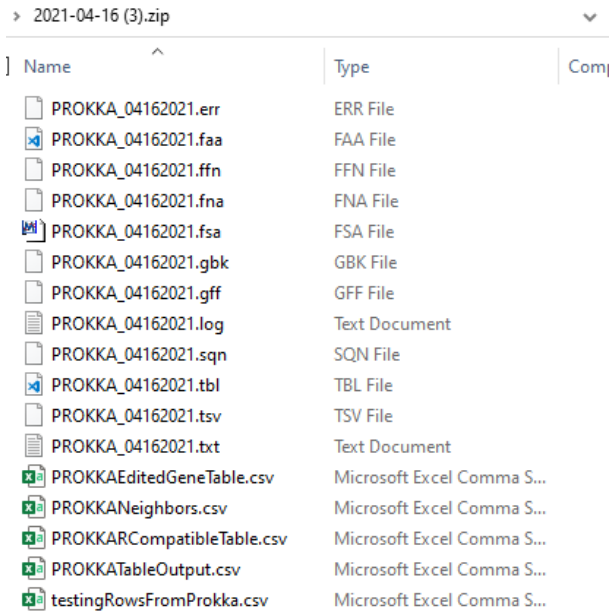


After:

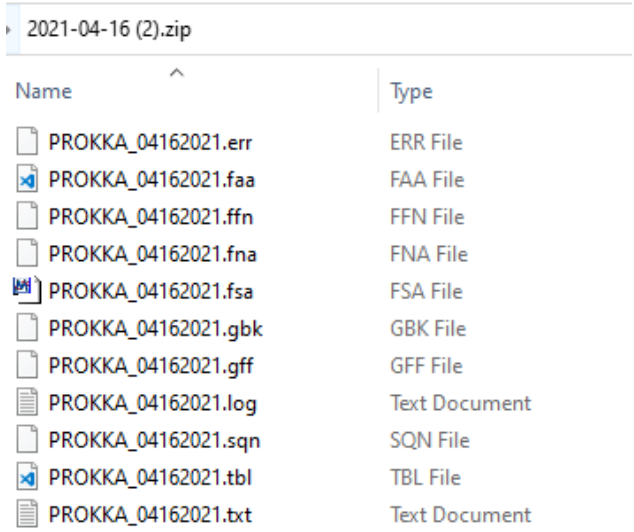


Additionally, it was recommended to only include Prokka output files in the users downloaded zip file. Previously, the csv files created in the Python scripts were included in the downloaded file as well, however these files are not likely to be utilized by users and therefore their inclusion in the downloaded file was not useful. Examples from before and after this change are below.

Before:



After:



Conclusion

Our primary research objectives to create a user interface for the annotation tool Prokka and to automate PLNU's plasmid annotation workflow were achieved. This tool provides a proof of concept for hosting Prokka and other command line tools on an AWS EC2 instance utilizing RShiny to build the

interface. Use of a t2.micro EC2 instance on AWS demonstrates that this application is not resource intensive and can be deployed on a small server at a low cost. Alternatively, this application could easily be hosted on a service like shinyapp.io, a self-service platform secured and maintained by RStudio, instead of having to host and maintain an individual server. With the rise of genome sequencing and necessity for genome annotation tools, easy-to-use interfaces like the one presented in this paper will become increasingly important and useful. Additionally, workflows with many moving parts, such as the plasmid annotation workflow presented in this paper, may be completed much faster with less complications when automated. Software installations take time and are localized to one computer, these constraints are easily alleviated when moving the software to a cloud-based web server which requires only a single installation of the required software and can be accessed from any computer by multiple people. Additionally, the utilization of a single cloud-based web server results in only a single computer system to be updated and maintained. Our implementation and deployment of command line software on a cloud-based web server prevents researchers from having to spend time completing difficult steps which often lead to various errors and complications.

Future Considerations

With our time constraint we were unable to output the command line information from Prokka to the user interface, therefore the user is not updated by Prokka of the progress of the annotation and will not be notified if the annotation failed. Realistically, Prokka output to keep the user informed of annotation progress is a necessity for widespread utilization of this tool. While we could not effectively get this feature working we believe there is a way for it to be done using the method “reactiveTimer” in a script running in the background while Prokka annotates. Alternatively, Prokka output could be written to a file to be accessed by the user in the case of annotation failure.

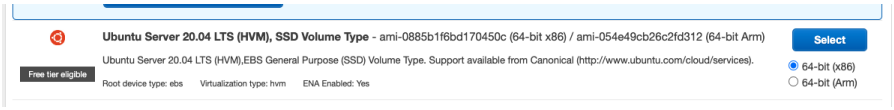
While the annotation tool Prokka catches many gene annotations, this application could be extended by adding an additional annotation software option to the interface to ensure maximum gene annotation capture and accuracy. For example, another workflow from PLNU’s microbiology lab

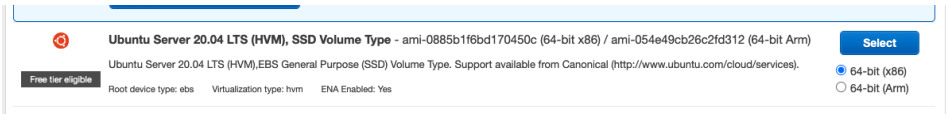
involved the use of the software Mobsuite in addition to Prokka. Further extension of the application could allow for more informative genome visualizations and make visualizations available for download.

Appendices

Appendix A

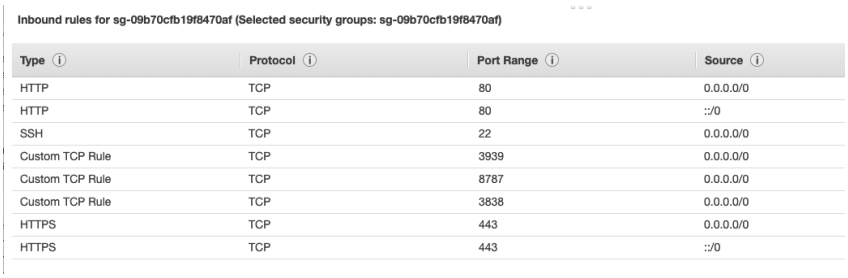
Start up an AWS EC2 Instance

1. Go to EC2 dashboard
    a. Instances
    b. Launch instances
    c. Step1:
        i. Choose an Amazon Machine Image (AMI)
            1. We chose most recent Ubuntu
            2. Ubuntu Server 20.04 LTS (HVM), SSD Volume Type 64-bit (x86)
            3. 

        ii. Choose an Instance Type
            1. free tier - t2.micro has 1 CPU with 1 GiB of memory
            2. Leave everything as default
            3. 

        iii. Configure Instance
            1. Leave as default
        iv. Add Storage
            1. Leave as default
        v. Add Tags
            1. Add “name” tag with value “annotation server”)
        vi. Configure Security Group
            1. Create new security group
                a. Security group name: anything you want, we suggest something to the point of r-shiny-server
                b. Description is optional
                c. Leave VPC
                d. Inbound Rules:
                    i. Description is optional
                        1. NOTE: During testing we allowed the source IP to be from anywhere, however it is strongly recommended to restrict the source IP address to sources from a single network.
                    ii. Type: HTTP, Source type: Custom, :Protocol: TCP, Source: 0.0.0.0/0, Port Range: 80

iii. Type: HTTP, Source type: Custom, :Protocol: TCP, Source: ::/0, Port Range: 80
iv. Type: SSH, Source type: Custom, :Protocol: TCP, Source: 0.0.0.0/0, Port Range: 22
v. Type: Custom TCP, Source type: Custom, Protocol: TCP, Source: 0.0.0.0/0, Port Range: 3939
vi. Type: Custom TCP, Source type: Custom, Protocol: TCP, Source: 0.0.0.0/0, Port Range: 8787
vii. Type: Custom TCP, Source type: Custom, Protocol: TCP, Source: 0.0.0.0/0, Port Range: 3838
viii. Type: HTTPS, Source type: Custom, Protocol: TCP, Source: 0.0.0.0/0, Port Range: 443
ix. Type: HTTPS, Source type: Custom, Protocol: TCP, Source: ::/0, Port Range: 443
x.



Inbound rules for sg-09b70cfb19f8470af (Selected security groups: sg-09b70cfb19f8470af)

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
SSH	TCP	22	0.0.0.0/0
Custom TCP Rule	TCP	3939	0.0.0.0/0
Custom TCP Rule	TCP	8787	0.0.0.0/0
Custom TCP Rule	TCP	3838	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0
HTTPS	TCP	443	::/0

Outbound Rules
xi. Type: All traffic, Destination Type: Custom, Protocol: All, Port Range: All, Destination: 0.0.0.0/0

Tags: Key = Name, Value = r-shiny-server

vii. Create a key pair :
1. Create new key pair
2. Name: rserverkp
3. File format: pem
4. Create key pair
a. It will then be downloaded to your computer, store it in a safe spot (this is the keys to the kingdom) recommended to put it in the .ssh file on your computer

d. Create Instance
i. Wait for instance to be created, this might take a few minutes

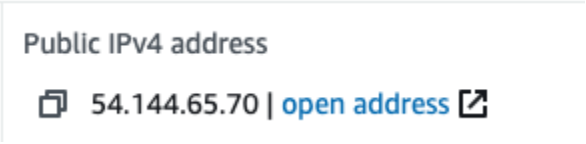
2. Go to Instances and select the instance you just created
a. Things you will often use from Instance Summary
i. Public IPv4 DNS
1. What it looks like: ec2-54-86-105-192.compute-1.amazonaws.com

2.
3. WARNING: IP addresses will likely change anytime you stop and restart you instance, so you will have to recopy the public IPv4 address, Private IPv4 Address, public IPv4 DNS and Private IPv4 DNS into commands anytime you stop and start your instance
4. You will use this to connect to your instance from command line, example of command is:

   Ssh -i ".ssh/rshinykp.pem"
   ubuntu@ec2-54-86-105-192.compute-1.amazonaws.com
   ".ssh/rshinykp.pem" - this will be unique to wherever you stored you kp

ii. Public IPv4 Address

1. What it looks like: 54.86.105.192



When connecting:

```
ssh -i ".ssh/rshinykp.pem" ubuntu@ec2-54-86-105-192.compute-1.amazonaws.com
```

a. Warning about fingerprint "Are you sure you want to continue connecting?" = yes

Commands to run for install of r server, r, rstudio, RShiny, Python

1. Sudo apt update
2. Sudo apt-get upgrade -y
3. Sudo apt-get install r-base -y
4. Create new user:
   a. Sudo useradd rstudio
   b. Sudo mkdir /home/rstudio
   c. Sudo passwd rstudio
   d. Enter new password
5. Sudo apt-get install gdebi-core
6. Sudo apt-get install r-cran-httpuv
7. wget https://download2.rstudio.org/server/bionic/amd64/rstudio-server-1.3.1093-amd64.deb
8. sudo gdebi rstudio-server-1.3.1093-amd64.deb
9. wget https://download1.rstudio.org/desktop/bionic/amd64/rstudio-1.2.5042-amd64.deb
10. sudo apt install ./rstudio-1.2.5042-amd64.deb
11. sudo gdebi rstudio-server-1.3.1093-amd64.deb

```
12. sudo su - \ -c "R -e
    \"install.packages('shiny',repos='https://cran.rstudio.com/')\""
13. wget
    https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.16.958
    -amd64.deb
14. sudo gdebi shiny-server-1.5.16.958-amd64.deb
15. sudo apt-get update
16. sudo apt-get install python3.8 python3-pip
17. //switch user to shiny
18. su - shiny
19. R //go into R
20. install.packages("shiny")
21. install.packages("reticulate")
```

Install prokka on the instance:

```
1. sudo apt-get install libdatetime-perl libxml-simple-perl
   libdigest-md5-perl git default-jre bioperl
2. sudo cpan Bio::Perl
3. git clone https://github.com/tseemann/prokka.git $HOME/prokka
4. $HOME/prokka/bin/prokka --setupdb
```

Common Errors

- If you are unable to get into rstudio after installation due to an error in loading shared libraries: libsmim3.so, complete following command:
  `sudo apt install libnss3`
- Issue: the sequence database have not been indexed. Please tun 'prokka --setupdb' first
    - Prokka looks for db in /var/lib/prokka/db, for some reason here it does not load all of the databases correctly, it only loads the cm and genus, so we need to copy over the hmm and the kingdom databases.
- If you ever run out of space on the instance (this happened to us a few times before we ended up at 20GB)
    - EBS -> Volumes -> select the correct volume -> actions -> modify volumes -> in size change to whatever size you believe you need (we went from 8 GB to 20 GB)

Bibliography

About antibiotic resistance. (2020, March 13). Retrieved April 13, 2021, from https://www.cdc.gov/drugresistance/about.html

Arumilli, M., Layer, R., Hytönen, M., & Lohi, H. (2020, February 10). Webgqt: A shiny server for genotype query tools for model-based variant filtering. Retrieved April 12, 2021, from https://www.frontiersin.org/articles/10.3389/fgene.2020.00152/full?report=reader

Frost, L. S., & Thomas, C. M. (2014, May 4). Naming and Annotation of Plasmids. Retrieved April 24, 2021, from https://link.springer.com/referenceworkentry/10.1007%2F978-1-4614-6436-5_568-2

Http. (2015, May 25). Retrieved April 12, 2021, from https://techterms.com/definition/http#:~:text=Stands%20for%20%22Hypertext%20Transfer%20Protocol,%2C%20laptop%2C%20or%20mobile%20device

Lensink, M. (2019, May 04). Computational analysis of the relationships between antibiotic resistance AND Plasmid BACKBONE GENES. Retrieved April 13, 2021, from https://nnu.whdl.org/computational-analysis-relationships-between-antibiotic-resistance-and-plasmid-backbone-genes?language=en

Rstudio. (n.d.). Retrieved March 13, 2021, from https://www.rstudio.com/products/rstudio/

RShiny. (n.d.). Retrieved march 01, 2021, from https://shiny.rstudio.com/

Seemann, T. (2020, August 3). README.md Prokka: rapid prokaryotic genome annotation. GitHub. Retrieved April 13, 2021, from https://github.com/tseemann/prokka/blob/master/README.md

Shintani, M., Sanchez, Z., & Kimbara, K. (2015, March 12). Genomics of microbial plasmids: Classification and identification based on replication and transfer systems and host

taxonomy. Retrieved April 12, 2021, from https://www.frontiersin.org/articles/10.3389/fmicb.2015.00242/full

Tcp-ip ports and how they work. (n.d.). Retrieved April 13, 2021, from https://www.bullguard.com/bullguard-security-center/pc-security/computer-security-resources/tcp-ip-ports

Teja, R. (2021, April 03). 16 types of computer ports and their functions. Retrieved April 12, 2021, from https://www.electronicshub.org/types-of-computer-ports/

Ventola, C. (2015, April). The antibiotic resistance crisis: Part 1: Causes and threats. Retrieved April 12, 2021, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4378521/