The Diversity Found Among Carbapenem-Resistant Bacteria

Galen Edward Card

Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Microbiology Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/6949

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
The Diversity Found Among Carbapenem-Resistant Bacteria

Galen Edward Card

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Richard A. Robison, Chair
Kim L. O’Neill
Joel S. Griffitts

Department of Microbiology and Molecular Biology
Brigham Young University

Copyright © 2018 Galen Edward Card
All Rights Reserved
ABSTRACT

The Diversity Found Among Carbapenem-Resistant Bacteria

Galen Edward Card
Department of Microbiology and Molecular Biology, BYU
Master of Science

This work will look at two factors that add to the diversity of carbapenem resistant bacteria. First, it focuses on the diversity of carbapenemase resistance plasmids. 446 plasmids were characterized by size, gene content and replicon groups. We identified that on average, over 30% of the encoded proteins on each plasmid have an unknown function. Plasmid sizes ranged from 1.6kb to 500kb, with an average of around 100kb and median of 80kb. Additionally, six replicon groups account for 80% of all the carbapenemase resistance plasmids. We also highlight the lack of data available for carbapenemase carrying plasmids from bacterial genera other than Escherichia and Klebsiella, and plasmids that carry the New Delhi metallo-β- lactamase or the Verona-integron encoded metallo-β-lactamase.

Second, we characterized the β-lactamase diversity of a single carbapenemase resistant Klebsiella pneumoniae. This isolate encodes six distinct β-lactamases, all of which are functional, and three of which are redundant. Additionally, we determined that the CTX-M-15 cephalosporinase imparts a greater fitness when grown in aztreonam (a monobactam) than ceftazidime (a cephalosporin). Finally, we show that individually, these β-lactamases do not account for the elevated levels of resistance seen in the parent strain, indicating that the passive resistance mechanisms (i.e. efflux pumps, altered membrane porins) may play a larger role than originally thought.

Keywords: Antimicrobial resistance, β-lactamase, carbapenem resistant Enterobacteriaceae, Klebsiella pneumoniae, Extended-spectrum β-lactamase, ESBL, plasmid, horizontal gene transfer
ACKNOWLEDGEMENTS

I would like to take this space to express my thanks to all who have helped me in my pursuit of this Master’s. First, I’d like to thank my committee. Dr. Robison, thank you for providing an environment where I could grow and develop individual thought in my research. Dr. Griffitts, thank you for your help and guidance through my struggles with the molecular cloning work. Dr. O’Neill, thank you for your key insights in all the committee meetings, your input helped me address issues that I would not have noticed on my own.

Second, my sincerest thanks to my family. Amy and Liam, coming home to you makes going back to the lab the next day easier. For my parents, thank you for your support and glowing pride in me. To the Underwoods, thank you for providing a place for us to live and more, allowing me to focus on academics.

Third, thank you to my fellow graduate students in the Robison Lab and the MMBio Department. Your friendship and assistance has been immeasurable.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td>i</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>The Rise of Antimicrobial Resistance</td>
<td>1</td>
</tr>
<tr>
<td>The use of antimicrobials in agriculture</td>
<td>1</td>
</tr>
<tr>
<td>The clinical misuse of antimicrobials</td>
<td>2</td>
</tr>
<tr>
<td>Spread of antimicrobial resistance within a bacterial population</td>
<td>2</td>
</tr>
<tr>
<td>β-lactams and Their Hydrolases</td>
<td>5</td>
</tr>
<tr>
<td>Extended-Spectrum β-lactamases</td>
<td>8</td>
</tr>
<tr>
<td>Carbapenems and Their Hydrolases</td>
<td>11</td>
</tr>
<tr>
<td>Klebsiella pneumoniae carbapenemase</td>
<td>11</td>
</tr>
<tr>
<td>New-Delhi metallo-β-lactamase</td>
<td>12</td>
</tr>
<tr>
<td>Verona integron-encoded metallo-β-lactamase</td>
<td>12</td>
</tr>
<tr>
<td>Imipenem resistant Pseudomonas metallo-β-lactamase</td>
<td>12</td>
</tr>
<tr>
<td>Summary</td>
<td>13</td>
</tr>
<tr>
<td>CHAPTER 1</td>
<td>14</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>14</td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>14</td>
</tr>
</tbody>
</table>
METHODS ................................................................................................................................ 16

Sequence acquisition ............................................................................................................. 16

Plasmid gene composition .................................................................................................... 17

Incompatibility group/Replicon typing and plasmid characterization .................................. 17

Statistical analyses ................................................................................................................ 17

RESULTS .................................................................................................................................. 18

Plasmid gene composition .................................................................................................... 18

Plasmid Incompatibility group/Replicon typing ................................................................... 20

DISCUSSION ............................................................................................................................ 20

Funding ...................................................................................................................................... 23

Acknowledgements ................................................................................................................... 23

Conflicts of Interest ................................................................................................................... 23

CHAPTER 2 ................................................................................................................................. 24

ABSTRACT .............................................................................................................................. 24

INTRODUCTION ..................................................................................................................... 24

METHODS ................................................................................................................................ 25

Genome sequencing, assembly, and annotation ................................................................. 25

β-lactamase cloning .............................................................................................................. 26

Growth curve analysis ......................................................................................................... 27

Statistical analysis ............................................................................................................... 29

RESULTS .................................................................................................................................. 29

*Klebsiella pneumoniae* 1300761 possesses six β-lactamase-encoding genes .................... 29

Resistance profiles of β-lactamase clones ......................................................................... 29
β-lactam preferences for a single β-lactamase ................................................................. 30

Growth of the parent strain *K. pneumoniae* 1300761 is not inhibited by any β-lactam tested ........................................................................................................................................ 33

DISCUSSION .................................................................................................................... 33

ACKNOWLEDGEMENTS ................................................................................................. 34

SUMMARY ...................................................................................................................... 35

REFERENCES ................................................................................................................ 36

APPENDIX A .................................................................................................................. 46

APPENDIX B .................................................................................................................. 57

Supplementary Bioinformatics Methods ......................................................................... 57

APPENDIX C .................................................................................................................. 99

APPENDIX D .................................................................................................................. 100

APPENDIX E .................................................................................................................. 101
LIST OF TABLES

TABLE 1: The four classes of β-lactam antibiotics and examples of each. ................................. 6

TABLE 2: PCR primers for the cloning of β-lactamase genes found in Klebsiella pneumoniae
1300761 ........................................................................................................................................ 28

TABLE 3: Key Words used to characterize CR-plasmid gene content. ..................................... 46

TABLE 4: CR-plasmid accession numbers ................................................................................. 99

TABLE 5: Percent of plasmids belonging to each incompatibility group.................................. 100

TABLE 6: Relative abundance of incompatibility groups among carbapenemase-carrying
plasmids. ................................................................................................................................. 101
LIST OF FIGURES

FIGURE 1: Acquisition of antibiotic resistance gene within a transposable element. ...................... 4
FIGURE 2: The four classes of β-lactam antibiotics. ........................................................................ 7
FIGURE 3: β-lactamase mediated hydrolysis of a cephalosporin. .................................................. 9
FIGURE 4: ESBL variants of TEM-1. ............................................................................................ 10
FIGURE 5: Characteristics of carbapenemase-encoding plasmids. ............................................... 19
FIGURE 6: Relative abundance of incompatibility groups among plasmids. ................................. 21
FIGURE 7: Growth curves grouped by growth conditions. .......................................................... 31
FIGURE 8: Growth curves of the β-lactamase clones. .................................................................. 32
INTRODUCTION

The Rise of Antimicrobial Resistance

Since the dawn of the antimicrobial era in 1937 with the introduction of sulfonamides, and the subsequent resistance of microbes arising in 1942, we have been in an arms race against rapidly evolving bacteria (1). Like clockwork, with each implementation of a new antimicrobial, resistance to that antimicrobial has appeared shortly thereafter (2). During the past several decades we have seen the rapid emergence of multi-drug resistant (MDR) and pan-resistant bacteria (3). With limited treatment options for these MDR organisms, and no treatments for pan-resistant organisms, we are facing what is being called the post-antimicrobial era, a time in which a seemingly routine infection presents the threat of death. Indeed, this threat is real with fatality rates of certain MDR bacteria reaching 50% (4). Many factors play a part in the rise and dissemination of antimicrobial resistance. The most important are the use of antimicrobials in agriculture, the clinical misuse of antimicrobials, and the facile spread of resistance within a bacterial population.

The use of antimicrobials in agriculture – What started as a prophylactic measure to prevent loss of livestock and enhance weight gain of food animals has led to a burgeoning healthcare crisis (5). Furthermore, it has been predicted that antibiotic use in agriculture will increase by 67% from 2010 through 2030, despite the restrictions that have been placed on their use in many countries (6). When antimicrobials are mixed in livestock feed, they quickly become diluted as rain and runoff mixes with the feed. The diluted antimicrobial then reaches a sub-inhibitory concentration that doesn’t kill some bacteria. Instead, it creates a selective pressure that kills a majority, allowing the few bacteria that can cope with the diluted antimicrobials propagate (7-10). This leads to rapid mutation and evolution as the bacteria improve their
resistance mechanisms to the antimicrobials. This then contributes to the human healthcare crisis as the zoonoses found on farms enter the human population through contaminated food products (11-13). Studies have shown that many species within the *Enterobacteriaceae* family that are found on farms have also been found on hospital surfaces and isolated from infected patients (14-17), giving credence to the threat posed to human health from the use of antimicrobials in agriculture.

*The clinical misuse of antimicrobials* – The misuse of antimicrobials in a clinical setting has two parts: the prescription of antimicrobials for a non-susceptible infection and prescribing a prolonged antimicrobial regimen (2, 18-20). Since antibiotics have no effect on viral infections, using them to treat viral infections only provides an opportunity to select for resistant isolates and should not be done. Second, the World Health Organization has shown through current research that prolonged courses of antimicrobials may increase the rates of antimicrobial resistance (2, 21). This persistent exposure to antimicrobials provides an environment wherein the bacteria have time to mutate and develop resistance to the antimicrobial. Proper antimicrobial stewardship in the healthcare setting is essential if we are to slow the spread of resistance.

*Spread of antimicrobial resistance within a bacterial population* – The third, and perhaps most important, factor that contributes to the rise and dissemination of antimicrobial resistance is the facile transfer of antimicrobial resistance genes within a bacterial community. Many of the antimicrobial resistance genes are found within mobile genetic elements such as plasmids and transposons. Transposons, in their simplest form, are stretches of DNA that encode machinery that can replicate, excise, and integrate these regions into other DNA sequences (22). Throughout their “lifespan”, transposons can acquire genes from their host chromosome and transfer them to plasmids (2, 18, 23-25). These genes can then be passed within a bacterial community as the
plasmids are shared. The opposite can also occur. Resistance genes can be transferred from a plasmid to the host’s chromosome. This is a permanent event. Once the resistance gene has entered the chromosome, there is no straightforward way to eliminate the gene. On the other hand, plasmids are more transient, and there are methods that can ‘cure’ bacterial strains of plasmids (26-28). Figure 1 shows a possibility of how a transposon may acquire and transfer resistance genes as it inserts and removes itself from the area surrounding the gene.

As mentioned, transposons can, and often do, integrate into plasmids. A plasmid is a circular piece of extra-chromosomal DNA that is maintained and replicated along with the chromosome. Ranging in size from less than 1kb to well over 200kb, they have the capability of carrying numerous genes. These genes can fall into several categories ranging from basic housekeeping or metabolism genes, to critical virulence genes, like the toxins responsible for the lethality of *Bacillus anthracis*. One unique feature of some plasmids is that they also carry a cluster of *tra* genes, or transfer genes. These genes provide a means for the plasmid to pass promiscuously between strains of bacteria via horizontal transfer (2, 24). Plasmids can also carry genes that cause the host bacterium to die if it does not retain the plasmid (29-32). These ‘plasmid addiction systems’ function using a toxin/antitoxin strategy. Encoded on the plasmid is a toxin, and its corresponding antitoxin. Of these two protein products, the toxin component is more stable. If the plasmid is lost from the host strain, the levels of antitoxin within the cytoplasm will decrease as it degrades quicker than the toxin. This in turn leaves the toxin free to exert its lethal effects within the cell. All these mechanisms contribute to the facile spread and maintenance of antimicrobial resistance among bacterial populations.
FIGURE 1: Acquisition of antibiotic resistance gene within a transposable element.
1) Wild type DNA receives transposon. 2) A transposition event occurs, leaving a scar upstream of an antibiotic resistance gene as the transposon jumps downstream. 3) A second transposition event leads to two possible scenarios where the antibiotic resistance gene is transferred to another mobile genetic element such as a plasmid. Other recombination events are possible; however, they do not transfer the antibiotic resistance to the new DNA.
**β-lactams and Their Hydrolases**

The most widely known class of antibiotics and one for which resistance has become a grave issue due to the reasons mentioned, are the β-lactams. So named due to the presence of a β-lactam ring as the central chemical backbone, these antibiotics have a four-membered ring, as depicted in Figure 2. Table 1 lists the four classes of β-lactams and provides examples of each. The first β-lactam antibiotic is also the original antibiotic discovered, penicillin. Penicillin and its derivatives, along with all β-lactams employ the same mode of action to kill bacteria. Known as cell wall inhibitors, β-lactams bind to and inactivate the penicillin-binding protein. This protein, so named for penicillin’s action against it, is responsible for covalently cross-linking peptidoglycan during bacterial cell wall synthesis. Without cross-linked peptidoglycan, cell morphology becomes more elongated, cell structure is fragile, and bacterial lysis occurs in most environments as water rushes into the cell to balance osmolarity. However, as with all antibiotics, bacterial resistance to β-lactams quickly arose after their discovery.

This resistance is mediated by a β-lactam specific hydrolase, or a β-lactamase, of which there are several types and various classification schemes. The simplest classification was implemented by Ambler in 1980, in which β-lactamases were grouped based on protein sequence (33). His scheme divides the β-lactamases into four classes, Ambler Classes A through D. Classes A, C, and D are serine-mediated β-lactamases (34). The class B β-lactamases are metallo-β-lactamases (MBLs) and require a zinc ion to assist in the hydrolysis reaction (35).

Figure 3 (adapted from original (36)) depicts how a serine β-lactamase catalyzes the hydrolysis of a β-lactam. First, a serine residue of the β-lactamase will attack the carbonyl, pushing electrons of the double bond onto the oxygen (Step 1). This leaves a highly unstable
TABLE 1: The four classes of β-lactam antibiotics and examples of each.

<table>
<thead>
<tr>
<th>β-lactam Antimicrobials</th>
<th>Penicillin</th>
<th>Penicillin G</th>
<th>Penicillin V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penicillins</td>
<td>Aminopenicillin</td>
<td>Ampicillin</td>
<td>Amoxicillin</td>
</tr>
<tr>
<td></td>
<td>Carboxypenicillin</td>
<td>Carbenicillin</td>
<td>Ticarcillin</td>
</tr>
<tr>
<td></td>
<td>Penicillinase-resistant penicillin</td>
<td>Methicillin</td>
<td>Nafcillin</td>
</tr>
<tr>
<td>Monobactams</td>
<td>Aztreonam</td>
<td>Cephalothin</td>
<td>Cephalexin</td>
</tr>
<tr>
<td></td>
<td>1st Generation</td>
<td>Cefamandole</td>
<td>Cefaclor</td>
</tr>
<tr>
<td></td>
<td>2nd Generation</td>
<td>Cefoxitin</td>
<td>Cefotetan</td>
</tr>
<tr>
<td>Cephalosporins</td>
<td>3rd Generation</td>
<td>Ceftriaxone</td>
<td>Ceftazidime</td>
</tr>
<tr>
<td></td>
<td>4th Generation</td>
<td>Cefotaxime</td>
<td>Ceftazolinem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cefepime</td>
<td>Cefpirome</td>
</tr>
<tr>
<td>Carbapenems</td>
<td>Imipenem</td>
<td>Doripenem</td>
<td>Ertapenem</td>
</tr>
<tr>
<td></td>
<td>Meropenem</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 2: The four classes of β-lactam antibiotics. The core β-lactam is highlighted in red. Each derivative within each class will deviate in the R groups. When hydrolyzed, the bond between the nitrogen and the carbonyl carbon is cleaved. A few examples are given.
negatively charged oxygen. To reduce this strain, the electrons collapse back down, and orbital resonance dissipates the energy across the molecule, opening the β-lactam ring in the process (Step 2). Then a two-step proton transfer from water to the serine residue of the β-lactamase resolves the hydrolysis reaction (Steps 3-5).

The most commonly encountered β-lactamases are the class A β-lactamases TEM (named for the patient in which it was first identified, Temoniera) and SHV (sulphydryl variable) types, with 90% of the ampicillin resistance encountered in *E. coli* mediated by TEM-1 (18). And due to their widespread prevalence, β-lactamase inhibitors have been developed. These inhibitors do not affect the activity of penicillin binding protein, and administration of these in conjunction with a β-lactam can kill the bacteria. However, these inhibitors are only effective against the serine mediated β-lactamases, exhibiting no effect on the class B MBLs due to their use of zinc ions in the hydrolysis reaction (37, 38).

*Extended-Spectrum β-lactamases* – One cause of multi-drug resistant bacteria is due to the emergence of extended-spectrum β-lactamases (ESBLs). Many of these ESBLs are from the Ambler Class A and are derivatives of TEM and SHV type β-lactamases (2, 33). More specifically, ESBLs are classified as oxyimino-cephalosporinases (2, 23), and are able to hydrolyze penicillin as well as cephalosporins such as ceftazidime and cefepime (Table 1). Many of these ESBLs have arisen due to only a few point mutations in either TEM or SHV β-lactamases (2, 39, 40). These point mutations alter the active site of the β-lactamase enough to accept a diverse range of β-lactams. However, they have also been shown to increase their susceptibility to β-lactamase inhibitors such as clavulanate; but of course, additional mutations can make them resistant (2, 38, 41). Figure 4 shows an example of the mutations that lead from TEM-1 to several ESBL TEM variants (2).
FIGURE 3: β-lactamase mediated hydrolysis of a cephalosporin.
A serine residue of the β-lactamase will attack the carbonyl, pushing electrons of the double bond onto the oxygen (Step 1). This leaves a highly unstable negatively charged oxygen. To reduce this strain, the electrons collapse back down, and orbital resonance dissipates the energy across the molecule, opening the β-lactam ring in the process (Step 2). Then, a two-step proton transfer from water to the serine residue of the β-lactamase resolves the hydrolysis reaction (Steps 3-5). [36] (Adapted from original.)
FIGURE 4: ESBL variants of TEM-1.
The common mutations among ESBL variants of TEM-1 are listed. The amino acid numbering is according the conventions set forth by Ambler. *TEM-2 is not an ESBL, but several ESBLs are derivates of TEM-2. **TEM-50 and TEM-68 are resistant to β-lactamase inhibitors. [2]
Carbapenems and Their Hydrolases

In another effort to avoid the havoc wrought by β-lactamases on the efficacy of these antibiotics, an additional class of β-lactams (carbapenems) was discovered. Because resistance to carbapenems is very infrequent, they are used as a last resort for treating infections to avoid the development of resistance. However, resistance to carbapenems developed anyway. Currently there are about nine diverse types of carbapenemases falling into Ambler Classes A, B, and D (42, 43). Each of those nine types have several variations. We will focus on four clinically relevant types found in Enterobacteriaceae, the Class A serine-mediated Klebsiella pneumoniae carbapenemase (KPC), and the three Class B metallo-β-lactamases (MBL): The New Delhi MBL (NDM), the Verona integron-encoded MBL (VIM), and the Imipenem resistant Pseudomonas-type MBL (IMP).

Klebsiella pneumoniae carbapenemase – First identified in 2001 (44), KPC was not the first carbapenemase, as several MBLs that could hydrolyze carbapenem had already been identified in Japan in the 1990’s (45). This initial variant (KPC-1) provided resistance to many of the β-lactams, including all the cephalosporins and aztreonam, and was also resistant to the β-lactamase inhibitors clavulanic acid and tazobactam (44). A recent review indicates that there are currently 12 reported variants of the KPC enzyme (46). While KPC may not be the first carbapenemase identified, it is the most common in the United States. As of February 27, 2018 the Centers for Disease Control and Prevention (CDC) report that KPC positive infections have been reported from all 50 states and the District of Columbia (47). KPC enzymes have also been reported from many other nations and in numerous gram-negative species, including Acinetobacter baumanii, Pseudomonas aeruginosa, and nearly all the Enterobacteriaceae (48-50).
**New-Delhi metallo-β-lactamase** – Originally isolated from India in 2008, there are currently more than ten reported variants of NDM (51). It is present in 34 states (47) and multiple countries including the United Kingdom, Pakistan, India, Sweden and others (50). The NDM carbapenemases have shown greater affinities for the penicillins, cephalosporins, and a few of the carbapenems than the VIM and IMP carbapenemases (43). Additionally, a minimum inhibitory concentration assay when NDM is cloned into susceptible strains show it conferring high levels of resistance to penicillins (>256 μg/mL), cephalosporins (>256 μg/mL), the monobactam aztreonam (>24 μg/mL), and all of the carbapenems (>16 μg/mL) (43), but other reports indicate that NDM cannot hydrolyze aztreonam and, as a MBL, it is resistant to β-lactamase inhibitors (52).

**Verona integron-encoded metallo-β-lactamase** – VIM has 14 reported variants with amino acid content varying up to 10% (51). VIM originated from *Pseudomonas aeruginosa* in the Mediterranean in 1997, but quickly spread into *Enterobacteriaceae* and proceeded to spread globally. Reports indicate that VIM can hydrolyze all β-lactams except monobactams and, as an MBL, it is resistant to β-lactamase inhibitors like clavulanate as tazobactam (53). Like the other carbapenemases, plasmids are the primary mechanism for horizontal gene transfer of this carbapenemase. According to the CDC, only 11 states have reported VIM positive infections (47).

**Imipenem resistant Pseudomonas metallo-β-lactamase** – IMP shares many of the same characteristics as VIM, but the amino acid content between the two diverges by 70% (51). IMP also represents the most diverse type of carbapenemase with 18 variants reported (51). Isolated in 1991 in Japan from *Pseudomonas*, it is the earliest carbapenemase discovered of the four, and is resistant to the inhibitor clavulanic acid (54). Currently, IMP has been found in many of the
enteric organisms, including *Serratia, Providencia, and Klebsiella*. As of February 2018, 13 states have reported IMP positive infections (47). As with many of the other carbapenemases, IMP has the ability to hydrolyze many of the β-lactams, but it cannot hydrolyze the monobactams (55).

Summary

While it is evident that much has been reported on the carbapenemases themselves, there is a distinct lack of published papers characterizing the diversity of plasmids that carry one of these four carbapenemases. Additionally, many of the reviews cited here mention that these carbapenemase-resistance plasmids carry multiple β-lactamases, but the relationship and interplay between the β-lactamases on a single plasmid is not well understood. The following two chapters will clarify these two points.
CHAPTER 1
Characterization of Carbapenemase-Resistance Plasmids

Galen E. Card, Brandon D. Pickett, Perry G. Ridge, Richard A. Robison

ABSTRACT

Carbapenem-resistant bacteria have quickly become a critical concern in nosocomial infections. In treating these infections, a rapid diagnosis is crucial. Current practices may take up to 76 hours, by which time the infection may become systemic, and the mortality rate is near 50%. To aid in carbapenemase understanding and detection, this report characterizes the gene content and replicon types of 446 carbapenemase-carrying plasmids available in GenBank and identifies the six most prevalent replicon types among these plasmids. The importance of this work is twofold: First, there is no published work that characterizes the plasmids that carry some of the most threatening antibiotic resistance genes, the carbapenemases. Having this information available can aid in knowing where efforts need to be placed to complete our understanding of these plasmids. Second, it highlights challenges that must be overcome if we are to adequately diagnose and restrict the spread of these plasmids.

Key words: Plasmid, Antimicrobial resistance, Carbapenemase, Enterobacteriaceae

BACKGROUND

Nosocomial infections have quickly become a significant cause of mortality. In 2002, the US Centers for Disease Control and Prevention estimated that the national mortality rate due to hospital acquired infections was 5.8% (56). In 2011, that rate increased to 10.4% (57). While these same reports show that the chance of acquiring an infection at the hospital has decreased, the infections are becoming more lethal.
One significant reason for this increase in mortality is the acquisition of antibiotic resistance in bacterial populations (25). Bacterial strains such as the carbapenem resistant Enterobacteriaceae (CREs), and multi-drug resistant Pseudomonas aeruginosa present diagnostic challenges which in turn lead to poor prognoses. Treatment of these bacterial infections usually begins with the administration of standard antibiotic regimens. The ineffectiveness of the initial treatment is usually apparent within 24-48 hours. At this time, the physician needs to reevaluate, order additional antibiotic susceptibility tests, and administer a more advanced antibiotic regimen. This new treatment may include carbapenem antibiotics. For resistant Enterobacteriaceae and Pseudomonas, this is an additional, ineffective 24-48-hour period before it is apparent that the patient’s condition is not improving. At this time, about 76 hours after initial diagnosis, the infection may have become systemic. Once a CRE infection has become systemic, the mortality rate is near 50% (58).

Antibacterial resistance is usually conferred to these organisms through mobile genetic elements, predominately extra-chromosomal DNA called plasmids (25). Plasmids often carry the molecular machinery to replicate themselves. This machinery allows for the transfer of the plasmid between different bacterial strains, and sometimes between any gram-negative bacteria (24). Furthermore, the antibiotic resistance genes on the plasmid can be located within a transposable element. This transposable element has the potential to replicate and integrate itself into new DNA sites, increasing the rate of spread (59). Additionally, many carbapenemase-carrying plasmids are large; therefore, they often carry a toxin/antitoxin plasmid addiction system to prevent the bacterium from losing the plasmid (31).

To assist in the identification and treatment of drug resistant infections, a better understanding of these carbapenemase carrying plasmids is needed. This brief report is the first
large-scale attempt to characterize the diversity of plasmids carrying carbapenemases from the *Klebsiella pneumoniae*-producing carbapenemase (KPC), the New-Delhi metallo-β-lactamase (NDM), the Verona-integron encoded metallo-β-lactamase (VIM), and the IMP-type metallo-β-lactamase (IMP) families in seven clinically-relevant gram-negative bacteria (*Enterobacter aerogenes, Enterobacter cloacae, Escherichia coli, Klebsiella pneumoniae, Pseudomonas aeruginosa, Providencia stuartii, and Serratia marcescens*).

**METHODS**

**Sequence acquisition**

532 complete plasmid sequences were obtained from GenBank by a discontiguous megablast nucleotide search (60) of four representative carbapenemase genes (IMP, KPC, NDM, VIM, Supplementary File 1) to allow for variations within the carbapenemase family. We employed the same Entrez strategy to filter for complete plasmids as used by Orlek et al. (61):


This blast search was done separately for the seven organisms of interest: *Enterobacter aerogenes, Enterobacter cloacae, Escherichia coli, Klebsiella pneumoniae, Pseudomonas aeruginosa, Providencia stuartii, and Serratia marcescens*. GenBank files were downloaded for each blast alignment that scored >80% identity and query coverage. These sequences were retrieved on 5 March 2018.
Plasmid gene composition

A list of key terms was derived by a manual survey of 10% of the acquired GenBank files, with cross reference to QuickGO, the European Bioinformatics Institute’s Gene Ontology reference database (62), to classify gene products into one of the following categories: (a) Antimicrobial resistance, with β-lactamases as a subset, (b) Plasmid transfer genes, (c) Toxin/antitoxin systems, (d) DNA maintenance, modifying, and repair proteins, (e) Mobile genetic elements, (f) Hypothetical genes, and (g) other. See Appendix A for the list of key terms.

Incompatibility group/Replicon typing and plasmid characterization

Plasmid incompatibility groups were determined by nucleotide BLAST (60, 63) against a local download of the PlasmidFinder v1.3 Enterobacteriaceae database (64) downloaded on 1 March 2018. The incompatibility groups were assigned when matches met the following criteria: (a) >=80% identity, (b) >=60% subject coverage, and (c) within 1% of the percent identity of the highest match. Accordingly, more than one incompatibility group could be reported for any given plasmid. Further characterization was accomplished as follows: (a) extracting the CDS regions for each plasmid, (b) searching these CDS regions for key terms using regular expressions, and (c) combining the results for plasmid groups of interest (e.g., those that belong to Enterobacteriaceae). Please see Appendix B for a more detailed description. This characterization of each plasmid and of groups of plasmids was accomplished using custom scripts, made freely available at https://github.com/ridgelab/plasmidCharacterization.

Statistical analyses

Since plasmid length distributions are not normal (left-skewed), all statistical analyses were performed with the Mann-Whitney U-test or the Kruskal-Wallis ranked ANOVA where
appropriate, for non-parametric distributions. In an effort to be conservative, statistical
significance was determined if P<0.0001.

RESULTS

Plasmid gene composition

Due to the inherent inconsistencies of GenBank record annotations, our search method
required discarding 86/532 accessions, leaving a total of 446 accessions in this analysis
(accession numbers available in Appendix C). The criteria for keeping an accession in the
analysis was if at least one and no more than six carbapenemase genes were identified on the
plasmid. Of those 446 plasmids, 198 carry KPC, 168 carry NDM, 49 carry IMP, and 31 carry
VIM. When divided by species, 7 belong to E. aerogenes, 33 to E. cloacae, 142 to E. coli, 235 to
K. pneumoniae, 18 to P. aeruginosa, 3 to P. stuartii, and 8 to S. marcescens. The mean size of all
carbapenemase-carrying plasmids was 104,222 bp, with a median length of 87,663 bp. The
largest plasmid was 500,840 bp and the smallest, 1,635 bp. The average percent gene content of
all plasmids was as follows: Antimicrobial resistance genes, 8.0%; Plasmid transfer genes,
15.8%; DNA modification genes, 14.7%; Mobile genetic elements, 9.3%; Hypothetical genes,
33.2%; Other/Metabolic genes, 18.9% (Figure 4A). The plasmids carried, on average, ~2 β-
lactamases, with 22.6% of the plasmids carrying three or more, and the most β-lactamases on a
single plasmid being six. The carbapenemase copy number of these plasmids ranged from 1-3,
with 97.98% of the plasmids harboring only one carbapenemase.

When comparing certain plasmid features such as the presence or absence of plasmid
addiction systems (236/446 or 52.9% of plasmids contain one), polymerase genes, or the family
of carbapenemase on the plasmid to plasmid length, the average length of plasmids that carry
addiction systems and polymerases are significantly larger than those that do not (Mann-Whitney
FIGURE 5: Characteristics of carbapenemase-encoding plasmids.
A) Mean percent gene content of all plasmids. B) Relationship between characteristics of interest and plasmid length. C) Relationship between species and plasmid length. D) Relationship between incompatibility group and plasmid length. Significance is determined against the average size of all plasmids. B-C) Mann-Whitney U-test, D) Kruskal-Wallis ranked ANOVA. **** P<0.0001. All error bars indicate the 95% CI.
U-test, P<0.0001, Figure 4B) and the average length of plasmids that carry KPC are smaller than those that carry IMP (Kruskal-Wallis ranked ANOVA, P<0.05, Figure 4B). However, removing an unusually large IMP plasmid (>500 kb) from this dataset eliminated this significance. When observing average plasmid length by species, the only near-significant difference was observed between *E. coli* and *K. pneumoniae*, with the latter being larger (P=0.0018, Figure 4C). It is important to note that these two species also represent most of the plasmids analyzed (377/446 or 84.5%).

Plasmid Incompatibility group/Replicon typing

No incompatibility group presented itself as the most abundant; however, the following six groups constitute 80.27% of the plasmids: IncA/C (53/446 or 11.88%), IncFIB (57/446 or 12.78%), IncFII (88/446 or 19.73%), IncN (61/446 or 13.68%), IncR (40/446 or 8.97%), and IncX3 (59/446 or 13.23%) (see Appendix D). Notably, 7.62% (34/446) of the plasmids could not be accurately typed using this method. 58 plasmids carried more than one replicon, and these were significantly larger than those that carried a single replicon (Mann-Whitney U-test P<0.0001, data not shown). Additionally, the following incompatibility groups were found to have an average length statistically different (Kruskal-Wallis ranked ANOVA, P<0.0001) than the average of all plasmids: IncA/C2, IncFIB, IncHI1B, and IncX3 (Figure 4D). Figure 5 shows the relative abundance of each incompatibility group among plasmids that carry the same family of carbapenemase (full dataset available, Appendix E).

DISCUSSION

One of the most notable findings of this study was the amount of hypothetical and uncharacterized genes found on these plasmids. It is possible that many of these genes may be phage derived. This is of great concern when considering phage therapy as an alternative to
FIGURE 6: Relative abundance of incompatibility groups among plasmids. Predominant incompatibility groups from each carbapenemase family: KPC, IncFIB 18.2%, IncFII 20.2%, IncN 17.7%, and IncR 13.6%; NDM, IncA/C2 17.9%, IncFIB 10.7%, IncFII 28.0%, and IncX3 29.2%; IMP, IncA/C2 22.4%, IncL/M 10.2%, IncN 34.7%, and NA 16.3%; VIM, IncA/C2 16.1%, IncN 12.9%, IncR 12.9%, and NA 35.5%.

Note: Percent totals are larger than 100% because some plasmids have multiple replicon types.
antibiotics. With potentially large regions of homology to phage genomes, a phage may incorporate into these plasmids through homologous recombination. Then, as the recombinant phage genome is packaged into the progeny phage, it may be possible that carbapenemases could be included, resulting in a replication-deficient phage vector capable of transferring a carbapenemase to a new bacterium through transduction. Before phage therapy of these organisms is seriously pursued, this concept should be investigated so that another mechanism of resistance transfer is not potentiated, as it has been shown for phage and other virulence genes (65).

Additionally, for non-amplification, DNA-based identification of carbapenemase production, it is important to realize that the plasmids of interest are quite large. With their median length over 80 kb, plasmid isolation becomes difficult if necessary for the application, and many of the replicon types identified are for low-copy number plasmids. This also compounds the difficulties when detecting carbapenemase gene presence from a whole-blood specimen, where concentrations are around 10 CFU/mL. This results in approximately 10 copies of an ~800 bp gene that needs to be identified amongst the millions of base pairs in a milliliter of blood.

Finally, this report has identified a few potential targets to slow the spread of carbapenemase plasmids. First, the antitoxin of the plasmid addiction system could be targeted (31). Doing so could prevent its binding with the toxin, resulting in the death of the host harboring the plasmid. Second, 90.4% (403/446) of the plasmids carry transfer genes to pass the plasmid between bacteria. Preventing pilus formation could dramatically reduce the spread of these plasmids. This method is currently being pursued by several groups and include strategies such as bacteriophage, colloidal clays, and antibodies (66). Finally, many of the plasmids carry a
plasmid partitioning system, responsible for ensuring that each daughter receives a copy of the plasmid. Targeting the motor or the partition-site binding protein of these systems, in conjunction with the toxin/antitoxin system, could dramatically reduce the spread and persistence of these plasmids in the hospital. These treatments could be used in a sterilization bath for medical equipment prior to traditional sterilization techniques.

In conclusion, there is an abundance of data for the commonly encountered KPC and NDM carbapenemases from *K. pneumoniae* and *E. coli*, and several non-traditional avenues that may be pursued to help stop the spread of these resistance plasmids. However, this report is lacking data for many of the other species, and for the VIM and IMP carbapenemases. Therefore, a greater surveillance of the other species and carbapenemases is needed. *P. aeruginosa* is a bacterium where much more data is needed to accurately characterize the diversity of carbapenemase-carrying plasmids in this highly significant pathogen.

Funding

This work was supported by the U.S. National Institutes of Health (R01 AI116989).

Acknowledgements

GEC conceptualized this analysis, determined the functional groups of interest, generated the key terms, and analyzed the output from the scripts. BDP wrote the scripts for the analysis and assisted in writing a portion of the manuscript. PGR and RAR advised this work and reviewed the manuscript. We thank the Fulton Supercomputing Laboratory ([https://marylou.byu.edu](https://marylou.byu.edu)) at Brigham Young University for their consistent efforts to support our research.

Conflicts of Interest

The authors have no conflicts of interest to declare.
CHAPTER 2

β-lactamase Diversity of a Single, Carbapenem-Resistant Enterobacteriaceae Isolate

Galen E. Card, Joel S. Griffitts Ph.D., Joshua D. Urquiaga, Richard A. Robison Ph.D.

ABSTRACT

Antibiotic resistance is quickly becoming an urgent problem in health care. One class of antibiotics, the β-lactams, has become severely compromised by emerging resistance. Resistance to last-resort β-lactams (the carbapenems) is quickly spreading across the globe. We investigated a carbapenem-resistant isolate of Klebsiella pneumoniae possessing six β-lactamase genes: CMY-6, CTX-M-15, NDM-4, OXA-1, SHV-11, and TEM-1. Each of these genes was functionally characterized in Escherichia coli using 5 β-lactam antibiotics (ampicillin, carbenicillin, ceftazidime, aztreonam, and imipenem). These tests revealed distinct as well as overlapping functions. Most notably, we observed that the carbapenemase NDM-4 provides a greater fitness advantage when grown in a cephalosporin than the cephalosporinase CTX-M-15. Also, we provide evidence that a sizable portion of the resistance that this strain of Klebsiella exhibits against aztreonam and imipenem is not enzyme mediated.

INTRODUCTION

To date, there are about 20 different derivates of β-lactam antibiotics approved for therapeutic use. Many of these antibiotics have been put on the World Health Organization’s ‘WATCH GROUP’, due to the higher potential for resistance among bacterial populations [1]. β-lactam antibiotics fall into four broad classes; penicillins, cephalosporins, monobactams, and carbapenems, each class representing several clinically important structural derivatives (Fig. 1). With such a wide variety of β-lactams available, most infections can be treated effectively with
these antibiotics. However, the mortality rate from nosocomial infections has been on the rise [2, 3] due to the increased incidence of multi-drug resistant infections [4, 5].

Many genera of the Enterobacteriaceae family have recently joined this class of multi-drug resistant bacteria [6, 7] as they have acquired genes encoding extended-spectrum β-lactamases (ESBL) and carbapenemases (carbapenem resistant Enterobacteriaceae, CRE). Septic infections with ESBL or CRE strains of Klebsiella pneumoniae have a mortality rate of near 50% [5].

We have in our collection various CRE isolates possessing 1-6 β-lactamase (bla) genes. In this work, we characterize one of these isolates (Klebsiella pneumoniae strain 1300761), a CRE isolate from which we have identified six distinct bla genes: CMY-6, CTX-M-15, NDM-4, OXA-1, SHV-11 and TEM-1. Of these, five belong to the Ambler class A or C, designated as serine mediated hydrolases, with the exception being NDM, which belongs to class B, the metallo-β-lactamases which require zinc ions in the active site to catalyze the reaction [8]. Three of them are recognized as ESBLs (CMY-6, CTX-M-15, and NDM-4,) and three are narrow-spectrum β-lactamases (OXA-1, SHV-11, and TEM-1). While hydrolytic activity of these bla genes is well documented through MICs, the in vivo fitness advantage provided by them by analyzing growth kinetics is not. In this study, we use a standardized susceptible E. coli strain to test resistance conferred by each of these bla genes, in response to challenge by five different β-lactam antibiotics. Our observations shed light on the relative contributions of each gene and contribute to our understanding of how multi-gene β-lactamase arsenals may function along with alternative resistance mechanisms to provide strong β-lactam resistance in CRE strains.

METHODS

Genome sequencing, assembly, and annotation
Genome sequencing and read processing

A carbapenem resistant *Klebsiella pneumoniae* isolate was obtained from the Centers for Disease Control and Prevention (*K. pneumoniae* 1300761) and DNA was extracted following the recommended protocol for the MagNA Pure LC system (Roche Life Sciences). DNA was quantified by fluorometry and 2μg was submitted to the BYU DNA Sequencing Center for 250bp paired end reads on an Illumina HiSeq 2500. Low complexity reads were filtered using PRINSEQ version 0.20.4 [9] and adapter sequence removal and quality trimming was accomplished using Trim Galore! version 0.4.3 with a phred score cutoff of 28. An additional 10bp were trimmed from the 5’ end of each read. All reads shorter than 150 bp were then discarded and if their paired read was longer than 150 bp and passed the other quality checks, they were retained as a singleton for use in assembly. Read quality statistics were then assessed using FastQC version 0.11.4 [10].

Genome assembly and annotation

The reads were assembled using the St. Petersburg Assembler (SPAdes) version 3.10.1 [11]. K-mer values of 21, 33, 55, 77, 99, 129 were used for the assembly iterations. Assembly statistics were compiled using QUAST version 4.0 [12]. Gene annotation was undertaken using Prokka version 1.12 [13].

β-lactamase cloning

TABLE 2 contains primer pairs used to introduce restriction sites, a synthetic ribosome binding site, and amplify the corresponding β-lactamases from *K. pneumoniae* 1300761. Each *bla* gene was individually cloned into pJG780 with XbaI/SalI restriction sites (plasmid sequence available in supplemental file 1) and transformed into NEB5-alpha (New England BioLabs), a DH5-alpha derivative, following the manufacturer’s provided protocol with a recovery
incubation of 90 minutes. The β-lactamase expression is under the control of a rhamnose-inducible promoter. Each clone was then sequence verified using the following plasmid specific sequencing primer: CTGTCAGTAACGAGAAGGTCG. The resulting strains were named using the following convention: Host vector \_bla (i.e. E. coli pJG780_CMY-6) and are referred to by the β-lactamase they produce (E. coli pJG780_CMY-6 is referred to as CMY-6)

Growth curve analysis

A single colony of the β-lactamase clones were grown in 5 ml of LB containing 30 μg/mL of kanamycin to ensure plasmid retention for 12-18 hours. Then, 100 μL was inoculated in a 5 mL 1-hour subculture containing 30 μg/mL kanamycin and 0.3% rhamnose to induce β-lactamase expression. Microtiter plates (96-well) were loaded with 190 μL of the selective media (30 μg/mL kanamycin, 0.3% rhamnose, appropriate β-lactamase) and inoculated with 10 μL of the 1-hour subculture. Preliminary results (data not shown) indicated that the β-lactamase clone growth curves should be performed at the following concentrations: ampicillin (16 μg/mL), carbenicillin (16 μg/mL), ceftazidime (8 μg/mL), aztreonam (8 μg/mL), and imipenem (2 μg/mL). These concentrations are half of the concentration used to determine antibiotic resistance by the Clinical Laboratory Standards Institute (74). This is relevant since approximately 80 μL of media was lost to evaporation over the course of the growth curve. The plates were then incubated in a BioTek Synergy HT Microplate Reader at 37 °C. OD\textsubscript{600} readings were taken after a brief shaking every 30 minutes over a 10.5-hour growth period. The growth curves were also performed on the parent strain (K. pneumoniae 1300761) in LB with the previously indicated antibiotics. Each growth curve was measured in duplicate, and the experiment was repeated three times.
<table>
<thead>
<tr>
<th><strong>bla</strong> gene</th>
<th><strong>Forward Primer</strong></th>
<th><strong>Reverse Primer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CMY-6</td>
<td><strong>cagctctagaggag</strong>GATTTCATGATGA</td>
<td>cagcgctcgacGCCTCATCGTCAGTT</td>
</tr>
<tr>
<td></td>
<td>AAAAATCGTTATGCTGC</td>
<td>ATTGCAGC</td>
</tr>
<tr>
<td>CTX-M-15</td>
<td><strong>cagctctagaggag</strong>AATCCCATGGTT</td>
<td>cagcgctcgaCGCTATTACAAACCGT</td>
</tr>
<tr>
<td></td>
<td>AAAAAATCACTGC</td>
<td>CGGTG</td>
</tr>
<tr>
<td>NDM-4</td>
<td><strong>cagctctagaggag</strong>AACTTGATGGAA</td>
<td>cagcgctcgcAGCCCATGGCTCA</td>
</tr>
<tr>
<td></td>
<td>TTGCCCAATATTATG</td>
<td>GCGC</td>
</tr>
<tr>
<td>OXA-1</td>
<td><strong>cagctctagaggag</strong>CTTATTATGAAA</td>
<td>cagcgctcgacGGGTTGGCGGATTT</td>
</tr>
<tr>
<td></td>
<td>AACACAATACATATCAAACCTCGC</td>
<td>GCCATTAG</td>
</tr>
<tr>
<td>SHV-11</td>
<td><strong>cagctctagaggag</strong>GTGTTATGCGTT</td>
<td>cagcgctcgacGGGTTAGCCTGATGCA</td>
</tr>
<tr>
<td></td>
<td>ATATTCGCCCTGTGT</td>
<td>G</td>
</tr>
<tr>
<td>TEM-1</td>
<td><strong>cagctctagaggag</strong>AAGAGTATGAGT</td>
<td>cagcgctcgacTCTGGTCTGACAGT</td>
</tr>
<tr>
<td></td>
<td>ATTCAACATTTCGTGTC</td>
<td>CCAATGCTTTAAC</td>
</tr>
</tbody>
</table>

Restriction site **Ribosome binding site**
Statistical analysis

A two-way ANOVA in conjunction with Fisher’s Least Significant Difference test was used to compare all time points against the control. Significance was determined if P<0.05. When a β-lactamase clone reached absorbance levels significantly different than the control before another in the same antibiotic, it was determined that the clone that reached significant absorbance levels first is more efficient at hydrolyzing that β-lactam. When comparing growth curves of a single β-lactamase in different growth conditions, if the growth curve in one antibiotic reached significant absorbance levels at an earlier time point than another, it was determined that that β-lactamase was more efficient at hydrolyzing the β-lactam that allowed quicker growth.

RESULTS

*Klebsiella pneumoniae* 1300761 possesses six β-lactamase-encoding genes

Illumina sequencing produced 3,653,470 paired reads, and read processing reduced that number to 3,370,288 with 80,534 retained as singletons, for approximately 150X coverage of the genome. The average quality score of all reads is greater than 37, with an average length of 215. Assembly generated 196 contigs larger than 1,000 bp, with an N50 of 238,732 and N75 of 112,412. The genome length is 5,972,622 bp. Annotation predicted 5,733 genes. Notably, 6 distinct *bla* genes were detected (Supplemental File 2) and confirmed by BLAST search [15] as the following: *bla*\_CMY-6, *bla*\_CTX-M-15, *bla*\_NDM-4, *bla*\_OXA-1, *bla*\_SHV-11, and *bla*\_TEM-1. These *bla* genes were then cloned into NEB5-α for further characterization.

Resistance profiles of β-lactamase clones

The control group in these Figs 2A-F is the pJG780 empty vector. When grown in nutrient rich, non-selective LB, the only significant results were CTX-M-15, NDM-4 and OXA-
1; which had a significantly lower carrying capacity at 630 minutes (Fig 2A). When growth occurred in LB ampicillin, all β-lactamases except CMY-6 reached growth levels significantly different than the control by 300 minutes and by 630 minutes, CMY-6 also reached significant levels (Fig 2B). In LB carbenicillin, all β-lactamases were at significant levels by 300 minutes, and CMY-6 after 630 minutes (Fig 2C). In LB ceftazidime, NDM growth was evident by 300 minutes and CTX-M-15 by 630 minutes (Fig 2D). CTX-M-15 was the only β-lactamase that hydrolyzed aztreonam, and significant levels were reached by 630 minutes (Fig 2E). NDM-4 was the only β-lactamase that hydrolyzed imipenem, with growth observed by 300 minutes, but not reaching significant levels until 630 minutes (Fig 2F). Complete growth curves are reported in Fig S1, raw data in Table S1.

β-lactam preferences for a single β-lactamase

In these comparisons, the growth curve for pJG780 grown in ampicillin was used as a negative control (Figs 3A-G). As expected, pJG780 only grew in LB, and attained growth levels significantly different from the negative control by 300 minutes (Fig 3A). CMY-6 was able to grow in ampicillin, and carbenicillin, however, this clone grew slowly in ampicillin, reaching growth levels different than the control by 630 minutes (Fig 3B). CTX-M-15 grew in all antibiotics except imipenem and growth was observed in ceftazidime by 630 minutes, but not yet above background levels. By 300 minutes growth was evident in ampicillin and carbenicillin and by 630 minutes for aztreonam (Fig 3C). NDM-4 grew in all antibiotics, except aztreonam. Ampicillin, carbenicillin and ceftazidime reached significance by 300 minutes and imipenem by 630 minutes. (Fig 3D). OXA-1, SHV-11, and TEM-1 only grew in ampicillin and carbenicillin, reaching significant levels in each case by 300 minutes (Fig 3E-H). Complete growth curves can be found in Fig S2, raw data in Table S1.
FIGURE 7: Growth curves grouped by growth conditions.
A) Nonrestrictive LB broth. B) Ampicillin. C) Carbenicillin. D) Ceftazidime. E) Aztreonam. F) Imipenem. *P<0.05 **P<0.01, ***P<0.001, ****P<0.0001.
FIGURE 8: Growth curves of the β-lactamase clones.
The clones were grown for 630 minutes, and the OD600 was measured every 30 minutes. A) The pJG780 empty vector control strain. B) The CMY-6 clone. C) The CTX-M-15 clone. D) The NDM-4 clone. E) The OXA-1 clone. F) The SHV-11 clone. G) The TEM-1 clone. H) The parent strain *K. pneumoniae* 1300761. *P<0.05 **P<0.01, ***P<0.001, ****P<0.0001.
Growth of the parent strain *K. pneumoniae* 1300761 is not inhibited by any β-lactam tested.

The two-way ANOVA indicated that antibiotic has no significant impact on growth, even though individual time points may have randomly showed significance from the LB control (i.e. imipenem at 630 minutes) (Fig 3H). Additionally, the lack of a significant lag phase when this isolate was introduced to antibiotic-containing media, indicated that these genes are constitutively expressed in *K. pneumoniae* 1300761.

**DISCUSSION**

The antibiotic crisis is reaching a crescendo as nosocomial bacteria acquire resistance to the common, and last resort, antibiotics. Furthermore, it appears that individually, these *bla* genes may not pose a large threat, as the fitness provided is relatively weak (i.e. imipenem, ceftazidime, aztreonam). But there appears to be a synergistic effect as they are combined, and that the additional resistance factors *Klebsiella* possesses (truncated porins, multi-drug efflux pumps) aid greatly in its resistance. At the concentrations assayed, it seems that NDM-4 is superior to CTX-M-15 as a cephalosporinase. Interestingly, it also appears that the cephalosporinase CTX-M-15 provides a greater fitness advantage to aztreonam (a monobactam) than ceftazidime (a cephalosporin). Finally, CMY-6 provides a better degree of fitness when grown in ampicillin then carbenicillin. *Tbla*his result is interesting, and several *bla*CMY genes have been shown to hydrolyze cephalosporins at aztreonam at high concentrations (76, 77).

Another path that could shed greater light on fitness provided by these various β-lactamases would be to perform competition assays between them in the various antibiotics. Additionally, this strategy could also help characterize the differences between the various carbapenemases.
In conclusion, this study helps us understand that these organisms may acquire redundant genes for the synergism acquired. This synergism provides the parent strain enhanced fitness in antibiotics such as aztreonam and imipenem, where only one of the antibiotics hydrolyzes it, but the parent strain grows normally. Additionally, this disparity in growth curve statistics may also indicate that the passive resistance mechanisms (i.e. efflux pumps, altered porins) play a more substantial role in resistance than previously thought, and they should receive more concentrated attention.

ACKNOWLEDGEMENTS

GEC was responsible for designing and performing experiments, assembling and annotating the genome, and drawing conclusions and he is the primary author of this manuscript. JSG helped design the molecular cloning methodology and assisted in the molecular cloning work. JDU helped perform experiments. RAR secured funding, and experiments were performed in his laboratory. We would like to express our thanks to the BYU Fulton Supercomputing Laboratory (https://marylou.byu.edu), the BYU Research Instrumentation Core (https://ricfacility.byu.edu), and the BYU DNA Sequencing Center (https://dnasc.byu.edu) for their consistent efforts to support our research. This work was funded by the U.S. National Institutes of Health (R01 AI116989). The authors have no conflicts of interest to declare.
SUMMARY

This thesis provides only a starting point for further investigation. Much is still needed to fully characterize these plasmids, and a greater surveillance of carbapenem-resistance plasmids is needed to create a more comprehensive picture. Furthermore, several points are identified in chapter 1 that can be exploited by small molecule inhibition to limit or eliminate the spread of these plasmids. Additionally, there is a wealth of data available for the *Escherichia coli* and *Klebsiella pneumoniae* carbapenem resistant isolates. This data could be mined for numerous points of interest and conclusions made.

Second, the assay set up in chapter two could be used to assess the level of fitness provided by the various carbapenemases. Additionally, as mention in chapter 2, competition assays between these strains would also help determine if a fitness advantage is provided by the β-lactamase activity, or if metabolic costs of producing the β-lactamase are a detriment to fitness.
REFERENCES


APPENDIX A

TABLE 3: Key Words used to characterize CR-plasmid gene content.

<table>
<thead>
<tr>
<th>Categories</th>
<th>Key Word</th>
<th>Python Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antimicrobial Resistance</td>
<td>aac</td>
<td>aac</td>
</tr>
<tr>
<td></td>
<td>aad</td>
<td>aad</td>
</tr>
<tr>
<td></td>
<td>aminoglyco*</td>
<td>aminoglyco[^s]</td>
</tr>
<tr>
<td></td>
<td>aph</td>
<td>aph</td>
</tr>
<tr>
<td></td>
<td>arr-</td>
<td>arr-</td>
</tr>
<tr>
<td></td>
<td>arsa</td>
<td>ars[a-dhr]</td>
</tr>
<tr>
<td></td>
<td>arsb</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>arsc</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>arsd</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>arsh</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>arsr</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>arsen*</td>
<td>arsen[^s]</td>
</tr>
<tr>
<td></td>
<td>bleomycin</td>
<td>bleomycin</td>
</tr>
<tr>
<td></td>
<td>catr</td>
<td>catr</td>
</tr>
<tr>
<td></td>
<td>chloramphenicol</td>
<td>chloramphenicol</td>
</tr>
<tr>
<td></td>
<td>cmea</td>
<td>cme[abc]</td>
</tr>
<tr>
<td></td>
<td>cmeb</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>cmec</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
resistance
rifampin
silver
streptomycin
sul

teller*
Antimicrobial Resistance, Beta-lactamase

<table>
<thead>
<tr>
<th>Term</th>
<th>Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>tera</td>
<td>ter[abcfw-z][^a-z]</td>
</tr>
<tr>
<td>terb</td>
<td>&quot;</td>
</tr>
<tr>
<td>terc</td>
<td>&quot;</td>
</tr>
<tr>
<td>terf</td>
<td>&quot;</td>
</tr>
<tr>
<td>terw</td>
<td>&quot;</td>
</tr>
<tr>
<td>terx</td>
<td>&quot;</td>
</tr>
<tr>
<td>tery</td>
<td>&quot;</td>
</tr>
<tr>
<td>terz</td>
<td>&quot;</td>
</tr>
<tr>
<td>tetr</td>
<td>tetr[^[a][^acycline]]</td>
</tr>
<tr>
<td>ampr</td>
<td>(?:^[^p])ampr</td>
</tr>
<tr>
<td>beta-lactam*</td>
<td>beta[^-]lactam[^s]</td>
</tr>
<tr>
<td>beta-lactam*</td>
<td>&quot;</td>
</tr>
<tr>
<td>bla</td>
<td>(?:^[^p])bla</td>
</tr>
<tr>
<td>cephalosporin*</td>
<td>cephalosporin[^s]</td>
</tr>
<tr>
<td>cmy-</td>
<td>(?:^[^p])cmy-</td>
</tr>
<tr>
<td>ctx-</td>
<td>(?:^[^p])ctx-</td>
</tr>
<tr>
<td>dha-</td>
<td>(?:^[^p])dha-</td>
</tr>
<tr>
<td>oxa-</td>
<td>(?:^[^p])oxa-</td>
</tr>
<tr>
<td>oxacillin*</td>
<td>oxacillin[^s]</td>
</tr>
<tr>
<td>penicillin*</td>
<td>penicillin[^s]</td>
</tr>
<tr>
<td>Antimicrobial Resistance, Beta-Lactamase, Beta-Lactamase Special</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>sfo-</td>
<td>(?:[^\p{P}])sfo-</td>
</tr>
<tr>
<td>shv-</td>
<td>(?:[^\p{P}])shv-</td>
</tr>
<tr>
<td>tem-</td>
<td>(?:[^\p{P}])tem-</td>
</tr>
<tr>
<td>carbapenem*</td>
<td>carbapenem[^s]</td>
</tr>
<tr>
<td>imp not (impa or impb or impc)</td>
<td>(?:^[^b-z])imp(?:$[^abc])</td>
</tr>
<tr>
<td>kpc</td>
<td>(?:^[^b-z])kpc</td>
</tr>
<tr>
<td>ndm</td>
<td>(?:^[^b-z])ndm</td>
</tr>
<tr>
<td>vim</td>
<td>(?:^[^b-z])vim</td>
</tr>
<tr>
<td>conjuga*</td>
<td>conjuga[^s]</td>
</tr>
<tr>
<td>fertility inhibition</td>
<td>fertility inhibition</td>
</tr>
<tr>
<td>fino</td>
<td>fino</td>
</tr>
<tr>
<td>icm*</td>
<td>icm[^s]</td>
</tr>
<tr>
<td>moba</td>
<td>mob[a-e]</td>
</tr>
<tr>
<td>mobb</td>
<td>&quot;</td>
</tr>
<tr>
<td>mobbc</td>
<td>&quot;</td>
</tr>
<tr>
<td>mobd</td>
<td>&quot;</td>
</tr>
<tr>
<td>mobfe</td>
<td>&quot;</td>
</tr>
<tr>
<td>pili*</td>
<td>pili[^s]</td>
</tr>
<tr>
<td>pilus</td>
<td>pilus</td>
</tr>
<tr>
<td>pilx</td>
<td>pilx</td>
</tr>
<tr>
<td>secretion system</td>
<td>secretion system</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>tivb*</td>
<td>tivb[^\s]</td>
</tr>
<tr>
<td>traa</td>
<td>tr[a-rtuwx](?:$</td>
</tr>
<tr>
<td>trab</td>
<td>&quot;</td>
</tr>
<tr>
<td>trac</td>
<td>&quot;</td>
</tr>
<tr>
<td>trad</td>
<td>&quot;</td>
</tr>
<tr>
<td>trae</td>
<td>&quot;</td>
</tr>
<tr>
<td>traf</td>
<td>&quot;</td>
</tr>
<tr>
<td>trag</td>
<td>&quot;</td>
</tr>
<tr>
<td>trah</td>
<td>&quot;</td>
</tr>
<tr>
<td>trai</td>
<td>&quot;</td>
</tr>
<tr>
<td>traj</td>
<td>&quot;</td>
</tr>
<tr>
<td>trak</td>
<td>&quot;</td>
</tr>
<tr>
<td>tral</td>
<td>&quot;</td>
</tr>
<tr>
<td>tram</td>
<td>&quot;</td>
</tr>
<tr>
<td>tran</td>
<td>&quot;</td>
</tr>
<tr>
<td>trao</td>
<td>&quot;</td>
</tr>
<tr>
<td>trap</td>
<td>&quot;</td>
</tr>
<tr>
<td>traq</td>
<td>&quot;</td>
</tr>
<tr>
<td>trar</td>
<td>&quot;</td>
</tr>
<tr>
<td>Toxin/Antitoxin System</td>
<td>&quot;</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>abrb</td>
<td>abrb</td>
</tr>
<tr>
<td>cbta</td>
<td>cbta</td>
</tr>
<tr>
<td>ccd[ab]</td>
<td>ccd[ab]</td>
</tr>
</tbody>
</table>

vir* not virulence

vir[^ugo\s]
<table>
<thead>
<tr>
<th>DNA Maintenance/Modification</th>
<th>chromosome</th>
<th>chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>dna</td>
<td>dna</td>
<td>dna</td>
</tr>
<tr>
<td>eex</td>
<td>eex</td>
<td>eex</td>
</tr>
<tr>
<td>entry exclusion</td>
<td>entry exclusion</td>
<td>entry exclusion</td>
</tr>
<tr>
<td>exca</td>
<td>exca</td>
<td>exca</td>
</tr>
<tr>
<td>toxi* (?:^[^a-z]</td>
<td>anti)toxi[^s]</td>
<td></td>
</tr>
<tr>
<td>yafo</td>
<td>yafo</td>
<td>yafo</td>
</tr>
</tbody>
</table>
helicase
integrase
kfra
kora
korb
methylase
nucleotide
para
parb
plasmid
recombinase (not serine or tyrosine recombinase)
relaxase
repa
replication
replication protein
ruma
single-strand binding protein
ssb
topb
topoisomerase
<table>
<thead>
<tr>
<th>DNA Maintenance/Modification</th>
<th>DNA Maintenance/Modification Special</th>
</tr>
</thead>
<tbody>
<tr>
<td>trfa</td>
<td>trfa</td>
</tr>
<tr>
<td>uvr*</td>
<td>uvr[^\s]</td>
</tr>
<tr>
<td>vagc</td>
<td>vag[cd]</td>
</tr>
<tr>
<td>vagd</td>
<td>&quot;</td>
</tr>
<tr>
<td>muca</td>
<td>muc[ab]</td>
</tr>
<tr>
<td>mucb</td>
<td>&quot;</td>
</tr>
<tr>
<td>polymerase</td>
<td>polymerase</td>
</tr>
<tr>
<td>umuc</td>
<td>umu[cd]</td>
</tr>
<tr>
<td>umud</td>
<td>&quot;</td>
</tr>
<tr>
<td>ista</td>
<td>ist[ab][?:$][^a-z0-9])</td>
</tr>
<tr>
<td>istb</td>
<td>&quot;</td>
</tr>
<tr>
<td>resolvase</td>
<td>resolvase</td>
</tr>
<tr>
<td>reverse transcriptase</td>
<td>reverse transcriptase</td>
</tr>
<tr>
<td>tnp</td>
<td>tnp</td>
</tr>
<tr>
<td>transpos*</td>
<td>transpos[^\s]</td>
</tr>
<tr>
<td>urf2</td>
<td>urf2</td>
</tr>
<tr>
<td>domain containing</td>
<td>domain[-]containing</td>
</tr>
<tr>
<td>domain-containing</td>
<td>&quot;</td>
</tr>
<tr>
<td>hypothetical</td>
<td>hypothetical</td>
</tr>
<tr>
<td>uncharacterized protein</td>
<td>uncharacterized protein</td>
</tr>
<tr>
<td>unknown function</td>
<td>unknown function</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>disrupted</td>
<td>disrupted</td>
</tr>
<tr>
<td>imperfect</td>
<td>imperfect</td>
</tr>
<tr>
<td>interrupted</td>
<td>interrupted</td>
</tr>
<tr>
<td>intron</td>
<td>intron</td>
</tr>
<tr>
<td>is(?:[a-z]{2})[0-9]{2,4}</td>
<td>is(?:[a-z]{2})[0-9]{2,4}</td>
</tr>
<tr>
<td>kl.pn.i3</td>
<td>kl.pn,i3</td>
</tr>
<tr>
<td>morpho</td>
<td>morpho</td>
</tr>
<tr>
<td>ncrna</td>
<td>ncrna</td>
</tr>
<tr>
<td>non functional</td>
<td>non-[ ]?functional</td>
</tr>
<tr>
<td>non-functional</td>
<td>&quot;</td>
</tr>
<tr>
<td>partial</td>
<td>partial</td>
</tr>
<tr>
<td>patho</td>
<td>patho</td>
</tr>
<tr>
<td>repeat region</td>
<td>repeat region</td>
</tr>
<tr>
<td>se.ma.*</td>
<td>se,ma,[s]</td>
</tr>
<tr>
<td>truncated</td>
<td>truncated</td>
</tr>
</tbody>
</table>
APPENDIX B

Supplementary Bioinformatics Methods

This is a more detailed explanation of the bioinformatics methods required for incompatibility group/replicon typing and plasmid characterization. This will describe a step-by-step walkthrough of the process. Please note that most of these steps will be simple data formatting. Also note that it would have been easier in some cases to combine multiple steps into one. The choice to separate each piece of the process was for clarity and to enable another to modify this process for their own purposes. For our work, all steps could be run interactively; i.e., not requiring a high-performance computing (HPC) architecture. Our work was completed on a machine running Red Hat Enterprise Linux.

Summary

This process begins with one fasta file and multiple GenBank files. The formats for these files are described in steps 0 and 2, respectively. The fasta file contains the incompatibility group sequences. In our work, this was a download of the PlasmidFinder v1.3 Enterobacteriaceae database (64). The GenBank files contain one or more GenBank records in them, where each record could itself be considered a GenBank file for a single accession number. Thus, these GenBank files are concatenations of multiple GenBank records. Effectively, this is how we grouped accessions of interest. The same accession may appear in multiple groupings. Note, if you attempt to re-use our process with your own data and have GenBank files as a single file per accession, combining them into groups will feel unnecessary. We began this way because that is what we had to start with.

The results of the entire process are CSV files with information about each plasmid in a group and a text file with summary statistics about each group. The file contains basic
information (e.g., plasmid length), the incompatibility group(s) the plasmid best aligns to, and some gene/function annotation based on key term searches of the GenBank file's CDS regions.

To accomplish this, each (input) group GenBank file is split into a single GenBank file per accession and the sequences are extracted as fasta files. The sequence lengths are recorded and these sequences are individually aligned (using the NCBI BLAST+ Suite (60, 63)) to the incompatibility group sequences. After filtering out the "best" alignments, the incompatibility group is determined and saved for later assimilation into the final outputs. The CDS regions are extracted from the GenBank files and searched for key terms using regular expressions. Each key term belongs to one or more categories. Matches in each category are counted and summarized in the final output. For more details on this searching strategy, please see step #11. The key terms are listed with their Python regular expression in Appendix A.

This summary concludes with an outline of the steps. Each step will be detailed, followed by the references. The code in the detailed steps has, in many cases, been simplified. In other cases, the code is several pages long and would be difficult to copy and paste effectively. Especially with Python code, readability suffers as lines wrap because a standard page is not wide enough to contain some code statements on a single line. Accordingly, we encourage you to visit the online repository for the code: https://github.com/ridgelab/plasmidCharacterization.

Outline of Steps

Step 0. Format Incompatibility Groups Fasta File

Step 1. Create Incompatibility Groups BLAST database

Step 2. Split Multi-Accession GenBank Files

Step 3. Extract ORIGIN Sequence from GB to Fasta

Step 4. Extract Group Lists
Step 0. Format Incompatibility Groups Fasta File

Input: Fasta file with incompatibility group sequences. Each sequence may be on one or
more lines. The headers might start with “Inc”.

Output: Same fasta file as the input, but sequences occur on only one line. Headers without
“Inc” now have “Inc” prepended.

Code:

Bash Command

```
awk -f formatIncGroupFasta.awk \\
    original_incomp-grp.fasta \\
    > incomp-grp.fasta
```

AWK Script (formatIncGroupFasta.awk)

```
#!/bin/awk -f
{
    if ( $0 ~ /^>.+$/ ) {
```
if ( NR != 1 ) {
    printf "\n";
}

if ( $0 ~ /^>Inc.+$/ ) {
    print $0;
} else {
    printf "%s%s\n", ">Inc", substr($0, 2);
}
}
else {
    printf "%s", $0;
}

END {
    printf "\n";
}

Step 1. Create Incompatibility Groups BLAST database

Input: Fasta file with incompatibility group sequences. Each sequence is on only one line.

The headers start with “>Inc”.

Output: BLAST database of the incompatibility group sequences.

Code:

Bash Command

```
makeblastdb \\
    -dbtype nucl \\
    -in incompatibility.fasta \\
    -input_type fasta \\
    -title incompatibility \\
    -parse_seqids \\
    -hash_index \\
    -out incompatibility \\
    -max_file_sz 2GB \\
    -logfile makeBlastDB.log
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (60, 63).
Step 2. Split Multi-Accession GenBank Files

Input: 1+ GenBank files, each with 1+ records. Each record is itself a GenBank file for a single Accession. Thus, the multi-accession GenBank files are simply concatenations of multiple single-accession GenBank files. Assume that these GenBank files are in a directory called `original_gb`.

Output: One GenBank file for each accession. If the same accession exists in more than one multi-accession file, assume they are the same and overwrite it. Assume that the output GenBank files will be in a directory called `plasmid_gb`.

Code:

Bash Command

```bash
cd plasmid_gb
while read ifn
do
    awk -f splitMultiGB.awk "${ifn}"
done < <(ls -l original_gb/*.gb)
```

AWK Script (splitMultiGB.awk)

```awk
#! /bin/awk -f
BEGIN {
    FS="[ ]+";
    accession="";
    ofn="";
}
{
    if ($0 == "//" || $0 == "")
    {
        accession = "";
        ofn = "";
    }
    else if ($1 == "LOCUS")
    {
        accession = $2;
        ofn = accession ".gb";
        print $0 > ofn;
    }
```
Step 3. Extract ORIGIN Sequence from GB to Fasta

Input: One GenBank file with a single accession in it. Assume it is in the directory `plasmid_gb` and it is named after the pattern `${ACCESSION}.gb`.

Output: One Fasta file with the sequence from the ORIGIN section of the GenBank file. The Fasta file has sequences that are each on only one line. It will be in the directory `plasmid_fasta`.

Code:

Bash Command

```bash
while read ifn
do
    ACCESSION=`basename "${ifn}" ".gb"`
    awk -f extractOriginSeqFromGBtoFasta.awk \ "plasmid_gb/${ACCESSION}.gb" \ > "plasmid_fasta/${ACCESSION}.fasta"
done < <(ls -l plasmid_gb/*.gb)
```

AWK Script (extractOriginSeqFromGBtoFasta.awk)

```bash
#! /bin/awk -f
BEGIN {
FS = "[[]]+";
    origin_found = 0; # false
}
{
```
if (origin_found)
{
    sub(/ *[0-9]+ /, "", $0);
    gsub(/ +/, "", $0);
    printf toupper($0);
}
else if ($1 == "ORIGIN")
{
    origin_found = 1; # true
    print ">" gensub(/^(.+).gb$/, \1, ";-1", gensub(/^.*$/,
"", ";-1", FILENAME));
}
}

END {
printf \"\n\n\nprint "done extracting ORIGIN seq from " FILENAME " to fasta" > "/dev/stderr";
}

Step 4. Extract Group Lists

Input: One GenBank file with a multiple accessions in it. Assume it is in the directory

original_gb and it is named after the pattern \${GROUP}.gb.

Output: Multiple text files, each with the extension ".list". Each file is a line separated list of accessibility numbers that make up the group. The files will be in a directory called groups with the name \${GROUP}.list.

Code:

Bash Command

while read ifn
do
    awk -f extractGroupLists.awk \
    "$ifn"
done < <(ls -l original_gb/*.gb)

AWK Script (extractGroupLists.awk)

#!/bin/awk -f
BEGIN {

Step 5. Blast Incompatibility Groups

Input: Fasta files. Each contains the sequence from a single accession. Assume they are in the directory `plasmid_fasta` and they are named after the pattern `${ACCESSION}.fasta`.

Input: The incompatibility groups BLAST database created in step #1. It is named `incompatibility`.

Output: One tab-separated value file for each input file. Each file is a modified version of the BLAST output format 6. The format is specified as seen using the -outfmt option with blastn. The columns are as follows: qseqid, sseqid, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, qseq, and sseq. The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c.tsv`. Note that a match was not included in the output if the percent identity was <80%.

Code:

Bash Command

```
THREADS=8

while read ifn
do
  ACCESSION=`basename "${ifn}" ".fasta``
  blastn \
    -query "${ifn}" \ 
    -strand both \ 
    -task blastn \ 
    -db incompatibility \ 
    -out blast_results/${ACCESSION}_fmt6c.tsv \ 
    -outfmt "6 qseqid sseqid pident length evalue qframe qlen qstart qend sframe slen sstart send scov qseq sseq" \ 
    -num_threads ${THREADS} \ 
    -perc_identity 80
  qlen qstart qend sframe slen sstart send qseq sseq"
  done < <(ls -1 plasmid_fasta/*.fasta)
```

BLAST Software

NCBI (United States National Center for Biotechnology Information) BLAST+ Suite version 2.4.0 (60, 63).

Step 6. Subset BLAST Results by Coverage Cutoff of 60%

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. Assume they are in the directory `blast_results` and they are named after the pattern `$_fmt6c.tsv`.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except some results may be omitted if the coverage was less than 60%. The files will be in a directory called `blast_results` and named after the pattern `$_fmt6c_cov60.tsv`. Note that a new column was inserted as column number 14 (1-based indexing). The columns will now be as follows: qseqid, sseqid, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.
Code:

Bash Command

```bash
while read ifn
do
    ACCESSION=`basename "${ifn}" _fmt6c.tsv`
    awk -f subCovCutoff60.awk "${ifn}"
    > "blast_results/${ACCESSION}_fmt6c_cov60.tsv"
done < <(ls -1 blast_results/*_fmt6c.tsv)
```

AWK Script (subCovCutoff60.awk)

```bash
#!/bin/awk -f

BEGIN {
    FS="\t";
    OFS="\t";
    ORS="\n";
    count=0;
}

{ 
    # 4 = length, 11 = slen, scov = length / slen
    scov = $4 / $11;
    if (scov >= 0.6) {
        count += 1
        # keep 1-13, add new column, keep 14-15 (will become 15-16)
        for (i = 1; i <= 13; i++)
            printf "%s", $i OFS;
        printf "%f", scov OFS;
        for (i = 14; i <= NF; i++)
            printf "%s", $i (i == NF ? ORS : OFS);
    }
}

END {
    print FILENAME ": " count > "/dev/stderr";
}
```

Step 7. Add Incompatibility Group as Column to BLAST Results

Input: Tab-separated value files. Each contains the results from blasting the sequence of a
single accession against the incompatibility groups BLAST database. It has an added column with the subject coverage and has only records with coverage >60%. Assume they are in the directory `blast_results` and they are named after the pattern `${ACCESSION}_fmt6c_cov60.tsv`.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except that an additional column is added. This column has the family or root of the incompatibility group from column #2 (sseqid). The files will be in a directory called `blast_results` and named after the pattern `${ACCESSION}_fmt6c_cov60_fam.tsv`. Note that a new column was inserted as column number 3 (1-based indexing). The columns will now be as follows: qseqid, sseqid, fam, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.

Code:

Bash Command

```bash
while read ifn
do
    ACCESSION=`basename "${ifn}" "_fmt6c_cov60.tsv"`
    awk -f addFamCol.awk "${ifn}" > "blast_results/${ACCESSION}_fmt6c_cov60_fam.tsv"
done < <(ls -1 blast_results/*_fmt6c_cov60.tsv)
```

AWK Script (addFamCol.awk)

```awk
#! /bin/awk -f

BEGIN {
    FS="\t";
    OFS="\t";
    ORS="\n";
}

{ # 2 = subject_id, keep 1-2, add new column, keep 3-16 (will become 4-

```
Step 8. Filter Best Matches in BLAST Results

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. It has two added columns with the subject coverage (and has only records with coverage >60%) and family. Assume they are in the directory blast_results and are named after the pattern

${ACCESSION}_fmt6c_cov60_fam.tsv.

Output: One tab-separated value file for each input file. Each file is a copy of its respective input file except that some results are omitted. The “best” results are retained. “Best” is defined as the result(s) with the highest percent identity and those that have percent identities within only 1 percent of the highest one. The files will be in a directory called

blast_results and named after the pattern ${ACCESSION}_fmt6c_cov60_fam_best.tsv.

As in the input file, the columns will be as follows: qseqid, sseqid, fam, pident, length, evalue, qframe, qlen, qstart, qend, sframe, slen, sstart, send, scov, qseq, and sseq.

Code:

Bash Command

```bash
while read ifn
do
```

ACCESSION=`basename "${ifn}" "_fmt6c_cov60_fam.tsv"`

python3 filterBestResults.py \ 
"${ifn}" \ 
> "blast_results/${ACCESSION}_fmt6c_cov60_fam_best.tsv"
done <<(ls -l blast_results/*_fmt6c_cov60_fam.tsv)

Python Version


Python Script (filterBestResults.py)

def handleArgs():
    if len(sys.argv) != 3:
        sys.stderr.write("ERROR: You must provide 2
arguments:
input blast results cov60 fam
output blast results file")
        sys.exit(1)

    input_br = sys.argv[1]
    output_br = sys.argv[2]

    return input_br, output_br

# MAIN

if __name__ == "__main__":
    import sys

    ibrfn, obrfn= handleArgs()

    # set some handy vars
    records = [] # each line
    per_ids = [] # percent identities (the 4th column)

    with open(ibrfn, 'r') as ifd:
        for line in ifd:
            records.append(line)
            per_ids.append(float(line.rstrip('
').split('	')[3]))

    # figure out which ones to keep
    keep = []

    max_per_id = max(per_ids) if len(records) > 0 else 0.0

    for i,per_id in enumerate(per_ids):
        if abs(max_per_id - per_id) <= 1.0:
            keep.append(i)
Step 9. Extract Incompatibility Families

Input: Tab-separated value files. Each contains the results from blasting the sequence of a single accession against the incompatibility groups BLAST database. It has two added columns with the subject coverage (and has only records with coverage >60%) and family. Only the “best” results remain. Assume they are in the directory `blast_results` and are named after the pattern `{ACCESSION}_fmt6c_cov60_fam_best.tsv`.

Output: One file for each input file. Each file is a line-delimited list of incompatibility group roots/families. The files will be in a directory called `blast_results` and named after the pattern `{ACCESSION}_families.list`.

Code:

Bash Command

```
while read ifn
do
    ACCESSION=`basename "${ifn}"_fmt6c_cov60_fam_best.tsv`
    cut -f 3 "${ifn}" | sort | uniq > blast_results/"${ACCESSION}_families.list"
done < <(ls -1 blast_results/*_fmt6c_cov60_fam_best.tsv)
```

Step 10. Extract Plasmid Search Regions

Input: This Python program requires 3 inputs. 1- The accession number of the plasmid it will
extract the search regions from. 2- The directory where the output will be placed. 3- The directory where the GenBank file is located for that plasmid. We assume the GenBank file is named after the pattern ${ACCESSION}.gb.

Output: One text file containing the lines from input GenBank file that will be searched using the key terms. We assume the output file will be named after the following pattern: 

${ACCESSION}_searchRegions.txt. For convenience, it will also generate a copy of the input GenBank file with shell color codes, marking the CDS regions in blue, the portions of the CDS regions that will be included in green, and the portion of the CDS regions that will not be searched in red. This file will have the same name as the .txt file, but will have the extension .gb instead of .txt. Note that intended search space is to consider each CDS region as a separate entity. However, only the following subsections of each CDS region are to be considered: \function, \gene, \note, and \product.

Code:

Bash Command

```
while read ifn
do
    ACCESSION=`basename "${ifn}" ".gb"`
    python3 extractPlasmidSearchRegions.py "${ACCESSION}" "plasmid_searchRegions" 
    plasmid_gb
done < <(ls -l plasmid_gb/*gb)
```

Python Version


Python Script (extractPlasmidSearchRegions.py)
# ======== #
# FUNCTIONS #
# ======== #

def handleArgs():
    import sys

    if len(sys.argv) != 4:
        sys.stderr.write("ERROR: You must provide 3
arguments
1- plasmid accession
2- output search regions
dir
3- input gb dir")
        sys.exit(1)

    plasmid_accession = sys.argv[1]
    output_search_regions_dir = sys.argv[2].rstrip('/'
    input_gb_dir = sys.argv[3].rstrip('/'

    return plasmid_accession, output_search_regions_dir, input_gb_dir

def parseGbFile(input_gb_fn, output_search_regions_fn, output_gb_fn):
    with open(output_search_regions_fn, 'w') as osrd:
        with open(output_gb_fn, 'w') as ogbd:
            red = "\033[0;31m"
            green = "\033[0;32m"
            blue = "\033[0;34m"
            no_color = "\033[0m"

            with open(input_gb_fn, 'r') as ifd:
                subsection_names_of_interest = ("function", "gene", "note", "product")

                # skip from LOCUS to FEATURES
                line = ifd.readline()  # grab the first line
                while line.rstrip('n').lstrip('').split(' ')[0] != "FEATURES":
                    ogbd.write(line)
                    line = ifd.readline()

                # write then skip past FEATURES
                ogbd.write(line)
                line = ifd.readline()

                # write then skip past FEATURES
                ogbd.write(line)
                line = ifd.readline()

                # skip any lines necessary until CDS or ORIGIN is found
                tag_word = line.rstrip('n').lstrip(' ').split(' ')[0]
                while tag_word != "ORIGIN" and tag_word != "CDS":
ogbd.write(line)
line = ifd.readline()
tag_word = line.rstrip('\n').lstrip(' ')

# First time: found ORIGIN or CDS. If ORIGIN, we're done. If CDS, read through each CDS region, until ORIGIN.
# thereafter: found ORIGIN or CDS or other section name. If ORIGIN, we're done. If CDS, read through each CDS region, until ORIGIN. If section name, skip till ORIGIN or next CDS.
while tag_word != "ORIGIN":
  # first time: this loop will be skipped. Thereafter, if a section name (other than CDS), skip to next CDS or ORIGIN.
  while tag_word != "ORIGIN" and tag_word != "CDS":
    ogbd.write(line)
    line = ifd.readline()
tag_word = line.rstrip('\n').lstrip(' ').split(' ')[0]
  if tag_word == "CDS":
    # write the CDS line
    osrd.write(line)
o GBP.write(blue + line + no_color)
  else: # if tag_word == "ORIGIN":
    break
  # skip past the CDS line (guaranteed to now have a CDS line)
  line = ifd.readline()
tag_word = line.rstrip('\n').lstrip(' ')

# read through important data and stop at end of CDS (marked by next CDS or ORIGIN or other section name)
# first time: guaranteed inside a CDS region. Note that a CDS line is NEVER *immediately* followed by another section name line (at least in our data).
# thereafter: It could be anything between the CDS and ORIGIN.
while tag_word != "ORIGIN" and tag_word not in section_names:
  if line.rstrip('\n').lstrip(' ')[0] == '/': # it is a CDS subsection headerline
    subsection_name = line.strip().split('=')[0].lstrip('/').lower()
    subsection = 
    '.'.join(line.strip().split('="')[1:]).lower()
  if subsection_name in subsection_names_of_interest: # the subsection is one we care to look in
    osrd.write(line)
o GBP.write(green


if subsection[0] == '' and subsection[-1] != '': # the subsection spans multiple lines
    line = ifd.readline()
    tag_word = line.rstrip('
').lstrip(' ').split(' ')[0]
    while line.rstrip('
')[-1] != '': # keep searching to find the end of the subsection of interest
        osrd.write(line)
        ogbd.write(green + line + no_color)
        line = ifd.readline()
        tag_word = line.rstrip('
').lstrip(' ').split(' ')[0]
        osrd.write(line)
        ogbd.write(green + line + no_color)

    line = ifd.readline()
    tag_word = line.rstrip('
').lstrip(' ').split(' ')[0]
else: # the subsection is not one we care to look in
    #if
    #
    print("subsection len == 0")
    #
    print(line)
    #
    print(subsection_name)
    #
    print(subsection)
    # simple version that works, but doesn't write it all in red
    #ogbd.write(line)
    #line =
    #tag_word =
    #
    line = line.rstrip('
').lstrip(' ').split(' ')[0]

    version that actually makes it write it all in red
    ogbd.write(red +
    line + no_color)
    if
    len(subsection) and subsection[0] == '' and subsection[-1] != '': # the subsection spans multiple lines
ifd.readline()

line = line.rstrip('\n').lstrip(' ').split(' ')[0]

while line.rstrip('\n')[-1] != '"': # keep searching to find the end of the subsection of interest

ogbd.write(red + line + no_color)

line = ifd.readline()

tag_word = line.rstrip('\n').lstrip(' ').split(' ')[0]

ogbd.write(red + line + no_color)

line = ifd.readline()

else: # it is not a CDS subsection headerline

ogbd.write(line)

ogbd.write(red + line + no_color)

line = ifd.readline()

tag_word = line.rstrip('\n').lstrip(' ').split(' ')[0]

ogbd.write(line) # write the ORIGIN

ogbd.write(ifd.read()) # write the rest of the file (i.e., the sequence data)

# =====#
# MAIN  #
# =====#

if __name__ == "__main__":

import sys

# handle args
plasmid_accession, output_search_regions_dir, input_gb_dir = handleArgs() # set some helpful vars
osrn = output_search_regions_dir + '/\' + plasmid_accession + "\_searchRegions.txt"
ogeb = output_search_regions_dir + '/\' + plasmid_accession + "\_searchRegions.gb"
ignb = input_gb_dir + '/\' + plasmid_accession + ".gb"

# get CDS info (Antimicrobial Resistance CDS (%) ... Total CDS)
parseGbFile(ignb, osrn, ogbn)

# exit
sys.exit(0)
Step 11. Identify Plasmid Matches

Input: This Python program requires 3 inputs. 1- The accession number of the plasmid in which it will identify matches. 2- The directory where the input search regions file is located. 3- The directory where the output matches will be placed. We assume the input search regions file is named after the pattern ${ACCESSION}_searchRegions.txt.

Output: One tab-separated value file containing matches. We assume the output file will be named after the following pattern: ${ACCESSION}_matches.tsv. The columns of the file are as follows:

1. Ignored (True/False)
2. Categories (c1[,c2,…,cN])
3. Search Term
4. CDS Region

Column 1 is a simple flag denoting if the term was to be ignored. This could also be determined based on the second column, but it was convenient to have a simple flag as its own column. Column 2 contains the category (categories) that the search term belonged to. Column 3 contains the regular expression used. Column 4 contains the CDS region that was searched (all tabs and newlines were converted to \t (backslash and a t, not a tab) and \n (backslash and an n, not a newline) to not interfere with the tab-separated value file format and keep each record on a single line).

Search Strategy: The search terms are each part of one or more categories. It can belong to multiple categories only if the categories are subsets of each other. Five principal categories
exist, two of which have subcategories. The category structure is as follows:

- Antimicrobial Resistance
  - Beta-lactamase
    - Beta-lactamase Special
- Toxin/Antitoxin System
- DNA Maintenance/Modification
  - DNA Maintenance/Modification Special
- Mobile Genetic Elements
- Hypothetical Genes

The strategy could be described as top-to-bottom, in-to-out; i.e., Antimicrobial Resistance is more important that Toxin/Antitoxin System and Beta-lactamase Special is more important than Beta-lactamase and Antimicrobial Resistance. The reason these are shown nested instead of simply above their parents is because a match for a Beta-lactamase Special search term will increment the count for not only itself, but also its parents. If no matches are found, the CDS region being searched is classified as "Other". Some CDS regions will never be searched for these terms if they first match a term in a special "Ignored" category. Provided a CDS region is not to be ignored, it will be searched with Beta-lactamase Special terms, then Beta-lactamase terms, then Antimicrobial Resistance Terms, then Toxin/Antitoxin System terms, and so-forth, until a match is found (thus halting the search on this CDS region) or no more search terms remain (it is assigned to the "Other" category). All CDS regions are converted to lowercase before being searched as described. See Supplementary Table 1 for a table of search terms.
Code:

**Bash Command**

```bash
while read ifn
do
    ACCESSION=`basename "${ifn}" "_searchRegions.txt"`
    python3 identifyPlasmidMatches.py 
        "$ACCESSION" 
        plasmid_searchRegions 
        plasmid_matches
done < <(ls -l plasmid_searchRegions/*_searchRegions.txt)
```

**Python Version**


**Python Script (identifyPlasmidMatches.py)**

```python
# ========= #
# FUNCTIONS #
# ========= #

def handleArgs():
    import sys

    if len(sys.argv) != 4:
        sys.stderr.write("\n\nERROR: You must provide 3 arguments\n   1- plasmid accession\n   2- input search regions dir\n   3- output matches dir  \n\n")
        sys.exit(1)

    plasmid_accession = sys.argv[1]
    input_search_regions_dir = sys.argv[2].rstrip('/')
    output_matches_dir = sys.argv[3].rstrip('/')

    return plasmid_accession, input_search_regions_dir, output_matches_dir

def writeLineToMatchesFile(matches_fd, ignored, categories, search_term, cds_search_region):
    matches_fd.write(str(ignored)  + '	' + ','.join(categories) + '	' + search_term + '	' + convertCDSsearchRegionToOneLineStr(cds_search_region) + '
')

def ignoreCDS(cds_search_region, matches_fd):
    key_terms = [ r"truncated", r"interrupted", r"partial", r"disrupted", r"introns", r"kl\,pn\,i3", r"seq\,ma\,[s]\", r"morpho", r"repeat region", r"patho", r"ncrna", r"imperfect", r"non[ -]?functional", r"is\(?:[a-z]\{2\}\)0-9\{1,4\}" ]
```
def betaLactSpecialCopyNum(cds_search_region, matches_fd):
    key_terms = [ r"(?::|^)[^b-z]ndm", r"(?::|^)[^b-z]imp(?::|$[^abc])", r"(?::|^)[^b-z]vim", r"(?::|^)[^b-z]kpc", r"carbapenem[^s]"
    return searchCdsRegionForKeyTerms(cds_search_region, key_terms, matches_fd, True, ["Ignored"])

def betaLactSearch(cds_search_region, matches_fd):
    if not betaLactSpecialCopyNum(cds_search_region, matches_fd):
        return searchCdsRegionForKeyTerms(cds_search_region, key_terms, matches_fd, False, ["Antimicrobial Resistance", "Beta-lactamase", "Beta-lactamase Special"])
    else:
        return True

def antimicrobResistSearch(cds_search_region, matches_fd):
    if not betaLactSearch(cds_search_region, matches_fd):
        key_terms = [ r"aac", r"aad", r"aph", r"arr-", r"resistance", r"aminoglyco[^s]", r"streptomycin", r"chloramphenicol", r"cme[^abc]", r"catr", r"multidrug", r"efflux pump", r"mercur[^s]", r"teller[^s]", r"arsen[^s]", r"qace", r"macrolide", r"mph", r"silver", r"copper", r"flor", r"ter[^bceawz][^z]", r"fluoroquino[^s]", r"bleomycin",
        r"tetr(?:?::^[^a]acycline)", r"pco[^ers]", r"ars[^dhr]", r"sil[^acefprs]", r"(?:sulfonamide|trimethoprim|nickel)-[^resistant]", r"(?::|^)[^a-z]folp(?::|$[^a-z])", r"(?::|^)[^a-z]|[^sul[12]([^a-z])", r"(?::|^)[^a-z]|[^dfra(?::|$[^a-z])", r"(?::|^)[^a-y]ncr[^a-c,y](?::|$[^a-z])", r"(?::|^)[^a-z]|[^nirb(?::|$[^a-z])", r"rifamp(?:?::in[^cin])"
        return searchCdsRegionForKeyTerms(cds_search_region, key_terms, matches_fd, False, ["Antimicrobial Resistance"])
    else:
        return True

def plasmidTransferSearch(cds_search_region, matches_fd):
    key_terms = [ r"conjuga[^s]", r"pili[^s]", r"pilus", r"type[^iv]", r"secretion system", r"fertility inhibition", r"tivb[^s]", r"icm[^s]" ]
r"tra[a-rtuwxy](?::|$[^a-z])", r"trb[a-gilm]", r"mob[a-e]",
  r"fino",
  r"vir[^ugo\\s]", r"pilx"
]

return searchCdsRegionForKeyTerms(cds_search_region, key_terms,
matches_fd, False, ["Plasmid Transfer")]

def toxinSearch(cds_search_region, matches_fd):
  key_terms = [ r"(?::|[^a-z]\d|^a-z]anti)toxi[^\s]", r"stb[de]", r"hig[ab]",
    r"cbta",
    r"rel[be]", r"yha", r"yafo", r"ccd[ab]",
    r"abrb", r"par[de]", r"pem[ik]", r"hokg"
]

return searchCdsRegionForKeyTerms(cds_search_region, key_terms,
matches_fd, False, ["Toxin System")]

def dnaMaintSpecialCopyNum(cds_search_region, matches_fd):
  key_terms = [ r"muc[ab]", "umu[cd]", "polymerase"
]

return searchCdsRegionForKeyTerms(cds_search_region, key_terms,
matches_fd, False, ["DNA Maintenance", "DNA Maintenance Special")]

def dnaMaintSearch(cds_search_region, matches_fd):
  if not dnaMaintSpecialCopyNum(cds_search_region, matches_fd):
    key_terms = [ r"methylase", r"single-strand binding protein",
      r"ssb", r"topb",
      r"replication protein", r"kfra", r"kor[ab]", r"trfa",
      r"helicase", r"dna", r"chromosome", r"entry exclusion",
      r"ex", r"exca", r"nucleoti[^s]", r"topoisomerase",
      r"integrase", r"(?<!ser_|ine )recombinase",
      r"replication", r"nuclease",
      r"relaxase", r"plasmid", r"ruma", r"repa",
      r"uvr[^s]", r"par[ab]", r"vag[cd]"
]

return searchCdsRegionForKeyTerms(cds_search_region,
key_terms, matches_fd, False, ["DNA Maintenance")]
else:
  return True

def mobileGeneticElementsSearch(cds_search_region, matches_fd):
  key_terms = [ r"transpos[^\s]", r"reverse transcriptase", r"tnp",
    r"ist[ab]($|^a-z-9)\", r"resolvase", "urf2"
]

return searchCdsRegionForKeyTerms(cds_search_region, key_terms,
matches_fd, False, ["Mobile Genetic Elements")]

def hypotheticalGenesSearch(cds_search_region, matches_fd):
  key_terms = [ r"hypothetical", r"domain[ -]containing",
    r"uncharacterized protein", r"unknown function"
]

return searchCdsRegionForKeyTerms(cds_search_region, key_terms,
matches_fd, False, ["Hypothetical Genes")]

def convertCDSsearchRegionToOneLineStr(cds_search_region):
  return "\\n".join(list(map(lambda x: x.replace('\\t', "\\t"),
"\\n".replace('\\n', "\\"), cds_search_region)))
def searchCDSRegion(cds_search_region, matches_fd):
    # make cds_search_region all lowercase
    cds_search_region = list(map(lambda x: x.lower(), cds_search_region))
    # make all the search regions lowercase
    if not ignoreCDS(cds_search_region, matches_fd):
        if not antimicrobResistSearch(cds_search_region, matches_fd):
            if not plasmidTransferSearch(cds_search_region, matches_fd):
                if not toxinSearch(cds_search_region, matches_fd):
                    if not dnaMaintSearch(cds_search_region, matches_fd):
                        if not mobileGeneticElementsSearch(cds_search_region, matches_fd):
                            if not hypotheticalGenesSearch(cds_search_region, matches_fd):
                                writeLineToMatchesFile(matches_fd, False, ['Other'], 'NA', cds_search_region)
    return True

def searchCdsRegionForKeyTerms(cds_search_region, key_terms, matches_fd, ignored, categories):
    import re
    for search_sub_region in cds_search_region:
        for key_term in key_terms:
            if re.search(key_term, search_sub_region) is not None:
                writeLineToMatchesFile(matches_fd, ignored, categories, key_term, cds_search_region)
    return True

def parseSearchRegionFile(input_sr_fn, matches_fn):
    import re
    with open(input_sr_fn, 'r') as ifd:
        with open(matches_fn, 'w') as mfd:
            mfd.write("Ignored (True/False)\tCategories\tSearch Term\tCDS Region\n")
            cds_search_region = []
            # All data is important. Read through all CDS regions separately and search through them.
            # Each CDS region begins with CDS and ends with another CDS record or the end of file
            for search_sub_region in cds_search_region:
                if re.search(key_term, search_sub_region) is not None:
                    writeLineToMatchesFile(matches_fd, ignored, categories, key_term, cds_search_region)
            return True

            return False
Step 12. Generate Plasmid CSVs

Input: This Python program requires 5 inputs. 1- The accession number of the plasmid it will generate a CSV file for. 2- The directory where the output CSV file is to be placed. 3- The directory where the plasmid fasta file is located. We assume it is named after the pattern `${ACCESSION}.fasta`. 4- The directory where the input plasmid matches file is located. We assume it is named after the pattern `${ACCESSION}_matches.tsv`. 5- The directory where
the input incompatibility groups (derived from the BLAST results) are located. We assume it is named after the pattern `${ACCESSION}_families.list`.

Output: One comma-separated value file. It will be placed in the directory specified in the input position 2. We assume the output file will be named after the following pattern: `${ACCESSION}.csv`. The columns of the file are as follows:


Code:
Bash Command

```bash
while read ifn
do
  ACCESSION=`basename "${ifn}" *.fasta`
  python3 generatePlasmidCSV.py 
  "$ACCESSION" 
  plasmid_csv 
  plasmid.fasta 
  plasmid_matches 
  blast_results
done < <(ls -l plasmid.fasta/*.fasta)
```

Python Version


Python Script (generatePlasmidCSV.py)

```python
# ========= #
# FUNCTIONS #
# ========= #

def handleArgs():
  import sys

  if len(sys.argv) != 6:
    sys.stderr.write("ERROR: You must provide 5 \narguments\n1- plasmid accession\n2- output csv dir\n3- input fasta dir\n4- input matches dir\n5- input incompatibility groups blast output dir\n")
    sys.exit(1)

  plasmid_accession = sys.argv[1]
  output_csv_dir = sys.argv[2].rstrip('/')
  input_fasta_and_length_dir = sys.argv[3].rstrip('/')
  input_matches_dir = sys.argv[4].rstrip('/')
  input_incompatibility_groups_blast_output_dir = sys.argv[5].rstrip('/')

  return plasmid_accession, output_csv_dir, input_fasta_and_length_dir, input_matches_dir, input_incompatibility_groups_blast_output_dir

def CSVify(some_str):
  return '"' + some_str + '"'

def getPlasmidLength(input_length_fn):
  with open(input_length_fn, 'r') as ifd:
    return int(ifd.readline().rstrip('
'))

def getRegionCounts(categories):
  cats = [ "Antimicrobial Resistance", "Beta-lactamase", "Beta-lactamase...
counts = [0] * len(cats)

category_counts = {}

for category in sorted(categories):
    if not category in category_counts:
        category_counts[category] = 0
    category_counts[category] += 1

for i, cat in enumerate(cats):
    counts[i] = category_counts[cat] if cat in category_counts else 0

return counts

def updateCDScounts(cds_counts, cds_region_counts):
    for i, cds_region_count in enumerate(cds_region_counts):
        cds_counts[i] += cds_region_count

    return cds_counts

def parseMatchesFile(matches_fn):
    import re

    with open(matches_fn, 'r') as ifd:
        cds_counts = [0] * 10 # 10 CDS related columns in output

        # skip past the TSV header line
        ifd.readline()

        # grab first data line
        line = ifd.readline()

        while line != "" :
            fields = line.rstrip(\n').split(\t')

            ignore = True if fields[0] == "True" else False
            categories = fields[1].split(',')
            key_term = fields[2]
            cds_search_region = fields[3]

            if not ignore:
                cds_counts = updateCDScounts(cds_counts,
                                            getRegionCounts(categories))

            # grab the next line
            line = ifd.readline()

            tag_word = line.rstrip(\n').lstrip( '').split( '')[0]

        return cds_counts
def getPercentOfTotal(count, total):
    if total:
        return count / total
    else:
        return "NA"

def convertCdsInfoToOutputStr(antimicrob_resist_cds_count,
beta_lact_cds_count, beta_lact_special_copy_num,
plasmid_transfer_cds_count, \
    toxin_cds_count, dna_maint_cds_count, \
    dna_maint_special_copy_num, mobile_genetic_elements_cds_count, \
    hypothetical_genes_cds_count, other_cds_count):
    # initialize output list (will eventually become a giant string). Each
    # item will need to be easily converted to a string using str.
    output = []
    # find the total num of cds regions
    total_cds_count = sum((antimicrob_resist_cds_count, plasmid_transfer_cds_count, toxin_cds_count, \
                           dna_maint_cds_count, mobile_genetic_elements_cds_count, \
                           hypothetical_genes_cds_count, other_cds_count))
    # append columns to output
    # antimicrob resist (w/ beta lact)
    #  antimicrob resist
    output.append(antimicrob_resist_cds_count) # count
    output.append(getPercentOfTotal(antimicrob_resist_cds_count, total_cds_count)) # percent of total
    #  beta lact
    output.append(beta_lact_cds_count) # count
    output.append(getPercentOfTotal(beta_lact_cds_count, total_cds_count)) # percent of total
    output.append("Yes" if beta_lact_special_copy_num else "No") # present (Yes/No)
    # plasmid transfer
    output.append(plasmid_transfer_cds_count) # count
    output.append(getPercentOfTotal(plasmid_transfer_cds_count, total_cds_count)) # percent of total
    #  special copy num
    output.append(getPercentOfTotal(beta_lact_special_copy_num, beta_lact_cds_count)) # percent of beta lact
    output.append("Yes" if beta_lact_special_copy_num else "No") # present (Yes/No)
    # toxin system
    output.append(toxin_cds_count) # count
    output.append(getPercentOfTotal(toxin_cds_count, total_cds_count)) # percent of total
    output.append("Yes" if toxin_cds_count else "No") # present (Yes/No)
    # dna maint
    output.append(dna_maint_cds_count) # count
    output.append(getPercentOfTotal(dna_maint_cds_count, total_cds_count)) # percent of total
    return "
"
# percent of total
output.append(dna_maint_special_copy_num) # count
output.append(getPercentOfTotal(dna_maint_special_copy_num, dna_maint_cds_count)) # percent of dna maint
output.append(getPercentOfTotal(dna_maint_special_copy_num, total_cds_count)) # percent of total
output.append("Yes" if dna_maint_special_copy_num else "No") # present (Yes/No)

# mobile genetic elements
output.append(mobile_genetic_elements_cds_count) # count
output.append(getPercentOfTotal(mobile_genetic_elements_cds_count, total_cds_count)) # percent of total

# hypothetical genes
output.append(hypothetical_genes_cds_count) # count
output.append(getPercentOfTotal(hypothetical_genes_cds_count, total_cds_count)) # percent of total

# other (/unknown)
output.append(other_cds_count) # count
output.append(getPercentOfTotal(other_cds_count, total_cds_count)) # percent of total

# total
output.append(total_cds_count) # count

# convert all elements to str, join by ",", and add leading and trailing "
output = CSVify("\",\"".join(list(map(str, output))))

# return
return output

def getIncompatibilityGroups(input_incompatibility_groups_fn):
    with open(input_incompatibility_groups_fn, 'r') as ifd:
        return [line.rstrip('
') for line in ifd]

# ===== #
# MAIN #
# =====#

if __name__ == "__main__":
    import sys

    # handle args
    plasmid_accession, output_csv_dir, input_fasta_and_length_dir, input_matches_dir, input_incompatibility_groups_blast_output_dir = handleArgs()

    # set some helpful vars
    ocn = output_csv_dir + '/' + plasmid_accession + ".csv"
    ifn = input_fasta_and_length_dir + '/' + plasmid_accession + ".fasta"
    iln = input_fasta_and_length_dir + '/' + plasmid_accession + ".length"
    mfn = input_matches_dir + '/' + plasmid_accession + "_matches.tsv"
iign = input_incompatibility_groups_blast_output_dir + '/' + plasmid_accession + "_families.list"


# get necessary information
# get CSV Header
csv_header_output_str = CSVify("","".join(csv_header))

# get plasmid accession#
plasmid_accession_output_str = CSVify(plasmid_accession)

# get plasmid length
plasmid_length = getPlasmidLength(iiln)
plasmid_length_output_str = CSVify(str(plasmid_length))

# get CDS info (Antimicrobial Resistance CDS (%) ... Total CDS)
cds_info = parseMatchesFile(mfn)
cds_info_output_str = convertCdsInfoToOutputStr(*cds_info)

# get incompatibility groups
incompatibility_groups = getIncompatibilityGroups(iign)
incompatibility_groups_output_str = CSVify(', '.join(incompatibility_groups)) if len(incompatibility_groups) > 0 else CSVify("NA")

# write output
with open (ocn, 'w') as ocd:
    # csv header line
    ocd.write(csv_header_output_str + '\n') # csv header
    # csv data line
Step 13. Create CSVs from Plasmid CSVs

Input: The inputs required are the group list files that contain the plasmids in each group (see step #4) and the individual plasmid CSVs (see step #12). The group list files are assumed to be in the directory `groups` and named after the pattern `{GROUP}.list`. The plasmid CSVs are assumed to be in the `plasmid_csv` directory and named after the pattern `{ACCESSION}.csv`.

Output: One comma-separated value file containing the same header line as all the plasmid CSVs and a concatenation of the non-header lines from the plasmid CSVs. We assume the output file will be in the directory `group_csv` and will be named after the following pattern: `{GROUP}.csv`.

Code:

Bash Command

```bash
while read ifn
do
  GROUP=`basename "${ifn}" ".list``
ofn="group_csv/${GROUP}.csv"
  # get and write a header
  hfn=plasmid_csv/`head -q -n 1 "${ifn}" `.csv`
  head -q -n 1 "${hfn}" > "$ofn"
  # get and write the non-headers lines
  nhfns=`cat "${ifn}" | sed -r 's,^(.+)$,plasmid_csv/\1.csv,' | tr '
' ' '`
  tail -q -n +2 ${nhfns} >> "$ofn"
done < <(ls -l groups/*.list)
```
sed Note

sed must be GNU (https://www.gnu.org) sed. `-r` does not enable extended regular expression syntax with BSD (http://www.bsd.org) sed.

Step 14. Create Group Matches from Plasmid Matches

Note that this step is not technically necessary to generate the desired output (the group CSV files (step #13) and the group statistics files (step #15)). This is really for convenience in inspecting results.

Input: The inputs required are the group list files that contain the plasmids in each group (see step #4) and the individual plasmid matches (see step #11). The group list files are assumed to be in the directory `groups` and named after the pattern `${GROUP}.list`. The plasmid matches are assumed to be in the `plasmid_matches` directory and named after the pattern `${ACCESSION}_matches.tsv`.

Output: One text file containing the matches for the group. We assume the output file will be in the directory `group_matches` and will be named after the following pattern:

${GROUP}_matches.tsv.

Code:

_Bash Command_

```
while read ifn
do
    GROUP=`basename "${ifn}" ".list"`
    ofn="group_matches/${GROUP}_matches.tsv"
    fns="cat "${ifn}" | sed -r 's,^(.+)$,plasmid_matches/\1_matches.tsv,' | tr "\n" ' ' \
    head -q -n 1 ${fns} | head -n 1 > "${ofn}" \
    tail -q -n 2 ${fns} >> "${ofn}""
    done < </(ls -l groups/*.list)
```
**sed Note**

sed must be GNU (https://www.gnu.org) sed. `-r` does not enable extended regular expression syntax with BSD (http://www.bsd.org) sed.

**Step 15. Calculate Group Statistics from Group CSV**

Input: This Python program requires 2 inputs. 1- The CSV file for a group. Here, we show the CSV files in the directory `group_csv`, named after the pattern `${GROUP}.csv`. 2- The output statistics file for the group. Here, we show the statistics files in the directory `group_stats`, named after the pattern `${GROUP}.stats`.

Output: One text file named as described in position 2 of the input to the Python program.

That file is formatted as follows:

```
GROUP_NAME
===
Total # of Plasmids: ##

Incompatibility Groups Structure:
<table>
<thead>
<tr>
<th>Inc.</th>
<th>Group</th>
<th>Count</th>
<th>Size</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IncGrp1</td>
<td>#</td>
<td>#.###</td>
<td>#.###</td>
</tr>
<tr>
<td></td>
<td>IncGrp2</td>
<td>#</td>
<td>#####.###</td>
<td>#.###.###</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IncGrpN</td>
<td>#</td>
<td>#####.###</td>
<td>#.###.###</td>
</tr>
</tbody>
</table>

Plasmids Summary:
<table>
<thead>
<tr>
<th></th>
<th>Min:</th>
<th>Max:</th>
<th>Median:</th>
<th>Mean:</th>
<th>St. Dev.:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>####</td>
<td>#####</td>
<td>######</td>
<td>######.###</td>
<td>######.###</td>
</tr>
</tbody>
</table>

Key Words Structure:
<table>
<thead>
<tr>
<th>Key</th>
<th>Plasmid</th>
<th>Size</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>anti_microb_resist</td>
<td>##</td>
<td>######.###</td>
<td>######.###</td>
</tr>
<tr>
<td>anti_microb_resist_not</td>
<td>#</td>
<td>######.###</td>
<td>######</td>
</tr>
<tr>
<td>beta_lact</td>
<td>##</td>
<td>######.###</td>
<td>######.###</td>
</tr>
<tr>
<td>beta_lact_not</td>
<td>#</td>
<td>######.###</td>
<td>######.###</td>
</tr>
<tr>
<td>plasmid_transfer</td>
<td>##</td>
<td>######.###</td>
<td>######.###</td>
</tr>
<tr>
<td>plasmid_transfer_not</td>
<td>#</td>
<td>#####.###</td>
<td>#####.###</td>
</tr>
</tbody>
</table>
```
Plasmid Structure:
This information is already reported in the CSV file: GROUP_NAME.csv

Code:

Bash Command

```bash
while read gfn
do
    GROUP=`basename "$gfn" ".list``
    ifn="group_csv/${GROUP}.csv"
    ofn="group_stats/${GROUP}.stats"
    python3 calcGroupCSVstats.py\
        "${ifn}" \n        "${ofn}"
done < <(ls -1 groups/*.list)
```

Python Version


Python Script (calcGroupCSVstats.py)

```python
# FUNCTIONS #

def handleArgs(args, sefd, sexit):
    if len(args) != 3:
        sefd.write("\n\tERROR: Incorrect arguments\n\t	1- input group\n\t	2- output text file\n\n")
        sexit(1)
    ifn = sys.argv[1]
    ofn = sys.argv[2]
    return ifn, ofn

def writeIncGroupsStructure(ofd, inc_groups):
```
output = []
sizes = []

header1 = ("Inc.", "Plasmid", "Size", "Size")
header2 = ("Group", "Count", "Mean", "St. Dev.")

output.append(header1)
sizes.append(tuple(map(len, output[-1])))
output.append(header2)
sizes.append(tuple(map(len, output[-1])))

for inc_group in sorted(inc_groups.keys()):
    if inc_group != "NA":
        lengths = inc_groups[inc_group]
count = len(lengths)
mean = count
st_dev = 0
if count > 1:
    mean = stats.mean(lengths)
    st_dev = stats.stdev(lengths)
output.append((inc_group, str(count),
                   '{0:.3f}'.format(mean), '{0:.3f}'.format(st_dev)))
sizes.append(tuple(map(len, output[-1])))

ofd.write("Incompatibility Groups Structure:
")
for o, s in zip(output, sizes):
ofd.write('	')
ofd.write(o[0] + ' ' * (c0 - s[0] + 3))
ofd.write(o[1] + ' ' * (c1 - s[1] + 3))
ofd.write(o[3] + ' ' * (c3 - s[3] + 3))
ofd.write('
')

def getGroupStructureMeanStr(lengths):
    if len(lengths) > 0:
        return '{0:.3f}'.format(stats.mean(lengths))
    else:
        return "NA"

def getGroupStructureStDevStr(lengths):
    if len(lengths) > 1:
        return '{0:.3f}'.format(stats.stdev(lengths))
elif len(lengths) < 1: # == 0
return "NA"
else:  # == 1
    return str(lengths[0])

def writeGroupStructure(ofd, all_group_structure_fields):
    # set up the group structure arrays (to be populated with plasmid lengths)
    anti_microb_resist = []
    anti_microb_resist_not = []
    beta_lact = []
    beta_lact_not = []
    plasmid_transfer = []
    plasmid_transfer_not = []
    toxin = []
    toxin_not = []
    dna_maint = []
    dna_maint_not = []
    mob_gen_elem = []
    mob_gen_elem_not = []
    hypo_genes = []
    hypo_genes_not = []
    other = []
    other_not = []

    # extract the information and load it into the group structure arrays
    for group_structure_fields in all_group_structure_fields:
        # if three is a count, add it. else add it to the not. We're adding the length.
        length = group_structure_fields[0]
        anti_microb_resist_count = group_structure_fields[1]
        beta_lact_count = group_structure_fields[2]
        plasmid_transfer_count = group_structure_fields[3]
        toxin_count = group_structure_fields[4]
        dna_maint_count = group_structure_fields[5]
        mob_gen_elem_count = group_structure_fields[6]
        hypo_genes_count = group_structure_fields[7]
        other_count = group_structure_fields[8]

        if anti_microb_resist_count:
            anti_microb_resist.append(length)
        else:
            anti_microb_resist_not.append(length)

        if beta_lact_count:
            beta_lact.append(length)
        else:
            beta_lact_not.append(length)

        if plasmid_transfer_count:
            plasmid_transfer.append(length)
        else:
            plasmid_transfer_not.append(length)

        if toxin_count:
            toxin.append(length)
        else:
            toxin_not.append(length)

        if dna_maint_count:
            dna_maint.append(length)
else:
    dna_maint_not.append(length)
if mob_gen_elem_count:
    mob_gen_elem.append(length)
else:
    mob_gen_elem_not.append(length)
if hypo_genes_count:
    hypo_genes.append(length)
else:
    hypo_genes_not.append(length)
if other_count:
    other.append(length)
else:
    other_not.append(length)

# for each of the arrays, calc mean & st. dev., write to file

# calc
anti_microb_resist_mean_str = getGroupStructureMeanStr(anti_microb_resist)
anti_microb_resist_stdev_str = getGroupStructureStDevStr(anti_microb_resist)
anti_microb_resist_not_mean_str = getGroupStructureMeanStr(anti_microb_resist_not)
anti_microb_resist_not_stdev_str = getGroupStructureStDevStr(anti_microb_resist_not)
beta_lact_mean_str = getGroupStructureMeanStr(beta_lact)
beta_lact_stdev_str = getGroupStructureStDevStr(beta_lact)
beta_lact_not_mean_str = getGroupStructureMeanStr(beta_lact_not)
beta_lact_not_stdev_str = getGroupStructureStDevStr(beta_lact_not)
plasmid_transfer_mean_str = getGroupStructureMeanStr(plasmid_transfer)
plasmid_transfer_stdev_str = getGroupStructureStDevStr(plasmid_transfer)
plasmid_transfer_not_mean_str = getGroupStructureMeanStr(plasmid_transfer_not)
plasmid_transfer_not_stdev_str = getGroupStructureStDevStr(plasmid_transfer_not)
toxin_mean_str = getGroupStructureMeanStr(toxin)
toxin_stdev_str = getGroupStructureStDevStr(toxin)
toxin_not_mean_str = getGroupStructureMeanStr(toxin_not)
toxin_not_stdev_str = getGroupStructureStDevStr(toxin_not)
dna_maint_mean_str = getGroupStructureMeanStr(dna_maint)
dna_maint_stdev_str = getGroupStructureStDevStr(dna_maint)
dna_maint_not_mean_str = getGroupStructureMeanStr(dna_maint_not)
dna_maint_not_stdev_str = getGroupStructureStDevStr(dna_maint_not)
mob_gen_elem_mean_str = getGroupStructureMeanStr(mob_gen_elem)
mob_gen_elem_stdev_str = getGroupStructureStDevStr(mob_gen_elem)
mob_gen_elem_not_mean_str = getGroupStructureMeanStr(mob_gen_elem_not)
mob_gen_elem_not_stdev_str = getGroupStructureStDevStr(mob_gen_elem_not)
hypo_genes_mean_str = getGroupStructureMeanStr(hypo_genes)
hypo_genes_stdev_str = getGroupStructureStDevStr(hypo_genes)
hypo_genes_not_mean_str = getGroupStructureMeanStr(hypo_genes_not)
hypo_genes_not_stdev_str = getGroupStructureStDevStr(hypo_genes_not)
other_mean_str = getGroupStructureMeanStr(other)
other_stdev_str = getGroupStructureStDevStr(other)
other_not_mean_str = getGroupStructureMeanStr(other_not)
other_not_stdev_str = getGroupStructureStDevStr(other_not)

# write to file
ofd.write("Key Words Structure:\n")

# create columned output
output = []
sizes = []

header1 = ("Key", "Plasmid", "Size", "Size")
header2 = ("Word", "Count", "Mean", "St. Dev.")
output.append(header1)
output.append(header2)

output.append( ( "anti_microb_resist", str(len(anti_microb_resist)), anti_microb_resist_mean_str, anti_microb_resist_stdev_str ) )
output.append( ( "anti_microb_resist_not", str(len(anti_microb_resist_not)), anti_microb_resist_not_mean_str, anti_microb_resist_not_stdev_str ) )
output.append( ( "beta_lact", str(len(beta_lact)), beta_lact_mean_str, beta_lact_stdev_str ) )
output.append( ( "beta_lact_not", str(len(beta_lact_not)), beta_lact_not_mean_str, beta_lact_not_stdev_str ) )
output.append( ( "plasmid_transfer", str(len(plasmid_transfer)), plasmid_transfer_mean_str, plasmid_transfer_stdev_str ) )
output.append( ( "plasmid_transfer_not", str(len(plasmid_transfer_not)), plasmid_transfer_not_mean_str, plasmid_transfer_not_stdev_str ) )
output.append( ( "toxin", str(len(toxin)), toxin_mean_str, toxin_stdev_str ) )
output.append( ( "toxin_not", str(len(toxin_not)), toxin_not_mean_str, toxin_not_stdev_str ) )
output.append( ( "dna_maint", str(len(dna_maint)), dna_maint_mean_str, dna_maint_stdev_str ) )
output.append( ( "dna_maint_not", str(len(dna_maint_not)), dna_maint_not_mean_str, dna_maint_not_stdev_str ) )
output.append( ( "mob_gen_elem", str(len(mob_gen_elem)), mob_gen_elem_mean_str, mob_gen_elem_stdev_str ) )
output.append( ( "mob_gen_elem_not", str(len(mob_gen_elem_not)), mob_gen_elem_not_mean_str, mob_gen_elem_not_stdev_str ) )
output.append( ( "hypo_genes", str(len(hypo_genes)), hypo_genes_mean_str, hypo_genes_stdev_str ) )
output.append( ( "hypo_genes_not", str(len(hypo_genes_not)), hypo_genes_not_mean_str, hypo_genes_not_stdev_str ) )
output.append( ( "other", str(len(other)), other_mean_str, other_stdev_str ) )
output.append( ( "other_not", str(len(other_not)), other_not_mean_str, other_not_stdev_str ) )

for o in output:
    sizes.append(tuple(map(len, o)))
c0 = 0
c1 = 0
c2 = 0
c3 = 0
for size in sizes:
if size[0] > c0:
    c0 = size[0]
if size[1] > c1:
    c1 = size[1]
if size[2] > c2:
    c2 = size[2]
if size[3] > c3:
    c3 = size[3]

# actually write to file
for o,s in zip(output, sizes):
    ofd.write('	')
    ofd.write(o[0] + ' ' * (c0 - s[0] + 3))
    ofd.write(o[1] + ' ' * (c1 - s[1] + 3))
    ofd.write(o[3] + ' ' * (c3 - s[3] + 3))
    ofd.write('
')

########
# MAIN  
########
if __name__ == '__main__':

    import sys
    import statistics as stats
    ifn, ofn = handleArgs(sys.argv, sys.stderr, sys.exit)
    group_name = '.'.join(ifn.strip().split('/')[-1].split('.')[:-1])

    with open(ofn, 'w') as ofd:
        # write the groupname title to the output
        ofd.write(group_name + '
' + '=' * len(group_name) + '
')

        # parse the input file and extract necessary information
        with open(ifn, 'r') as ifd:

            # skip past header line
            ifd.readline()

            # set some handy vars
            total_number_of_plasmids = 0
            plasmid_lengths = []
            all_inc_groups = {}
            all_group_structure_fields = []

            # loop through each plasmid_record (line) in the input file
            for plasmid_record in ifd:
                # increment the total num of plasmids (one plasmid exists per line)
                total_number_of_plasmids += 1

                # split the record into its 28 separate columns/fields
                fields =
plasmid_record.rstrip('"').rstrip('"').rstrip('"').split(',"
","
"
"
")

    plasmid_accession = fields[0].strip('"')
plasmid_length = int(fields[1].strip('"'))
inc_groups = fields[28].strip('"').split('"
"
"
"
"
"
"
"
"
")

group_structure_fields = tuple(map(lambda field: int(field.strip('"')), (fields[1], fields[2], fields[4], fields[10], fields[12], fields[15], fields[21], fields[23], fields[25])))

    # capture length information
plasmid_lengths.append(plasmid_length)

    # capture info about inc groups
for inc_group in inc_groups:
    if inc_group not in all_inc_groups:
        all_inc_groups[inc_group] = []
all_inc_groups[inc_group].append(plasmid_length)

    # capture info about group structure
all_group_structure_fields.append(group_structure_fields)

    # write stuff to the output file
    # total number of plasmids
ofd.write("Total # of Plasmids: " + str(total_number_of_plasmids) + '\n')
ofd.write('\n') # extra newline

    # inc groups structure
writeIncGroupsStructure(ofd, all_inc_groups)
ofd.write('\n') # extra newline

    # group plasmids size
ofd.write("Plasmids Summary:\n")
ofd.write("\tMin: " + str(min(plasmid_lengths)) + '\n')
ofd.write("\tMax: " + str(max(plasmid_lengths)) + '\n')
ofd.write("\tMedian: " + str(stats.median(plasmid_lengths)) + '\n')

    if len(plasmid_lengths) > 1 else '0' + '\n')
ofd.write('\n') # extra newline

        # group structure
writeGroupStructure(ofd, all_group_structure_fields)
ofd.write('\n') # extra newline

        # plasmid structure
ofd.write("Plasmid Structure:\n")
ofd.write("\tThis information is already reported in the CSV file: " + ifn.split('/\n')[-1] + '\n')
ofd.write('\n') # extra newline
### APPENDIX C

#### TABLE 4: CR-plasmid accession numbers

<table>
<thead>
<tr>
<th>Accession #</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP008933 CP006661 CP018974 CP021534 CP021961 CP024836 CP026577 JX101693 KF732966</td>
</tr>
<tr>
<td>KX214669 CP006799 CP018977 CP021536 CP021962 CP024840 CP026584 JX104759 KF874496</td>
</tr>
<tr>
<td>KX214670 CP012902 CP018981 CP021546 CP022126 CP025006 CP026589 JX193301 KF874498</td>
</tr>
<tr>
<td>KP873171 CP014757 CP018992 CP021682 CP022693 CP025010 EU855787 JX193302 KF874499</td>
</tr>
<tr>
<td>CP010881 CP015835 CP018999 CP021687 CP023488 CP025039 EU855788 JX283456 KF914891</td>
</tr>
<tr>
<td>CP025626 CP015991 CP019001 CP021692 CP023554 CP025141 FJ628167 JX397875 KF954759</td>
</tr>
<tr>
<td>KT362706 CP016035 CP019006 CP021699 CP023871 CP025144 GU585907 JX424614 KF954760</td>
</tr>
<tr>
<td>MF353156 CP016402 CP019010 CP021709 CP023910 CP025147 GU595196 JX430448 KF976405</td>
</tr>
<tr>
<td>MF344563 CP016403 CP019014 CP021716 CP023914 CP025148 HF955507 JX442975 KF977034</td>
</tr>
<tr>
<td>AP010555 CP017937 CP019026 CP021734 CP023923 CP025463 HG969995 JX461340 KF992018</td>
</tr>
<tr>
<td>AP012055 CP017981 CP019053 CP021738 CP023926 CP025467 HG969998 JX988621 KF998104</td>
</tr>
<tr>
<td>AP018137 CP018366 CP019774 CP021743 CP023938 CP025517 HG969999 KC311431 KJ146687</td>
</tr>
<tr>
<td>AP018138 CP018426 CP019073 CP021754 CP023948 CP025948 HQ451074 KC845573 KJ187751</td>
</tr>
<tr>
<td>AP018141 CP018817 CP020049 CP021860 CP024192 CP026175 JN157804 KC887916 KJ187752</td>
</tr>
<tr>
<td>AP018142 CP018884 CP020110 CP021861 CP024522 CP026179 JN233705 KC887917 KJ440075</td>
</tr>
<tr>
<td>AP018143 CP018887 CP020119 CP021881 CP024529 CP026201 JN420336 KC887918 KJ440076</td>
</tr>
<tr>
<td>AP018144 CP018945 CP020848 CP021899 CP024557 CP026204 JN687470 KC820658 KJ217289</td>
</tr>
<tr>
<td>AP018145 CP018949 CP020854 CP021900 CP024805 CP026205 JN861072 KC820659 KJ217290</td>
</tr>
<tr>
<td>CP003224 CP018956 CP020902 CP021936 CP024818 CP026395 JQ349086 KF295829 KJ802404</td>
</tr>
<tr>
<td>CP003977 CP018959 CP021177 CP021941 CP024825 CP028570 HQ345074 KCB45573 KJ817751</td>
</tr>
<tr>
<td>CP004366 CP018963 CP021206 CP021947 CP024828 CP026401 JQ824049 KFC623109 KJ812998</td>
</tr>
<tr>
<td>CP004367 CP018968 CP021210 CP021952 CP024833 CP026474 JQ837276 KPD701335 KJ833392</td>
</tr>
<tr>
<td>MF344563 MF042356 KY882285 KY903014 KX711880 KU862632 KU167609 KR559890 KJ859826</td>
</tr>
<tr>
<td>MF344564 MF042357 KY887590 KY130431 KX756453 KU886034 KU295131 KR822247 KJ958927</td>
</tr>
<tr>
<td>MF344565 MF042358 KY887591 KY270849 KX783439 KU934011 KU295132 KT148595 KMA00601</td>
</tr>
<tr>
<td>MF344566 MF042359 KY887594 KY270850 KX783440 KX023261 KU295133 KT185451 KM877631</td>
</tr>
<tr>
<td>MF344567 MF133495 KY887595 KY271403 KX786648 KX094555 KU295134 KT345946 KM977631</td>
</tr>
<tr>
<td>MF344574 MF150120 KY887596 KY271413 KX786649 KX154765 KU295135 KT345947 KPD15289</td>
</tr>
<tr>
<td>MF511773 MF156708 KY930324 KY271414 KX830701 KX154765 KPD295136 KTE75788 KPD35882</td>
</tr>
<tr>
<td>MF547507 MF156709 KY930325 KY271415 KX868553 KX236178 KUF302800 KTF725789 KPF776609</td>
</tr>
<tr>
<td>MF557408 MF156711 KY978631 KY288024 KX881941 KX276209 KUF302801 KTF82613 KPF868646</td>
</tr>
<tr>
<td>MF557509 MF156713 LT009688 KY399972 KY928750 KX348144 KUF302802 KTF82615 KPF868647</td>
</tr>
<tr>
<td>MF557510 MF168402 LT009689 KY399973 KY928771 KX348145 KUF311941 KTF82616 KPF893385</td>
</tr>
<tr>
<td>MF557511 MF168403 LT216438 KY399974 KY928752 KX348146 KUF318419 KTF82618 KPK00015</td>
</tr>
<tr>
<td>MF582638 MF168404 LT381917 KY399975 KY960109 KX397752 KUF318421 KTF89376 KPF987218</td>
</tr>
<tr>
<td>MF679143 MF168405 MF402350 KY435936 KX601010 KX477677 KUF647721 KTF89595 KRF059864</td>
</tr>
<tr>
<td>MF679147 MF168406 MF402351 KY63220 KY20154 KY470734 KUF665641 KUF051707 KRF091915</td>
</tr>
<tr>
<td>MG049738 MF178139 MF402352 KY798505 KY01483 KY507346 KUF665642 KUF051708 KRF351290</td>
</tr>
<tr>
<td>MG053313 MF344561 MF402353 KY798506 KY062156 KX674681 KU726588 KUF051709 KRF559888</td>
</tr>
<tr>
<td>MG271839 MF344562 MF402354 KY798507 KY093013 KX683284 KU761328 KUF16708 KRF559889</td>
</tr>
<tr>
<td>MG516907 MG516908 MG516910 MG557998 MG557999 AM778842 CP011370 KCI189475</td>
</tr>
<tr>
<td>KG543497 KG603922 KG603933 KPB73172 KP975076 KUC78314 KXE19264 KX711879 KX889311</td>
</tr>
<tr>
<td>KY296095 KY494864 KY630469 MF168945 MF344578</td>
</tr>
</tbody>
</table>
### APPENDIX D

**TABLE 5**: Percent of plasmids belonging to each incompatibility group.

<table>
<thead>
<tr>
<th>Percent of plasmids</th>
<th>Inc Group</th>
<th>Percent of plasmids</th>
<th>Inc Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.22%</td>
<td>IncA/C</td>
<td>13.68%</td>
<td>IncN</td>
</tr>
<tr>
<td>11.88%</td>
<td>IncA/C2</td>
<td>0.67%</td>
<td>IncN2</td>
</tr>
<tr>
<td>0.22%</td>
<td>IncB/O/K/Z</td>
<td>0.67%</td>
<td>IncN3</td>
</tr>
<tr>
<td>0.67%</td>
<td>Col</td>
<td>0.22%</td>
<td>IncP1</td>
</tr>
<tr>
<td>1.35%</td>
<td>Col440I</td>
<td>0.90%</td>
<td>IncP6</td>
</tr>
<tr>
<td>1.79%</td>
<td>ColRNAI</td>
<td>1.35%</td>
<td>IncQ1</td>
</tr>
<tr>
<td>4.71%</td>
<td>IncFIA</td>
<td>0.22%</td>
<td>IncQ2</td>
</tr>
<tr>
<td>12.78%</td>
<td>IncFIB</td>
<td>8.97%</td>
<td>IncR</td>
</tr>
<tr>
<td>19.73%</td>
<td>IncFII</td>
<td>2.24%</td>
<td>repA</td>
</tr>
<tr>
<td>2.24%</td>
<td>IncHI1B</td>
<td>1.79%</td>
<td>IncU</td>
</tr>
<tr>
<td>0.45%</td>
<td>IncHI2</td>
<td>13.23%</td>
<td>IncX3</td>
</tr>
<tr>
<td>0.45%</td>
<td>IncHI2A</td>
<td>0.22%</td>
<td>IncX4</td>
</tr>
<tr>
<td>0.22%</td>
<td>IncI1</td>
<td>0.90%</td>
<td>IncX5</td>
</tr>
<tr>
<td>0.90%</td>
<td>IncI2</td>
<td>0.67%</td>
<td>IncX6</td>
</tr>
<tr>
<td>2.69%</td>
<td>IncL/M</td>
<td>0.90%</td>
<td>IncY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.62%</td>
<td>Unclassified</td>
</tr>
</tbody>
</table>

(Note: Total is greater than 100% because some plasmids have multiple incompatibility groups)
APPENDIX E

TABLE 6: Relative abundance of incompatibility groups among carbapenemase-carrying plasmids.

Supplementary Table 4: Relative abundance of incompatibility groups among carbapenemase-carrying plasmids.

<table>
<thead>
<tr>
<th>Carbapenemase Family</th>
<th>IncA/C</th>
<th>IncA/C2</th>
<th>IncB/O/K/Z</th>
<th>Col440I</th>
<th>ColRNAI</th>
<th>IncFIA</th>
<th>IncFIB</th>
<th>IncFII</th>
<th>IncHI1B</th>
<th>IncHI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPC</td>
<td>0.0%</td>
<td>4.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.5%</td>
<td>4.0%</td>
<td>5.6%</td>
<td>18.2%</td>
<td>20.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>NDM</td>
<td>0.0%</td>
<td>17.9%</td>
<td>0.6%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>6.0%</td>
<td>10.7%</td>
<td>28.0%</td>
<td>6.0%</td>
</tr>
<tr>
<td>IMP</td>
<td>0.0%</td>
<td>22.4%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>2.0%</td>
<td>0.0%</td>
<td>4.1%</td>
</tr>
<tr>
<td>VIM</td>
<td>3.2%</td>
<td>16.1%</td>
<td>0.0%</td>
<td>9.7%</td>
<td>3.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>6.5%</td>
<td>3.2%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Carbanapenemase Family</th>
<th>IncHI2A</th>
<th>IncI1</th>
<th>IncI2</th>
<th>IncL/M</th>
<th>IncN</th>
<th>IncN2</th>
<th>IncN3</th>
<th>IncP1</th>
<th>IncP6</th>
<th>IncQ1</th>
<th>IncQ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPC</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>2.5%</td>
<td>17.7%</td>
<td>0.0%</td>
<td>1.0%</td>
<td>0.5%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>0.5%</td>
</tr>
<tr>
<td>NDM</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>1.2%</td>
<td>3.0%</td>
<td>1.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>1.8%</td>
<td>0.0%</td>
</tr>
<tr>
<td>IMP</td>
<td>4.1%</td>
<td>2.0%</td>
<td>0.0%</td>
<td>10.2%</td>
<td>34.7%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>VIM</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>12.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Carbanapenemase Family</th>
<th>IncR</th>
<th>repA</th>
<th>IncU</th>
<th>IncX3</th>
<th>IncX4</th>
<th>IncX5</th>
<th>IncX6</th>
<th>IncY</th>
<th>NA</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPC</td>
<td>13.6%</td>
<td>5.1%</td>
<td>2.5%</td>
<td>5.6%</td>
<td>0.0%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>5.6%</td>
</tr>
<tr>
<td>NDM</td>
<td>4.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>29.2%</td>
<td>0.6%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.6%</td>
<td>2.4%</td>
</tr>
<tr>
<td>IMP</td>
<td>2.0%</td>
<td>0.0%</td>
<td>6.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>16.3%</td>
</tr>
<tr>
<td>VIM</td>
<td>12.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>35.5%</td>
<td></td>
</tr>
</tbody>
</table>

Note: Totals are greater than 100% because some plasmids carry more than one replicon type.