Computational Analysis of the Relationships Between Antibiotic Resistance and Plasmid Backbone Genes

Author:
Mariele Lensink

Abstract:

Antibiotic-resistant bacteria are becoming one of the leading public health threats globally. Although, the presence of plasmid-encoded antibiotic resistance has been both extensively researched and well-documented, the relationships between antibiotic resistance genes (ARGs) and plasmid backbone structure have not. Understanding these relationships would provide important insight into the history of how bacterial plasmids have developed and the potential of plasmid backbones to acquire ARGs. This research aims to provide a comprehensive analysis of the relationships between ARGs and plasmid backbone structure. By implementing a computational approach to characterize a large sample of plasmids representative of each incompatibility group, we determined what ARGs are most frequently associated with given plasmid backbones as well as common insertion patterns.

Introduction

The dissemination of antibiotic resistance genes (ARGs) in bacteria is a growing threat to public health worldwide. One contributing factor to the rapid spread of bacteria that are resistant to clinically relevant antibiotics is plasmid mediated horizontal gene transfer [3]. Plasmids are "circular DNA molecules that replicate independently of the chromosome and are able to transfer horizontally between bacteria by conjugation" [3]. These mobile genetic elements have been extensively researched, and studies have shown that "once resistance genes have become established on successful plasmids, they may rapidly spread across different strains, species, or even genera" [4]. This is because plasmids containing ARGs provide their host cell with a survival advantage over its competitors, such as the ability to thrive in the presence of otherwise lethal antibiotics. As a result, "plasmids carry a considerable variety of genes, including those that confer antibiotic resistance" [1]. This development in resistance in response to environmental pressures is a considerable threat to human health and presents serious challenges in regards to the efficacy of modern medical practice [1].

Although the existence of ARGs in plasmids has been studied extensively, lack of consistency and standardization in plasmid classification and the annotation of genes belonging to plasmids has inhibited scientists' ability to further understand plasmid evolution. Traditionally, plasmids have been classified and categorized into incompatibility groups, which were originally defined as the failure of two plasmids to reside in a host cell and be inherited together. More recently, incompatibility groups are being defined in terms of genetic similarity of the plasmid replicon, which is the region of the plasmid that encodes functions for the activation and control of replication [2]. This method of plasmid typing, however, is becoming antiquated because it makes the assumption that variation in the modular structure of plasmids, meaning that related functions are clustered in specific regions of the DNA, is due to different phylogenetic origins and that plasmids are built via the random juxtaposition of these different functional modules [6]. Instead, plasmid classification is starting to rely more on evolutionary strategy, in other words, the genes in the plasmid genome that are responsible for the functions related to its own survival and propagation. These genes make up the "plasmid backbone". It is the backbone genes (BGs) that

determine characteristics such as copy number, transfer frequency, host range, stability, and other qualities.

Therefore, developing a standardized system of naming backbone genes is imperative to the successful classification of plasmids based on modular function. There currently exist many discrepancies in the annotation of backbone genes. For example, multiple names exist for identical proteins, and often distantly related proteins can have the same name. These errors are the result of biases in sequence analysis programs and propagated via automated annotation programs [5].

Recently, Thomas et al. proposed a standardized plasmid backbone gene naming convention to help reconcile some of the issues in the annotation of plasmid backbone genes. Building on this, [17] used some of the reference plasmids and their backbone genes, and applied this nomenclature to identify other plasmid backbone genes showing high sequence similarity to these known references. The export feature of this tool allows one to export plasmid backbone genes that can be reasonably identified and called based on this new nomenclature. With some modification, these annotated genes can then be turned into a curated database following a standardized naming convention and applied to the automated annotation of plasmids. This standardized naming convention allows for the systematic comparison of plasmid organization and structure. Understanding the structure and organization of plasmids allows us to better understand the relationships and interactions between mobile genetic elements and the plasmid genes themselves, likely leading to differences in plasmid function.

It is for this reason that understanding the relationship between backbone genes and the insertion patterns of ARGs is so important and yet factors determining how and where ARGs insert into the plasmid backbone have not been determined. It is known that ARGs are inserted into plasmid sequences via transposition (transposons) and site specific recombination mechanisms (integron gene cassettes) [1]. Most transposons are thought to not have preference for specific insertion sites on plasmids but instead insert into new sites more or less at random [1]. Although integrons and their gene cassettes utilize site-specific recombination, the placement of the integron itself (the *int* gene for example) is also thought to be random, thus there is no discovered determinant for the overall insertion placement of the gene cassettes.
However, disruptions to the plasmid backbone organization could be deleterious to the plasmid and thus may play a role in limiting the locations of insertions and accessory load.

This research aims to deduce the relationships between the plasmid backbone genes of various incompatibility groups and ARGs by using bioinformatics tools and computational methods to annotate, identify, and analyze a large representative population of naturally occurring bacterial plasmids. First, a database was created by selecting backbone genes from the standardized backbone database whose names we can confidently call because they have been taken from a reference plasmid with a known determined incompatibility group. ARGs and nucleotide sequences representative of incompatibility groups were also pulled from the distinguished databases, resFinder and plasmidFinder, respectively [18,2]. All sequences underwent extensive formatting to create a database compatible with the annotation tool, Prokka [16]. Prokka was then used to annotate a large representative group of 8,895 plasmids from Genbank using the database created in this study as the priority reference. Output from Prokka was then run through python scripts to pull genes annotated only with our database and create tables of ARGs and BGs with their corresponding plasmid name as well as its assigned incompatibility group. Gene tables created by these python scripts were then analyzed for patterns and figures were created using RStudio.

In response to the movement from typing plasmids by incompatibility group toward a new method of categorizing plasmid backbones based on modular function, this research uses standardized backbone naming in aims to quantify distribution preferences for particular incompatibility groups and locations within them. In this way, results from this study can provide direction for research within the constructs of both categorizing methods in terms of how these patterns and relationships affect antibiotic resistance.

Methods:

Building the Database:

A database was built consisting of the nucleotide sequences of resistance genes for *Enterobacteriaceae* from resFinder, nucleotide sequences representative of incompatibility groups for classification from the plasmidFinder database, and backbone genes from the standardized naming database. The sequences from the standardized naming database were specifically curated for backbone genes that could be confidently named because they were pulled from a well-studied reference plasmid with a confirmed incompatibility group. Genes from this database then underwent extensive reorganizing, reformatting, and data cleaning. Each of these sequences from each database was then translated from nucleotides into amino acids by a code written in Python utilizing Biopython methods[14]. A tag was added to the beginning of each entry name according to its sequence type (ex: "RES" was added to each ARG name) to distinguish between annotation categories to be used in Python scripts later in the workflow. Each entry of the database was then organized and formatted according to the annotation tool, Prokka's standards.

It should be noted that the selection process for BGs from the annotation correction tool purposefully limits the reference BGs in our database to BGs with high confidence. This greatly limits the BGs incorporated into the database and could possibly eliminate novel BGs.

Comprehensive Plasmid Sample Group:

A large file in fasta format of 8895 plasmids were downloaded from NCBI's database in early December, 2018. Our NCBI query specified all plasmids between 30,000 and 200,000 base pairs, excluding artificially constructed plasmid (see appendix). At the time this was essentially every complete, naturally occurring plasmid on the database at the time. New submissions occur regularly and the database is continually growing.

Plasmid Annotation:

Each of the 8895 plasmids were annotated by the Prokka annotation tool in batches of ~1,000 using our created database as the priority reference and an e-value of 1e-01. Of the 10 output files created, the file with the ".tbl" suffix was used due to its optimal formatting as the input file for another tool written in Python (see appendix).

Data Extraction and Analysis:

Relevant information from the Prokka Output file was then extracted and reformatted into csv files using multiple Python scripts we created (see below). Data queries and statistical calculations were made using Microsoft Excel. Figures were created and data analysis was performed using RStudio [15].

Results:

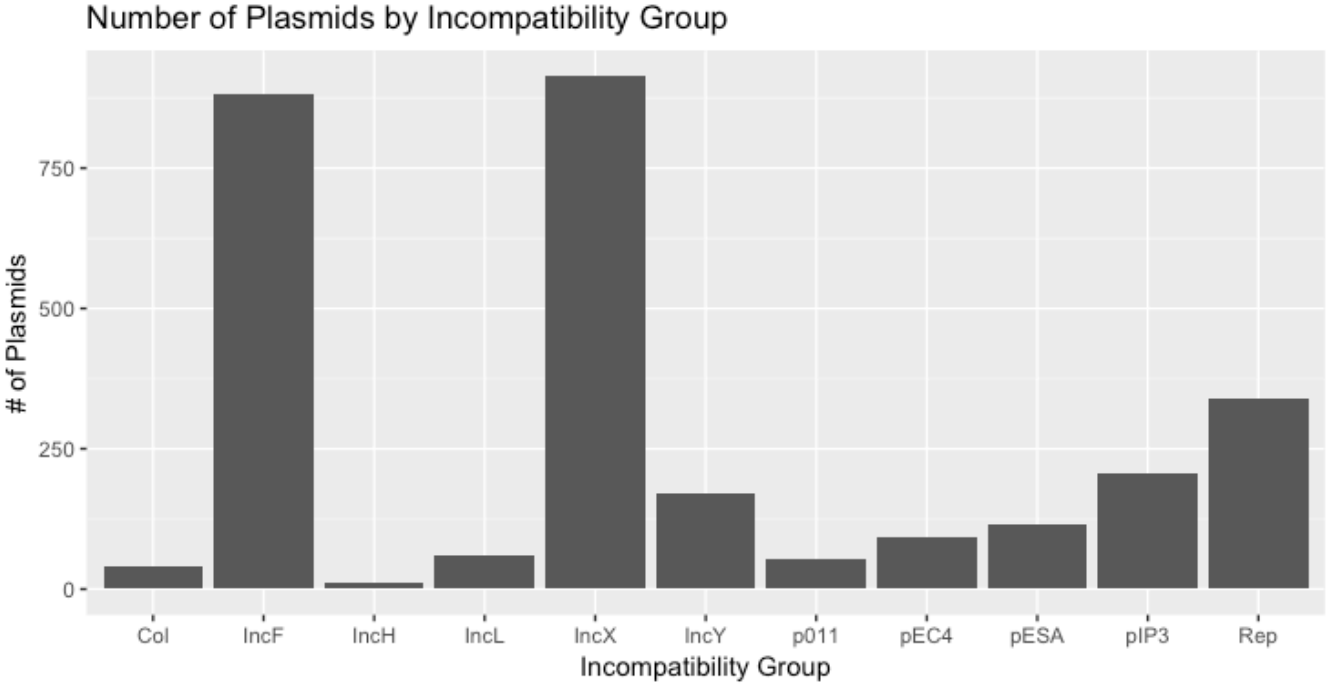
Of the 11 incompatibility groups identified, IncX had the most plasmids with 915, followed closely by IncF with 885[Figure 1]. There is a significant disparity between these two incompatibility groups and the remaining 9 groups, which together range from 11 (IncH) to 338 (Rep)[Figure1].

Of the various backbone genes annotated, the rep gene was most commonly found adjacent to an ARG. Every incompatibility group analyzed, except for IncH, displayed the Rep gene as most frequent [figure2]. It ranged from ~25% to ~45% of the total number of BG's nearest to an ARG in every incompatibility group [Figure 2]. Other more frequent neighboring BG's were TivB11, Pep, and the Par family proteins [Table1].

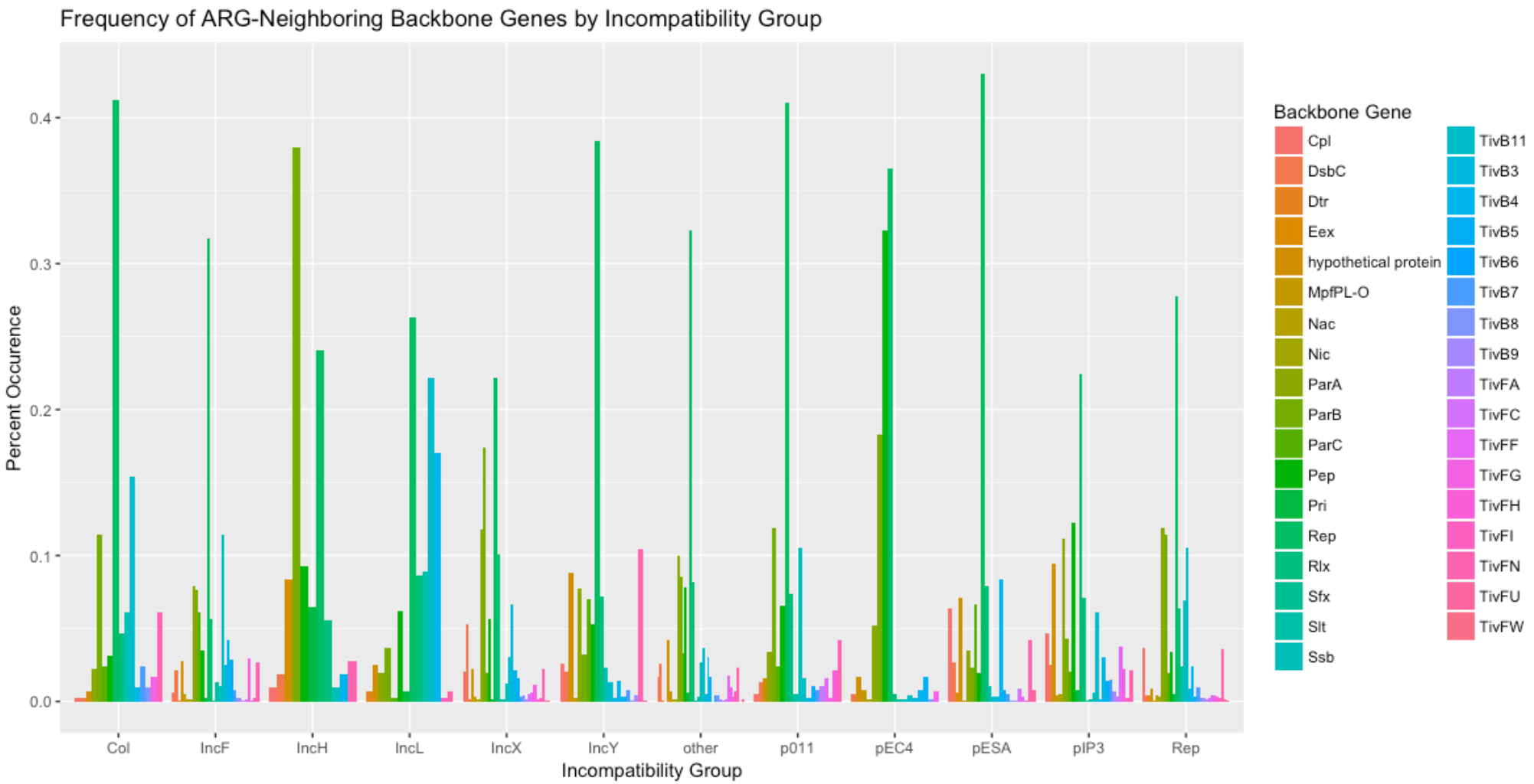
The IncX group showed the most diverse distribution of ARGs with 45 followed by pIP3 with 43, and then IncF with 38 [Table 2]. Other incompatibility groups, such as IncL and pEC4, displayed a less varied distribution and instead favored 1 or 2 frequent genes [Figure3].

Although IncX was the most common plasmid type, IncF displays a higher number of resistance genes. Both IncX and IncF harbor the greatest number of genes resistant to beta-lactam antibiotics, followed by aminoglycosides, then macrolides and tetracylines. Aminoglycoside resistance genes are the most common for IncY and pESA, but macrolide resistant genes are the most common for pIP3. The rep incompatibility group displays a similar distribution to that of both IncF and IncX, but in much smaller quantities [Figure 3].

IncF had the greatest number of ARGs for aminoglycosides, phenicols, beta-lactams, oxazolidinone, colistin, sulfonamides, and trimethoprim [Figure 4]. IncX had the greatest number of ARGs for macrolides, quinolones, tetracyclines, and fosfomycin [Figure 4]. Both IncF and IncX together had the greatest and second greatest number of ARGs for all but sulfonamides and trimethoprim, where IncF had the most but IncY had the second most before IncX [Figure 5].



[Figure 1] Displaying the number of plasmids annotated in each incompatibility group.



[Figure 2] Bar graph showing frequency of occurrence of the nearest adjacent backbone genes to any given resistance gene for each occurring incompatibility group. It should be noted that 3 incompatibility groups, IncB, pENT, and pYE8 were removed from the analysis due to lack of data.

	Col	IncF	IncH	IncL	IncX	IncY
1	Rep	Rep	ParB	Rep	Rep	Rep
2	TivB11	TivB11	Rep	TivB11	ParB	TivFN
3	ParB	ParA	Pep	TivB4	ParA	Hypothetical
4	Ssb	ParB	hypothetical	Slt	Rlx	ParA
5	TivFN	ParC	Pri	Rlx	Tivb3	Rlx
	p011	**pESA**	**pEC4**	**pIP3**	**Rep**	**Other**
1	Rep	Rep	Rep	Rep	Rep	Rep
2	ParB	Rlx	Pep	Pep	ParA	ParA
3	Ssb	TivB4	ParB	ParA	ParB	ParB
4	Rlx	hypothetical	ParA	hypothetical	TivB11	rlx
5	Pep	ParC	hypothetical	Rlx	ssb	Pep

[Table 1] Displays the 5 most frequent neighboring BG's in descending order for each incompatibility group.

Incompatibility Group	**Col**	**IncF**	**IncH**	**IncL**	**IncX**	**IncY**
# of ARG Types	24	38	15	17	45	29
	p011	**pESA**	**pEC4**	**pIP3**	**Rep**	**Other**
	23	24	29	43	33	55

[Table 2] Number of different types of antibiotic resistance genes present in each incompatibility group.

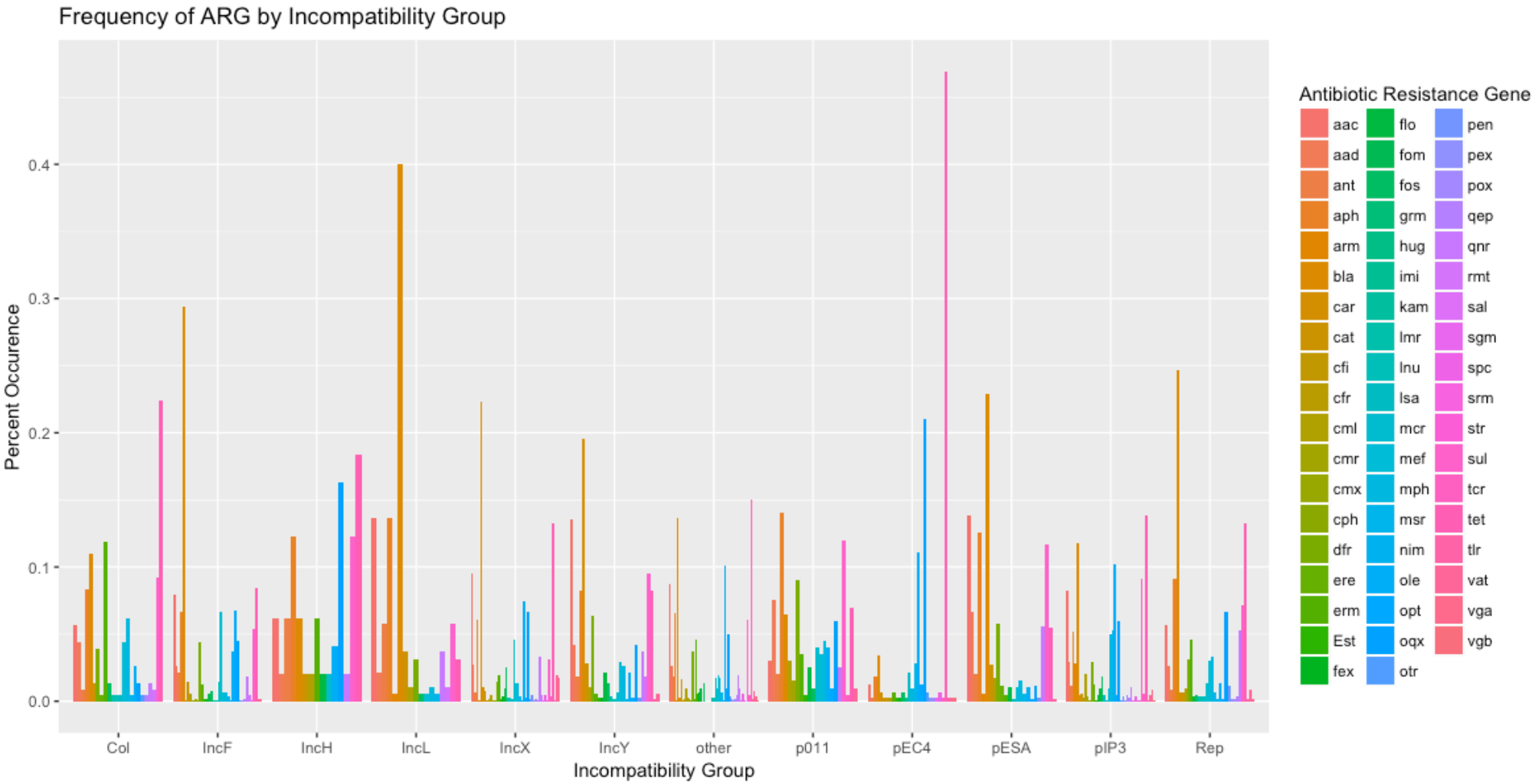
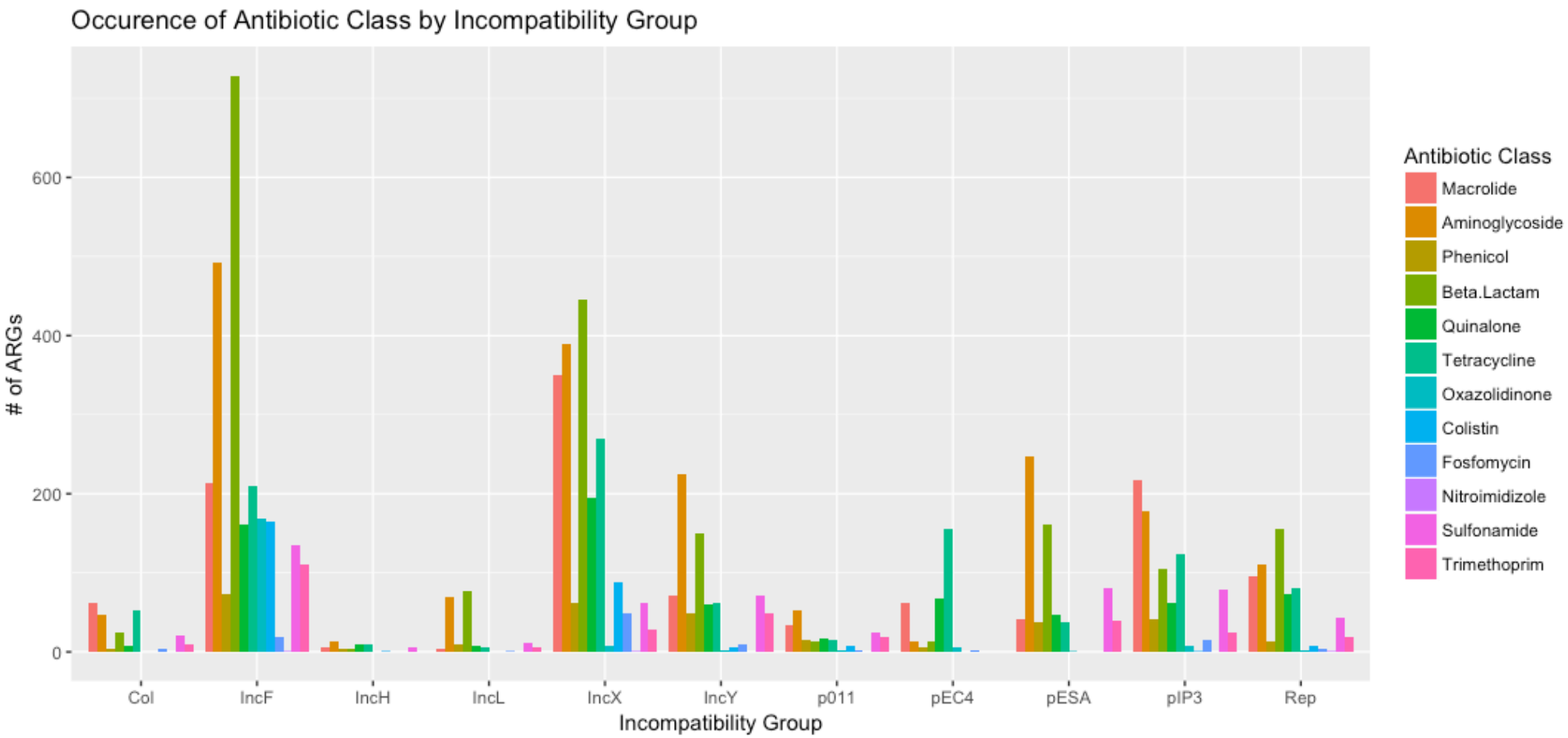
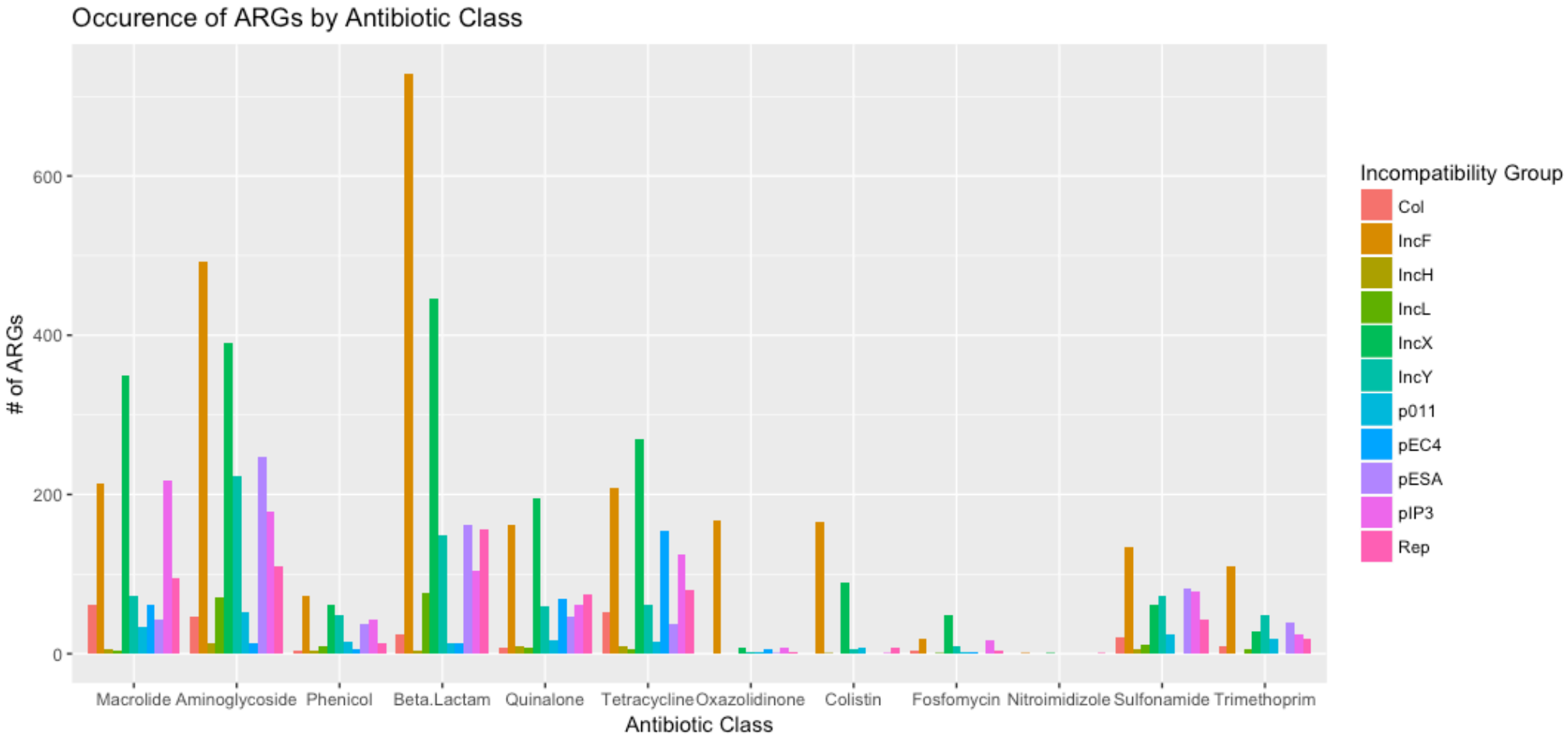


Figure [3] Bar graph showing the percent composition of occuring resistance genes for each incompatibility group. It should be noted that each three incompatibility groups, IncB, pENT, and pYE8 were removed from the analysis due to lack of data.



[Figure 4] Bar plot displaying the distribution of ARGs grouped by their antibiotic class for each incompatibility group.



[Figure 5] Bar graph showing counts of ARGs categorized by antibiotic type in accordance with resFinder database for each incompatibility group.

Discussion:

The drastic disparity in the representation of each incompatibility group, most importantly the high prevalence of IncF and IncX plasmids, presents a hypothesis for the natural composition of plasmids found in nature. IncF plasmids are known to be one of the predominant existing incompatibility groups in nature, whereas IncX plasmids have been thought to be in low prevalence [7,8]. However, after new replicon typing developments, it has been discovered that IncX plasmids in bacterial populations are much more common than had been previously understood. In our study, IncX plasmids were the most prevalent. This could partly be due to the algorithms used in annotation and data extraction, or the divergent nature of the nucleotide sequences of the IncX backbone [7]. Furthermore, it should be noted that the program written is limited to one incompatibility annotation for plasmid, selecting the last sequence found in the genome based on the starting location of the plasmid in the sequence itself. Therefore, the program does not account for hybrid replicons. An update to accommodate and account for hybrid backbones is a first priority area of further research.

Of the backbone genes detected that had ARGs inserted adjacent to them, the Rep gene was by far the most common, being the top most frequent BG neighbor in all but one incompatibility group. This poses interesting implications for the evolutionary strategy of insertion next to the Rep gene. For example, if the ARG requires the replicon promoter for transcription, it is possible that an increase in resistance to an ARG and copy number could act synergistically. In this way, copy number could increase with resistance, therefore encouraging more replication of the plasmid and greater prevalence in response to environmental pressure.

Of the 35 neighboring BGs annotated, only 15 showed up in the top 5 for each incompatibility groups. Furthermore, 7 of these were subtypes of the Par and Tiv families, therefore 8 distinct gene types were present. Although there is variation in the order of frequency between incompatibility groups, these results do indicate a preference for insertion of ARGs along the backbone, because if a BG is determined as a top-5 most-frequent neighbor, it is much more likely to be top-5 among other incompatibility groups as well.

Both of the most common plasmid types, IncF and IncX, show the broadest range of antibiotic resistance genes. IncF plasmids are widely understood to have a narrow host range, so to have a large variety of ARGs in comparison to other more promiscuous incompatibility groups poses interesting implications for possible survival strategies of this plasmid type [10]. Similarly, IncX and IncH plasmids are also understood to have narrow host ranges [7, 11]. In this study, IncH plasmids have neither a significant number nor variety of ARGs. In contrast, IncX plasmids have the greatest variation in ARGs annotated. It is possible that both IncF and IncX share similar evolutionary strategies as narrow host-range plasmids that persist by harboring a wide variety of resistance genes, enabling them to compete in and adapt to many different environmental pressures that their host range is exposed to. The only plasmid identified as having an intermediate/broad host range was IncL, and most of its resistances were for aminoglycosides and beta-lactams as well, implying thus far that neither antibiotic class shows particular favor towards a narrow or broad host-range backbone [11].

Of the ARGs annotated, those conferring resistance to beta-lactams, aminoglycosides, macrolides, and quinolones were the most common. Previous studies have identified *bla CTX-M* (beta lactam resistance), *rmtB* (aminoglycoside resistance), and *oqxB* (quinolone resistance) as the most common ARGs detected in the IncF plasmid family, indicating that IncF plasmids are likely to “carry multiple resistances determinants that render them resistant to different antibiotic classes

simultaneously"[9]. Data from this study supports these findings, as beta-lactams and aminoglycosides were the most common resistances found. Quinolone resistance genes however, were found at nearly the same number as macrolide resistance genes, which were 3rd most prevalent. Although IncX plasmids are understood to be associated most commonly with quinolone resistance, our study displays a distribution of resistance genes very similar to that of the IncF plasmids [8].

It is widely known and accepted that bacteria resistant to antibiotics are rapidly emerging, threatening the efficacy of our current antibiotics [12]. Having a profile of which resistances are being more commonly accumulated could direct further research in the discovery and development of new antibiotics or treatment methods against bacterial infection. Our data shows high frequencies of beta-lactam resistance, which is concurrent with the rise in serious infections by gram-negative bacteria in clinical settings that are multi-drug resistant, especially those producing extended-spectrum beta-lactamases (ESBLs) [12]. The CDC's assessment of bacterial threats classifies carbapenem-resistant *Enterobacteriaceae* as an urgent threat, and multi-drug resistant bacteria and ESBL producing *Enterobacteriaceae* as serious threats. Our data shows that each incompatibility group detected is capable of harboring ARGs to multiple antibiotic classes, and beta-lactam resistance is often one of the most common. Another one of the most common resistances found in our data were aminoglycosides, which have broad spectrum activity against pathogens, but are also known to act potently against *Enterobacteriaceae* [13].

Genes conferring resistance to beta-lactams, aminoglycosides, macrolides, quinolones, and tetracyclines occur in many of the different incompatibility groups with no particular inclination toward broad or narrow host-range plasmid families. Other less common antibiotics such as oxazolidinone and colistin, clearly favor the IncF incompatibility group. This could either be due to the structure and function of the IncF backbone, or its high prevalence in nature.

Conclusion:

This research serves as a pilot study aiming to formulate a new method of plasmid analysis and make available more data to aid and direct plasmid research in future studies. There are many patterns and relationships that can be further determined from the data collected and analyzed, and many hypotheses that can be derived from the results of this computational analysis. Recognizing the relationships present is the first step towards more targeted research that will provide us with explanations as to why these relationships exist and what their evolutionary purpose is.

Appendix:

NCBI Query:

((((Plasmid) AND Complete) AND 30000:200000[Sequence Length] AND bacteria[filter] AND biomol_genomic[PROP] AND plasmid[filter])) NOT vector NOT cloning

ProkkaReformat.py

```
#! /usr/bin/env python

#  Finds genes in Prokka output annotated from a user defined database, noted by annotation name_db.faa
#  Input csv output from Prokka (.tbl file)
#  Output csv with the start index, stop index, ARG name, Backbone Gene Name, Incompatibility Group, plasmid name

import sys
import re
import csv
import copy

def rowsFromProkka(inFile, dbDelim = ".faa"):
    ### Note that infile is a .csv file of output from Prokka
        # Input
    #   inFile = csv file for reading Prokka output
    #   dbDelim = the key indicating user defined database in the Prokka output
    # Output
    #   outlist = a list of each CDS's data formatted as
    #   (start, stop, arg, bg, inc croup, plasmidName, keep (Boolean indicating if it was from a passed in database)
    with open(inFile, "rU") as csvfile:

        csvreader = csv.reader(csvfile, delimiter= '\t')

        #create an empty dictionary to temporarily hold the information for one gene
        tempout = {"start" : "NA", "stop" : "NA", "geneName" : "NA", "plasmidName" : "NA", 'keep' : False}
        outlist = []

        for line in csvreader:
            # identify lines specifying the start of a new plasmid
            if len(line) == 1 and line[0].find(">")==0:
                # identify when the plasmids are switching and write out the previous record
                outlist.append([tempout[I] for I in tempout])
                # clear tempout
                tempout = {"start" : "NA", "stop" : "NA", "resName" : "NA", "geneName" : "NA","incGroup" : "NA",\
                 "plasmidName" : line[0][line[0].index(" "):], 'keep' : False}

            # identify rows starting a new CDS
            elif len(line) == 3 and line[2] == 'CDS':
                outlist.append([tempout[I] for I in tempout])
                # replace values in tempout, note that we only need to
                # replace the plasmid name when we get to a new plasmid
                tempout["start"] = line[0]
```

```
                tempout["start"] = line[0]
                tempout["stop"] = line[1]
                tempout["geneName"] = "NA"
                tempout["resName"] = "NA"
                tempout["incGroup"] = "NA"
                tempout['keep'] = False

            # identify if the CDS is from database and tag for keeping
            elif len(line)>4 and line[3] == 'inference' and line[4].find(dbDelim)>-1:
                tempout['keep'] = True

            # identify the gene name if it has one
            elif len(line)>4 and line[3] == 'product':
                if "group" not in line[4] and "RES" not in line[4]:
                    tempout['geneName'] = line[4]
                    tempout['incGroup'] = "NA"
                    tempout['resName'] = "NA"
                if "group" in line[4]:
                    tempout['incGroup'] = line[4]
                    tempout['geneName'] = "NA"
                    tempout['resName'] = "NA"
                if "RES" in line[4]:
                    tempout['resName'] = line[4]
                    tempout['geneName'] = "NA"
                    tempout['incGroup'] = "NA"
    return outlist

def cleanList(CDSlist):
    ### function removes rows that were not from the current database
    # Input
    #   CDSlist = a list of the information contained for each CDS in one row.
    #       Last element determines if it was from user defined database
    # Out
    #   returns a list of only the rows from CDS's from our database,
    #   removes column identifying keeper rows
    return [row[:-1] for row in CDSlist if row[-1]]

print(rowsFromProkka("prokkaAnnotation.tbl"))#insert name of input file (from prokka) here
cleanedUp = (cleanList(rowsFromProkka("prokkaAnnotation.tbl")))#insert name of input file (from prokka) here


with open('GeneTableOutput.csv','w') as csvfile: #name of output file here

    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(cleanedUp)
```

GeneTableEdit.py

```
#takes gene table file and alters entries to be more compatible with R
#repeats inc group name in every corresponding row, standardizes inc group variations
#input is .csv file from prokkaReformat.py
#output is another gene table that is compatible with R
import re
import csv

with open("inputGeneFile.csv", 'r') as csvreadfile: #input file here
    csvreader = csv.reader(csvreadfile)
    incdict = {}
    outList = []
    incNEW=""
    plasmid = ""
    #builds a dictionary of plasmids and their corresponding inc groups
    for line in csvreader:
        plasmid = line[5]
        if not line[4] == 'NA':
            incOLD = line[4]
            #gets rid of subtypes of inc groups
            if 'Col' in line[4] or 'col' in line[4] or 'Rep' in line[4] or 'rep' in line[4]:
                incNEW = incOLD[5:8]
            else:
                incNEW = incOLD[5:9]
            incdict[plasmid]=incNEW
    print(incdict)
```

```
with open("inputGeneFile.csv", 'r') as csvreadfile: #reopened same file
    csvreader = csv.reader(csvreadfile)
    resName1 = ""
    for line in csvreader:
        start= line[0]
        stop = line[1]
        #gets rid of RES tag to identify gene type and standardizes name
        if "RES" in line[2]:
            resName1 = line[2]
            resName = resName1[3:]
        else:
            resName = line[2]
        geneName = line[3]
        plasmid = line[5]
        #assigns inc group to outlist according to to dictionary
        if plasmid in incdict.keys():
            incgroup = incdict[plasmid]
        else:
            incgroup = "other"
        outList.append([start,stop,resName,geneName,incgroup,plasmid])

with open('outputfile.csv','w') as csvwritefile: #insert name of output file
    csvwriter = csv.writer(csvwritefile)
    csvwriter.writerows(outList)
```

FindNeighbors.py

```
#takes gene table and finds the nearest backbone genes up and downstream of every ARG
#input file: file created by prokkaReformat.py
#output file is .csv of ARG, upstream BG, downstream BG, inc group, and plasmid name
import csv
with open("GeneTableInput.csv",'r') as  csvreadfile: #write in path to input file
    csvreader = csv.reader(csvreadfile)
    tempN1 = ""
    N1 = ""
    N2 = ""
    outList = []
    resList = []
    afterRES = False
    for line in csvreader:
        #set gene in BG column to upstream neighbor temporarily
        if not line[3] == "NoGene" and afterRES == False:
            tempN1 = line[3]
        #if theres an ARG, set temp upstream neighbor to neighbor 1 and assign ARG
        if not line[2] == "NoGene":
            N1 = tempN1
            resName = line[2]
            afterRES = True
            #resList accounts for the possibility of multiple ARGs in a row
            resList.append(resName)
        #first BG that occurs after the res gene is neighbor 2 (downstream)
        if not line[3] == "NoGene" and afterRES == True:
            N2 = line[3]
            afterRES = False
            #every ARG in a cluster will share the same up/downstream BG neighbors
            for gene in resList:
                outList.append([gene,N1,N2,line[4],line[5]])
            tempN1 = line[3]
            resList = []
with open('OutputFile.csv','w') as csvwritefile: #assign name for output file
    csvwriter = csv.writer(csvwritefile)
    csvwriter.writerows(outList)
```

ResGeneEdit.py

```
#takes output .csv file from GeneTableEdit.py and edits to make more compatible with R
#changes instances of "NA" to "NoGene", cleans ARG names
#output .csv file of the same format
import csv
with open("inputFile.csv",'r') as  csvreadfile: #write output from genetableedit.py
    csvreader = csv.reader(csvreadfile)
    start = ""
    stop = ""
    ResGene1 = ""
    ResGene = ""
    BackboneGene = ""
    IncGroup = ""
    PlasmidName = ""
    outlist = []
    for line in csvreader:
        start = line[0]
        stop = line[1]
        #cleans/standardizes ARGs by removing subgroups
        ResGene1 = line[2]
        ResGene = ResGene1[:3]
        #change "NA"s to NoGene in ARG column
        if ResGene == "NA":
            ResGene = "NoGene"
        BackboneGene = line[3]
        #change "NA"s to NoGene in BG column
        if BackboneGene == "NA":
            BackboneGene = "NoGene"
        IncGroup = line[4]
        PlasmidName = line[5]

        outlist.append([start,stop,ResGene,BackboneGene,IncGroup,PlasmidName])
with open('OutputFile.csv','w') as csvwritefile: #write in output file name
    csvwriter = csv.writer(csvwritefile)
    csvwriter.writerows(outlist)
```

translator.py

```
#!/usr/bin/env python
from Bio import SeqIO

outFile = "resGenes.faa"
outhandle = open(outFile, 'w')

inSeqs = SeqIO.parse(open("resGenes.fna", "rU"), "fasta")
for record in inSeqs:
    record.seq = record.seq.translate()
    print(record)
    SeqIO.write(record, outhandle, "fasta")
outhandle.close()
```

Works Cited

[1] Bennett, P. M. (2009). Plasmid encoded antibiotic resistance: Acquisition and transfer of antibiotic resistance genes in bacteria. *British Journal of Pharmacology,153*(S1). doi:10.1038/sj.bjp.0707607

[2] Carattoli, A., Zankari, E., García-Fernández, A., Larsen, M. V., Lund, O., Villa, L., . . . Hasman, H. (2014). In SilicoDetection and Typing of Plasmids using PlasmidFinder and Plasmid Multilocus Sequence Typing. *Antimicrobial Agents and Chemotherapy,58*(7), 3895-3903. doi:10.1128/aac.02412-14

[3] Evolution of Plasmid-Mediated Antibiotic Resistance in the Clinical Context. (2018, July 23). Retrieved from https://reader.elsevier.com/reader/sd/pii/S0966842X18301422?token=F82E68DBF3479182EFC4BEB79DCD952E3CB1267B74C51DA295CCB21CF8A95D96DED6287CE62FDE1569EB9E5C11AAAA75

[4] Wintersdorff, C. J., Penders, J., Niekerk, J. M., Mills, N. D., Majumder, S., Alphen, L. B., . . . Wolffs, P. F. (2016). Dissemination of Antimicrobial Resistance in Microbial Ecosystems through Horizontal Gene Transfer. *Frontiers in Microbiology,7*. doi:10.3389/fmicb.2016.00173

[5] Thomas, C. M., Thomson, N. R., Cerdeño-Tárraga, A. M., Br own, C. J., Top, E. M., & Frost, L. S. (2017). Annotation of plasmid genes. *Plasmid,91*, 61-67. doi:10.1016/j.plasmid.2017.03.006

[6] Thomas, C. M. (2011). Evolution and Population Genetics of Bacterial Plasmids. *Plasmid Biology,*509-528. doi:10.1128/9781555817732.ch25

[7] Johnson, T. J., Bielak, E. M., Fortini, D., Hansen, L. H., Hasman, H., Debroy, C., . . . Carattoli, A. (2012). Expansion of the IncX plasmid family for improved identification and typing of novel

plasmids in drug-resistant Enterobacteriaceae. *Plasmid,68*(1), 43-50. doi:10.1016/j.plasmid.2012.03.001

[8] Dobiasova, H., & Dolejska, M. (2016). Prevalence and diversity of IncX plasmids carrying fluoroquinolone and β-lactam resistance genes inEscherichia colioriginating from diverse sources and geographical areas. *Journal of Antimicrobial Chemotherapy,71*(8), 2118-2124. doi:10.1093/jac/dkw144

[9] Yang, Q., Sun, J., Li, L., Deng, H., Liu, B., Fang, L., . . . Liu, Y. (2015). IncF plasmid diversity in multi-drug resistant Escherichia coli strains from animals in China. *Frontiers in Microbiology,6*. doi:10.3389/fmicb.2015.00964

[10] Villa, L., García-Fernández, A., Fortini, D., & Carattoli, A. (2010). Replicon sequence typing of IncF plasmids carrying virulence and resistance determinants. *Journal of Antimicrobial Chemotherapy,65*(12), 2518-2529. doi:10.1093/jac/dkq347

[11] Suzuki, H., Yano, H., Brown, C. J., & Top, E. M. (2010). Predicting Plasmid Promiscuity Based on Genomic Signature. *Journal of Bacteriology,192*(22), 6045-6055. doi:10.1128/jb.00277-10

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

[13] Krause, K. M., Serio, A. W., Kane, T. R., & Connolly, L. E. (2016, June). Aminoglycosides: An Overview. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4888811/

[14] Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., . . . Hoon, M. J. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics,25*(11), 1422-1423. doi:10.1093/bioinformatics/btp163

[15] RStudio Team (2015). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.

[16] Seemann T. *Prokka: rapid prokaryotic genome annotation*, Bioinformatics 2014 Jul 15; 30(14):2068-9. PMID:24642063

[17] Botts, R., Lindsey, Z., Ustick, L., Peterson, K., Dawne, P., Brown, C. and Cummings, D. PlBackNMR. https://zaclindsey.shinyapps.io/plasmidbackbone2/

[18] Zankari, E., Hasman, H., Cosentino, S., Vestergaard, M., Rasmussen, S., Lund, O., . . . Larsen, M. V. (2012). Identification of acquired antimicrobial resistance genes. *Journal of Antimicrobial Chemotherapy,67*(11), 2640-2644. doi:10.1093/jac/dks261