Compatibility patterns in antibiotic resistant genes in pathogens

Master's thesis in Biotechnology

AGNES LINDBOM

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Compatibility patterns in antibiotic resistant genes in pathogens

AGNES LINDBOM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2021

Compatibility patterns in antibiotic resistant genes in pathogens
AGNES LINDBOM

Supervisor: Erik Kristiansson, Department of Mathematical Sciences
Examiner: Erik Kristiansson, Department of Mathematical Sciences

Master's Thesis 2021
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Compatibility patterns in antibiotic resistant genes in pathogens
AGNES LINDBOM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Antibiotic resistance is increasing globally and is a substantial threat to the public health with higher costs and longer hospital stays. Infections of antibiotic resistant bacteria are harder to treat and without actions, there is a risk common infections eventually can become life-threatening again. Antibiotic resistance can be acquired by mutations in existing genes or from resistance genes transferred from other bacteria by horizontal gene transfer. The reason why some genes are transferred could be explained by the compatibility of the gene, but currently, there is not so much information what would be needed to make a gene compatible and able to be transferred and make new antibiotic resistant bacteria.

The aim of this project is to investigate whether there are compatibility patterns in antibiotic resistant genes in the pathogens *Escherichia coli* and *Klebsiella pneumoniae*. This was done by three analyses, kmer analysis, frequency analysis and analysis of regions surrounding the gene. The project also included predictive models to investigate the predictive ability of a logistic regression model. The data was collected from NCBI and ResFinder and core genomes from the species were used. The antibiotic resistant genes were divided into groups whether they were present or not in the species after using BLAST.

The kmer analysis used the kmer distributions of different kmer lengths in three methods; squared Euclidean distance, absolute maximum values and maximum value and gave similar results for both species. For smaller kmer lengths, differences could be seen for the species in the median and p-values between the antibiotic resistant genes and the core genome genes. For increasing kmer lengths less differences could be seen. In the frequency analysis the genes were merged into genes groups and the values from kmer analysis were compared against the number of hits from the BLAST results. Many values clustered around the median values and the gene groups with the most hits were close to the median values while the values far away from the median values did not have many hits. No clear conclusion about different antibiotic classes could be seen. In the analysis of regions surrounding the gene, sequences of 100 bp upstream and downstream of each gene were cut and the genes were divided into groups whether they were present in both or one species. There could be seen differences between the unique groups of the species, while the difference between the shared groups were surprisingly low. On kmer level, the kmers that differed most between the species had no clear correlation, potentially they could be related to the higher GC content in *K. pneumoniae*. Predictive models were created with logistic regression for all genes and three different antibiotic classes. The model for all genes included the length and 21 kmers out of 64 possible kmers. The model performed better than a random classification with the best values of true positive rate of 72% and false positive rate of 11%. The other models included fewer kmers and most of the classifier performed better than a random classification.

In this project analyses have developed and from the results it has been found compatibility patterns of the antibiotic resistant genes by looking at the gene sequences and the regions surrounding the genes. From the gene sequences it has also been possible to predict the gene compatibility in a predictive model.

Keywords: antibiotic resistance, pathogens, horizontal gene transfer, gene compatibility, predictive model, logistic regression, bioinformatics

Contents

List of Figures

List of Tables

1
Introduction

Antibiotic resistance is increasing globally and is a substantial threat to the public health with higher costs, longer hospital stays and increased mortality. Infections of antibiotic resistant bacteria are harder or impossible to treat and without actions, there is a risk that common infections and minor injuries eventually can become life-threatening again [1].

Antibiotic resistance can be acquired either by mutations in existing genes or from resistance genes transferred from other bacteria by horizontal gene transfer. The number of resistance genes found in pathogenic bacteria is increasing but the reason why some genes are transferred and some are not is not clear. One reason could be explained by the compatibility of the gene i.e., if the gene is able to be incorporated into the genome of the pathogen. Currently, there is not so much information what would be needed to make a gene compatible and therefore it is not possible to predict which resistance genes that could be transferred and make new antibiotic resistant bacteria [2].

With the knowledge of why genes would be transferred it could increase the understanding of antibiotic resistance helping future research on how to develop new antibiotics and predict future resistant genes in bacteria to mitigate the potentially damage.

Aim

The aim of this project is to investigate whether there can be seen any patterns in antibiotic resistant genes related to gene compatibility in the pathogens *Escherichia coli* and *Klebsiella pneumoniae*. This will be done by looking at various features including distribution of kmers and the surroundings of the genes by developing different methods. The project will also include a predictive model to investigate the predictive ability of a logistic regression model using data from previously developed methods.

2
Theory

In this chapter theory about why and how antibiotic resistance is acquired and the bacteria used in the project are described. Also, there can be found sections about predictive models, logistic regression and concepts related to the performance of a model.

2.1 Antibiotic resistance

Antibiotic resistance occurs when bacteria develop the ability to protect them from antibiotics that will stop their growth or kill them. It can occur by a mutation in the genome of the bacteria or by acquiring antibiotic resistance genes from other bacteria by horizontal gene transfer. The antibiotics are grouped based on their chemical structure, for example aminoglycosides, macrolides, beta-lactams (penicillin) and tetracyclines. The targets of the antibiotics include the main mechanisms; to inhibit cell wall synthesis, depolarize the cell membrane, inhibit protein synthesis, inhibit nuclei acid synthesis and inhibit metabolic pathways in bacteria. To avoid the mechanisms of the antibiotics the pathogen develop mechanisms from the resistance genes to inactivate the drugs in different ways as to limit the uptake of the drug, modify the drug target and to inactive the drug or the active efflux of a drug [3].

2.2 Horizontal gene transfer

Horizontal gene transfer is a naturally occurring process between bacteria to exchange genetic material and is the primary mechanism to spread the antibiotic resistance in bacteria. It has a big role in the evolution of bacterial genomes, the bacteria can rapidly acquire new capabilities to survive and adapt to environmental changes. There are three main types of horizontal gene transfer: transformation, transduction and conjugation. In transformation, the bacteria take up foreign genetic material from the environment, transduction is the process in which bacterial DNA is moved from a bacteria to another by a virus bacteriophage and bacterial conjugation is a process where the transfer of DNA occurs via a plasmid from a donor cell to a recipient cell during direct cell contact [2].

2.3 Bacteria

In this project genomes from two species of pathogenic bacteria have been used, *Escherichia coli* and *Klebsiella pneumoniae*.

2.3.1 *Escherichia coli*

Escherichia coli is a gram-negative bacteria part of the family *Enterobacteriaceae*. Most *E. coli* strains are harmless and can be found in human as an important part of a healthy intestinal tract [4]. *E. coli* is well-studied with an extensive knowledge of its genetics and genomics and has become the most important model organism. Particularly the strain K-12, adapted to the laboratory environment, is one of the most frequently used strains and was already in 1997 one of the first organisms to be fully sequenced [5]. The genome of an *E. coli* strain consist of between 4000-5000 genes, where the K-12 strain consists of around 4400 genes [6].

2.3.2 *Klebsiella pneumoniae*

Klebsiella pneumoniae is also a gram-negative bacteria member of the *Enterobacteriacae* family and a relative to previously described *E. coli*. Normally *K. pneumoniae* can be found in the human intestine, where it does not cause any disease, but typically infect people with compromised immune systems. In hospital-acquired infections *K. pneumoniae* is an important pathogen and tends to acquire multidrug-resistance, with limited options for treatment of related infections [7]. A genome of *K. pneumoniae* consists of around 5400 genes [8].

2.4 Predictive model

A predictive model uses statistics to make predictions. The model typically uses a machine learning algorithm to learn from a training data set with known results to be able to make predictions from different or new data. The modeling results in predictions representing a probability of the target variable based on estimated significance from a set of input variable. Predictive models can broadly be classified into two categories, parametric and non-parametric and basically any statistically method can be used [9].

2.4.1 Logistic regression

In classification problems the logistic regression can be used as the statistical method to the predictive model. The logistic regression model is parametric, a type of a generalised linear model, where the dependent or response variable is binary and uses the logit function with the probability p, see Equation 2.1.

$$logit(p) = \log\left(\frac{p}{1-p}\right) \tag{2.1}$$

In the logistic regression model, the parameters β_i are estimated and fixed into the model. With predictors x_i from the input data, the model evaluates the probability by calculate the log-odds, the logarithm of the odds ratio, by using the logit function [10]. An example with three parameters and two predictors can be seen in Equation 2.2.

$$\textit{log-odds} = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{2.2}$$

In the equation, β_0 is the intercept and β_1 and β_2 are the parameters affecting the log-odds when the predictors increase or decrease. For example if $\beta_1 = 1$ would mean when increasing x_1 by 1 the log-odds would increase by 1.

2.4.2 Sensitivity and specificity

Sensitivity and specificity are used in classification models and measure the performance of the model. From a test there are four possible outcomes, if the test is positive and it is classified as positive, it is a true positive, if it is classified as negative, it is a false negative. If the test instead is negative and it is classified as negative, it is a true negative, if it is classified as positive, it is a false positive. [11]

The sensitivity of a test represents the probability of correctly identify the positives. In a highly sensitive test, there are few false negative results, meaning few positives are missed. When identification of all positives is of interest the sensitivity is not useful as it does not take into consideration the false positives [11]. The sensitivity can be calculated according to Equation 2.3.

$$\textit{sensitivity} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}} \tag{2.3}$$

In contrast, the specificity of a test represents the probability of correctly identify the negatives. In a high specific test, there are few false positive results and few negatives are missed. When identification of all negatives is of interest the sensitivity is not useful as it does not take into consideration the false negatives [11]. The specificity can be calculated according to Equation 2.4.

$$\textit{specificity} = \frac{\textit{true negatives}}{\textit{true negatives} + \textit{false positives}} \tag{2.4}$$

2.4.3 Receiver operating characteristic curve

To illustrate the specificity and sensitivity a receiver operating characteristic (ROC) curve can be used. The ROC curve is a graph created by plot the true positive rate (TPR) on the y-axis against the false positive rate (FPR) on the x-axis at various cutoffs for the classification. The true positive rate is the sensitivity and also known as the probability of detection, while the false positive rate is known as

the probability of false alarm and can be calculated as (1-specificity) [11]. Examples of ROC curves can be seen in Figure 2.1.

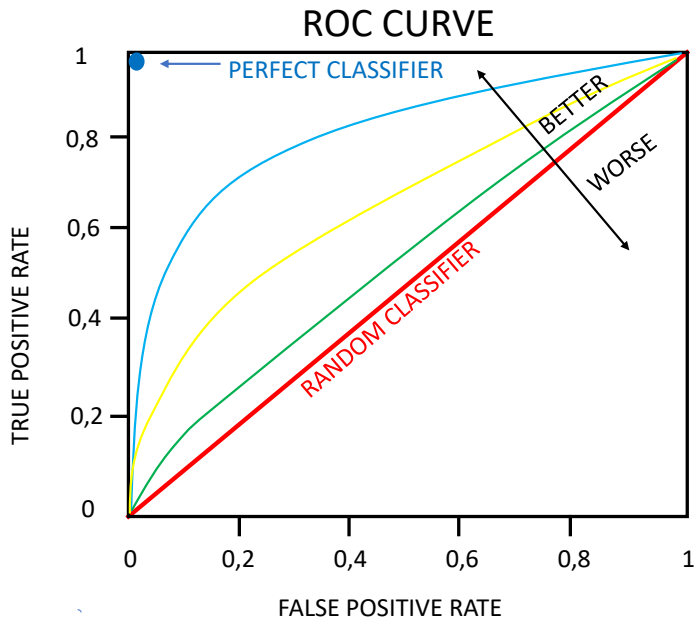


Figure 2.1: Examples of ROC curves. The sensitivity or true positive rate is on the y-axis and (1-specificity) or false positive rate is on the x-axis. The line y=x, in red, represents a random classification, the points below the line would perform worse than a random classification while the green, yellow and blue curves perform better than a random classification. The point (0,1), in dark blue, represents a perfect classification.

In the ROC plot there are some parts to note, the point (0, 1) represents a test with perfect classification, if a point is on the diagonal line y=x the performance of the test represents random classification and any point under the line performs worse than a random classification. Points on the lower left side make positive classifications with strong evidence with few false positive errors, but also have low true positive rates. On the other hand, points on the upper right side make positive classifications with weak evidence and all positives correctly, but they often have high false positive rates.

If the usefulness of the total test is of interest, the area under the curve can be calculated. Then the total measure of performance is provided across all possible classification cutoffs with the highest possible value of 1 [11].

3
Methods

In this chapter the methods used to investigate the gene compatibility of the antibiotic resistant genes are described. This includes how the DNA was collected, three different types of analyses and a predictive model.

3.1 DNA collection

Genomes and genes together with core genomes were collected for the species *Escherichia coli* and *Klebsiella pneumoniae*. The antibiotic resistant genes were obtained from ResFinder.

3.1.1 Genomes and genes

The genomes were downloaded through FTPs from the NCBI Reference Sequence Database (RefSeq) for genomes. For the species all available genome sequences at the time were downloaded and used in the analyses to get as much data as possible. The dates for downloading of the genomes were 2020-09-18 for *E. coli* and 2020-10-26 for *K. pneumoniae*. The number of genomes for each species can be seen in Table 3.1.

Table 3.1: Number of genomes downloaded from NCBI for the species.

Species	Number of genomes
E. coli	3 750 821
K. pneumoniae	1 237 986

3.1.1.1 ResFinder

The antibiotic resistant genes were obtained from ResFinder, a database using BLAST to identify acquired antibiotic resistant genes or to find chromosomal mutations mediating antibiotic resistance in input sequences from bacteria [12]. To this project the FASTA files of the antibiotic resistant genes were downloaded (2020-08-31) and then used [13].

ResFinder contained 2156 genes in 15 classes, listed below in Table 3.2. Note there are a total of 2134 unique genes and an additional 19 genes existing in two or three classes. For example, the gene cfr_1_AM408573 exists in both macrolides, oxazolidinones and phenicols.

Table 3.2: The number of antibiotic resistant genes from each antibiotic class obtained from the database ResFinder.

Antibiotic class	Number of genes
Aminoglycosides	169
Beta-lactams	1309
Colistin	3
Fosfomycin	21
Fusidic acid	2
Macrolides	131
Nitroimidazoles	14
Oxazolidinones	16
Phenicols	50
Quinolones	103
Rifampicin	9
Sulphonamides	49
Tetracyclines	104
Trimethoprim	53
Vancomycin	123
Total	2156

3.1.2 Core genomes

The core genome is defined as the set of genes present in the vast majority of the strains of a species. Usually the genome includes the essential genes, with genes for translation, transcription and metabolism [14]. Study the core genome genes could show if there are some features those genes have in common related to gene compatibility, as all genes are existing in all strains of the species.

The core genomes used in this project were obtained from Anna Johnning. The genes in the core genome were present in more than 95% of the input species data and the number of genes in the used core genomes can be seen in Table 3.3.

Table 3.3: Number of genes in the core genomes for the species.

Species	Number of genes
E. coli	1656
K. pneumoniae	2798

3.2 Analyses

In the project, three types of analyses were done; kmer analysis, frequency analysis and analysis of regions surrounding the gene. All used scripts were written in Python in the Linux terminal and example scripts of each analysis can be found in the Appendix A.1.1. The workflow of the analysis methods can be seen in Figure 3.1, the same data and methods were used in multiple of the analyses.

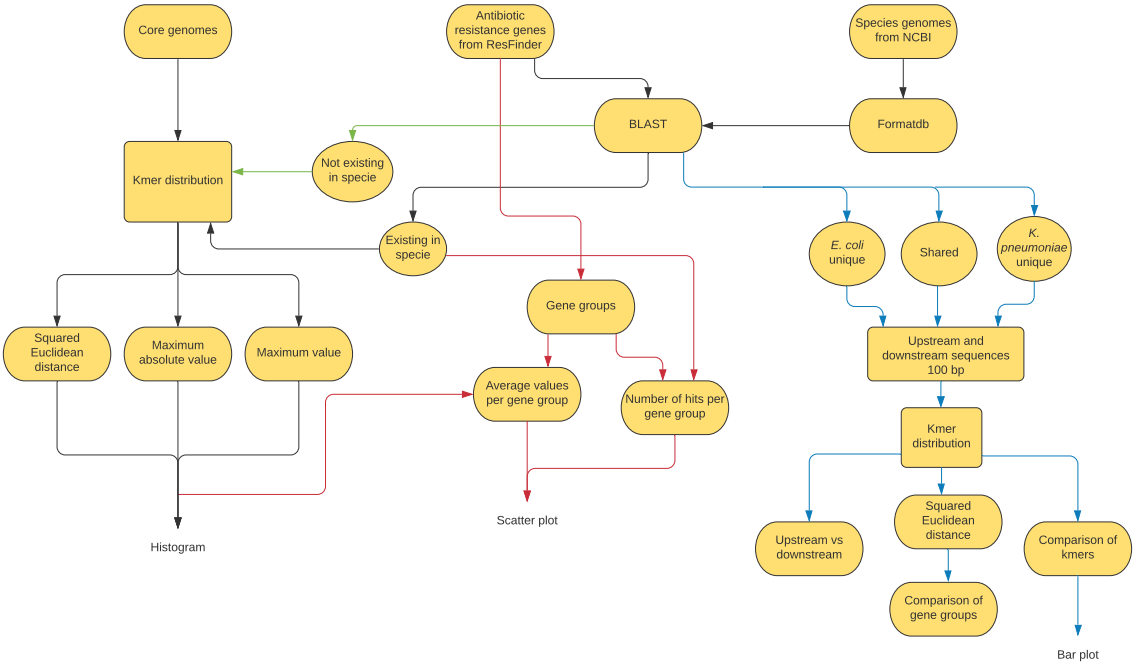


Figure 3.1: Workflow of the different analysis methods. Steps used in two or three analyses have black arrows, steps in kmer analysis have green arrows, steps in the frequency analysis have red arrows and steps in analysis of regions surrounding the gene have blue arrows.

3.2.1 Pre-processing of data

For each species, the core genome was provided in a FASTA file together with a file containing the gene positions. For each gene, the positions were used to cut the gene from the core genome, removing the gaps and then written into a new FASTA file.

A database per species was created from all the downloaded genomes from NCBI RefSeq using the tool formatdb for nucleotides. The database and the antibiotic resistant genes from ResFinder were run in BLAST to find which genes that were present in the species. To be considered as a hit, the conditions identity >95% and a length >75% were required to be fulfilled.

3.2.2 Kmer analysis

In the kmer analysis the two species *E. coli* and *K. pneumoniae* were run separately.

The BLAST result was used to separate the antibiotic resistant genes into two groups, genes present in the species and genes not present in the species. The number of genes in each group can be seen in Table 3.4.

Table 3.4: Number of antibiotic resistant in the two groups, genes present and not present in the species.

Species	Genes present in species	Genes not present in species
E. coli	1122	1012
K. pneumoniae	1174	961

For kmer of lengths from 1 to 8, three groups of genes, the antibiotic resistant genes present respectively not present in the species and the core genome genes, had their kmer distribution calculated by taking the frequency of each kmer in the gene and divide by the total number of kmers to get the fraction as the value. From all of the core genome genes also the kmer distribution of the average core genome gene was calculated.

The kmer distribution of the genes from each of the three groups were compared to the kmer distribution of the average core genome genes to calculate values with three different methods described later in this chapter; squared Euclidean distance, maximum absolute value and maximum value. This gave a value per gene and the values were visualized with histograms.

For each group of values the median value was obtained and the non-parametric Wilcoxon rank-sum test was used at all values from two groups at a time to see how the distribution of values differed, resulting in p-values [9].

3.2.2.1 Squared Euclidean distance

The squared Euclidean distance values were calculated for each gene as the sum of the squared value of the difference between the kmer distribution of the average core genome gene and the gene. The squared Euclidean distance takes all kmers into consideration, for kmer of length 3, see Equation 3.1.

$$squared\ Euclidean\ distance_{gene} = \sum_{i=AAA,AAC....TTT} (core\ genome_i - gene_i)^2 \quad (3.1)$$

3.2.2.2 Maximum absolute value

The maximum absolute values were calculated for each gene as the maximum absolute value of the difference between the kmer distribution of the average core genome gene and the gene, for the equation for kmer of length 3, see Equation 3.2.

$$maximum\ absolute\ value_{gene} = max|core\ genome_i - gene_i|_{i=AAA,AAC....TTT} \quad (3.2)$$

3.2.2.3 Maximum value

The maximum values were calculated for each gene as the positive difference of the kmer distribution values of the most common kmer in the average core genome gene and the most common kmer of the gene, for the equation for kmer of length 3, see Equation 3.3.

$$maximum\ value_{gene} = |max(core\ genome_i) - max(gene_i)|_{i=AAA,AAC....TTT} \quad (3.3)$$

3.2.3 Frequency analysis

To investigate whether the number of hits from the BLAST results had some correlations with the values from the kmer analysis a frequency analysis was done. Also in this analysis the two species *E. coli* and *K. pneumoniae* were run separately, but only the genes present in the species were used.

The number of hits in the BLAST result for the genes were counted. Some of the genes with only one or few different nucleotide would give the same number of hits and to avoid counting the same hits multiple times, the antibiotic resistant genes were merged into fewer group, for example the genes aac(3)-Ia, aac(3)-Ib and aac(3)-Ic were grouped in the new group aac(3)-I. Also, when hits overlapped with more than 50% only one hit was counted. It resulted in gene groups and the number of hits per groups. The number of gene groups used for each species from the present genes can be seen in Table 3.5.

Table 3.5: Number of gene groups used for each species from the present genes.

Species	Number of gene groups
E. coli	182
K. pneumoniae	240

For the kmer of lengths 1 to 8, the values from the three kmer analysis methods were used, for each gene group the average value was plotted against the square root of the number of hits in a scatter plot. Each antibiotic class were shown with different colors. To see the trend of the points a local regression smoothing method called Locally Weighted Scatterplot Smoothing (lowess) line was added to all the plots of the analysis to fit a function through the points [15].

3.2.4 Analysis of regions surrounding the gene

The analysis of regions surrounding the gene was done for *E. coli* and *K. pneumoniae* at the same time to investigate if surroundings of the genes could be of importance of the gene compatibility between the two species.

Based on the BLAST result the antibiotic genes were divided into three groups; genes that only were present in *E. coli* (called *E. coli* unique), genes that only were present in *K. pneumoniae* (called *K. pneumoniae* unique) and genes that were present in both of the two species (called shared). The number of genes for each group were 91 genes, 146 genes and 1001 genes but in the analyses the shared groups were done with randomly selected genes, 93 genes for *E. coli* and 143 genes for *K. pneumoniae*, different genes for each species, to more correspond to the number of genes in the unique groups.

For each gene, from the hits in the BLAST file the gene positions were used to cut a sequence of 100 base pairs upstream and downstream of the gene in the reference sequences found in the database. The sequences from the same shared gene for *E. coli* and *K. pneumoniae* were kept apart in two separate groups. The cut sequences had their kmer distribution calculated for kmer of lengths 1 to 7.

3.2.4.1 Upstream and downstream

An analysis was done to see whether there was a difference between the upstream and downstream sides of the genes. The sequences from the upstream and downstream sides were split into two quantities per group for the unique and shared groups. All sequences were converted to kmers and the total kmer distribution was calculated for each quantity. Then for each group the upstream and downstream quantities were compared using the non-parametric Wilcoxon signed-rank test to test if there were any statistical differences between the upstream and downstream sides [9].

3.2.4.2 Comparison of gene groups

In the next analysis the unique groups from before where used but the shared group was split into two groups, one each for the species. Then comparisons were done between the kmer distributions between the groups of genes:

E. coli unique vs *K. pneumoniae* unique
E. coli unique vs *E. coli* shared
E. coli shared vs *K. pneumoniae* shared
K. pneumoniae unique vs *K. pneumoniae* shared

The sequences from each group were converted to the total kmer distribution for each group and then the kmer distributions for the groups were compared with the method squared Euclidean distance, using the Equation 3.1, to get a value for each comparison.

3.2.4.3 Comparison of kmers

Not only to look at the groups in total, but an analysis was also done to look at the single kmers. The difference of kmer distribution value for each kmer was calculated between the unique respectively shared groups; *E. coli* unique vs *K. pneumoniae* unique and *E. coli* shared vs *K. pneumoniae* shared. A positive value meant a higher kmer distribution value in *E .coli*, while a negative value meant a higher kmer distribution in *K. pneumoniae*. From the two differences a third dataset was produced as the difference of the two previously data sets ((*E. coli* unique vs *K. pneumoniae* unique) vs (*E. coli* shared vs *K. pneumoniae* shared)), this to scale the numbers by remove the differences from the shared genes. The values from the third data set were sorted by absolute values and the top kmers were plot in a bar plot.

3.3 The predictive model

To investigate the prediction ability of classification, a logistic regression model was used. The model was done for *E. coli* in R with generalised linear model with the settings glm(response variable∼predictors, data=input data, family="binomial"). The input data was obtained from the previous analyses, the classification parameter if the antibiotics resistant genes were present or not present in *E. coli* based on the BLAST results, length of the gene and the kmer distribution values of kmers of length 3 of each gene. Out of the kmers the last kmer TTT was removed to avoid over-parameterization, resulting in 63 kmers. The genes members of several classes were removed, resulting in 2115 genes.

To create a model the binary response variable 0/1 (present/not present in *E.coli*) and the lengths of all genes were fixed in a loop where the kmer with the lowest p-value for each round was added into the model until the threshold 0,001 was exceeded. The model was used in the next step where the data was randomly split in half training and half testing data and for different cutoff values for the classification the model was run 100 times. Each round generated values of the specificity and sensitivity and for the total model, the average values were used and were plotted with a ROC curve.

3.3.1 All genes

In the first model all 2115 genes were used, where 1117 genes were present and 998 genes were not present in *E. coli*. An example script can be found in Appendix A.1.2.

3.3.2 Antibiotic classes

Models were also done for the genes from the larger antibiotic classes, seen in Table 3.6, resulting in three models. For these models the kmers were added until the p-value of 0,01 was exceeded, to include more kmers.

Table 3.6: Antibiotic classes and the number of genes used in the models.

Antibiotic class	Total genes	Present in *E. coli*	Not present in *E. coli*
Beta-lactams	1309	773	536
Aminoglycosides	166	72	94
Macrolides	129	40	89

4

Results

In this chapter the results from the project are presented in one section each for the three analyses and the predictive model. The sections are divided into subsections for the species and methods for kmer analysis and frequency analysis, the different comparisons for analysis of regions surrounding the gene and for all genes and the antibiotic classes for the predictive model.

4.1 Kmer analysis

In the kmer analysis three groups of genes, the antibiotic resistant genes present respectively not present in the species and the core genome genes, had their kmer distribution calculated. From all core genome genes the kmer distribution of the average core genome gene was calculated and compared to the kmer distribution of the genes from each of the three groups. It was done with three different methods to give a value per gene. For each group the median value was calculated and the Wilcoxon rank-sum test was used at two groups at a time to get a p-value and to see how the distribution of values differed.

The most interesting results from the three methods squared Euclidean distance, maximum absolute value and maximum values are presented first for *E. coli* and then for *K. pneumoniae*.

4.1.1 *Escherichia coli*

In this section the most interesting results for *E. coli* are presented. Additional plots can be seen in Appendix A.2.

4.1.1.1 Squared Euclidean distance

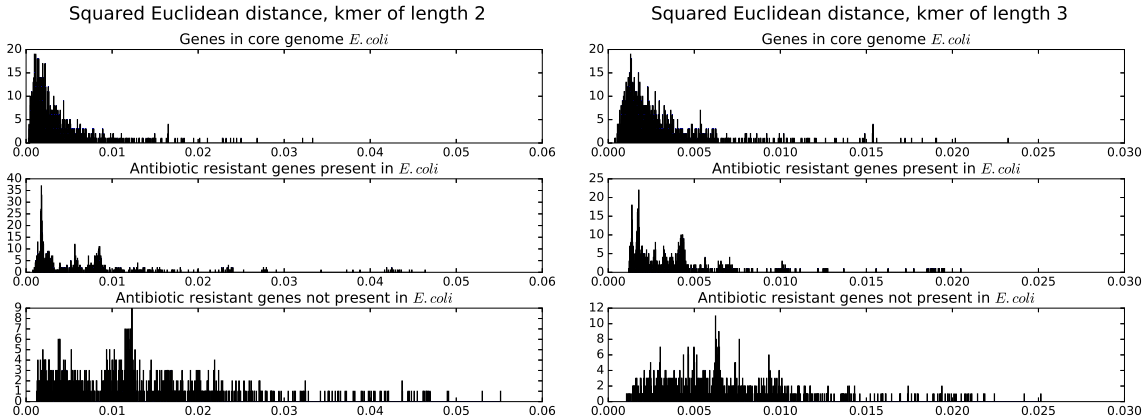
For the method squared Euclidean distance, the values were calculated for each gene as the sum of the squared value of the difference between the kmer distribution of the average core genome and the gene. The squared Euclidean distance took all kmers into consideration to give a value per gene in the three different groups. The median values from each of the three groups and p-values for the comparisons between the groups for kmer of lengths 1 to 8 are presented in Table 4.1.

Table 4.1: The median values from the calculation with the method squared Euclidean distance between the average core genome gene and the genes from three groups, core genome genes, antibiotic resistant genes present and not present in *E. coli* and the p-values for the comparisons between the groups.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,001993	0,005467	0,01815	5,43e-100	5,61e-59	8,54e-256
kmer 2	0,002910	0,004300	0,01168	8,49e-101	1,30e-53	6,91e-242
kmer 3	0,002079	0,003100	0,006169	1,90e-107	1,33e-36	3,41e-204
kmer 4	0,001782	0,002156	0,003225	2,43e-107	1,00e-07	7,09e-108
kmer 5	0,001622	0,001607	0,001939	5,41e-87	2,61e-04	1,09e-16
kmer 6	0,001562	0,001335	0,001445	2,93e-36	1,71e-19	2,96e-03
kmer 7	0,001547	0,001229	0,001307	2,68e-17	1,89e-29	3,55e-17
kmer 8	0,001542	0,001172	0,001259	9,03e-12	8,25e-34	1,09e-25

For the shorter kmer lengths, the difference of median values between the three groups was larger and with increasing kmer length the difference decreased. From kmer of length 5 the values of the groups were more similar with more or less the same median values.

The small p-values indicated significant differences between all three groups, with a larger difference between the core genome genes and genes not present in *E. coli* than the difference between the core genome genes and genes present in *E. coli*. The largest differences were for kmer of lengths 2 and 3, the values for all genes in the three groups can be seen in Figure 4.1.



(a) Squared Euclidean distance, kmer of length 2.

(b) Squared Euclidean distance, kmer of length 3.

Figure 4.1: Squared Euclidean distance values for *E. coli*, kmer of lengths 2 and 3. On the x-axis there are the values from the calculation with the method square Euclidean distance between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The distribution of the core genome genes looked similar in both figures for the two kmer lengths. The values of the antibiotic resistant genes present in *E. coli* were more similar to the core genomes genes than the antibiotic resistant genes not present in the *E. coli*, which had more spread values.

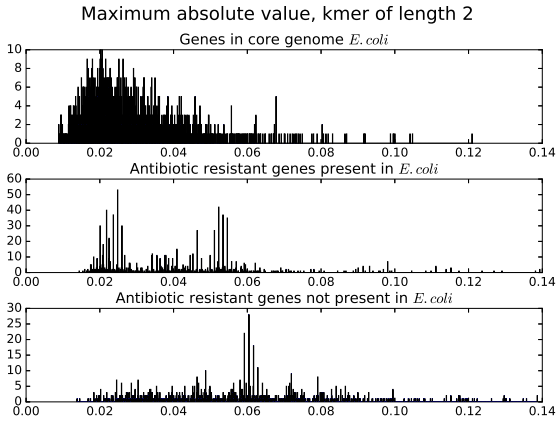
4.1.1.2 Maximum absolute value

For the method maximum absolute values, the values were calculated for each gene in the three different groups as the maximum absolute value of the difference between the kmer distribution of the average core genome gene and the gene. The median values from each of the three groups and p-values for the comparisons between the groups for kmer of lengths 1 to 8 are presented in Table 4.2.

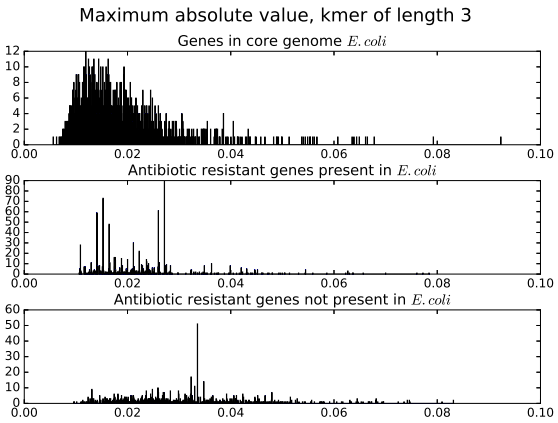
Table 4.2: The median values from calculation with the method maximum absolute value between the average core genome gene and the genes from three groups, core genome genes, antibiotic resistant genes present and not present in *E. coli* and the p-values for the comparisons between the groups.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,03320	0,04834	0,08812	1,75e-106	5,70e-40	7,34e-238
kmer 2	0,02550	0,03769	0,05657	1,41e-76	4,44e-63	1,54e-218
kmer 3	0,01625	0,02090	0,02955	3,09e-72	1,57e-49	7,90e-179
kmer 4	0,01044	0,01182	0,01562	4,61e-53	8,36e-18	1,97e-105
kmer 5	0,007174	0,006752	0,008454	2,87e-52	2,15e-06	2,37e-23
kmer 6	0,005096	0,004646	0,005243	8,87e-08	2,87e-04	0,270
kmer 7	0,003875	0,003418	0,003596	1,91e-03	2,95e-12	1,02e-06
kmer 8	0,003098	0,002320	0,002621	1,71e-25	2,93e-45	4,48e-15

For the method maximum absolute value, the trends were similar to the squared Euclidean distance. The median values of the shorter kmer lengths differed and the p-values were low between all groups. Also, here from kmer of length 5 the groups started to have similar median values.



(a) Maximum absolute value, kmer of length 2.



(b) Maximum absolute value, kmer of length 3

Figure 4.2: Maximum absolute values for *E. coli*, kmer of lengths 2 and 3. On the x-axis there are the values from the calculation with the method maximum absolute value between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The distribution of the core genome genes was similar in both figures. The values of the antibiotic resistant genes present in *E. coli* respectively not present in *E. coli* were more spread than for the squared Euclidean values, but the genes present in *E. coli* had their median values more similar to the core genome genes.

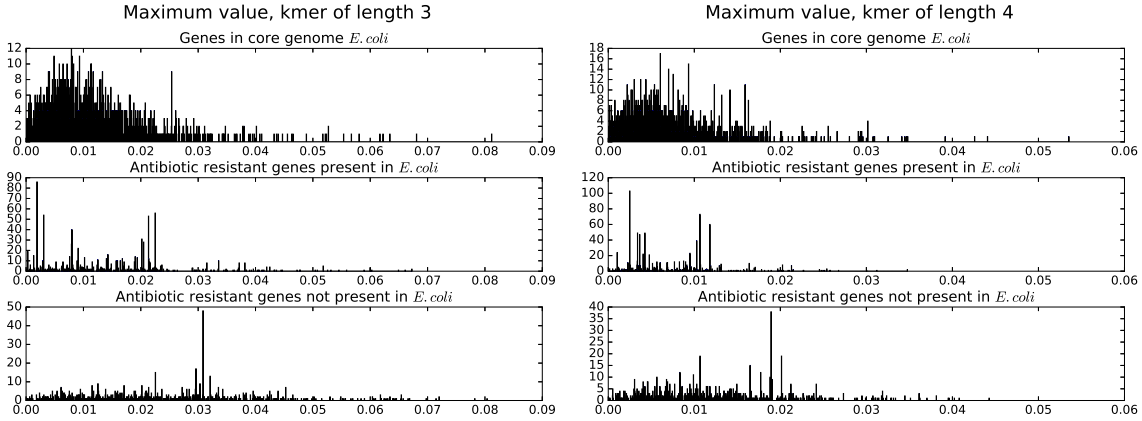
4.1.1.3 Maximum value

For the method maximum value, the values were calculated for each gene in the three different groups as the positive difference of the kmer distribution values of the most common kmer in the average core genome gene and the most common kmer of the gene. The median values from each of the three groups and p-values for the comparisons between the groups for kmer of lengths 1 to 8 are presented in Table 4.3.

Table 4.3: The median values from the calculation with the method maximum values between the average core genome gene and the genes from three groups, core genome genes, antibiotic resistant genes present and not present in *E. coli* and the p-values for the comparisons between the groups.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,01322	0,02454	0,05441	2,73e-82	4,84e-47	4,27e-193
kmer 2	0,01143	0,02005	0,04514	1,36e-36	1,70e-63	2,48e-175
kmer 3	0,009873	0,01308	0,02253	2,81e-57	8,10e-10	3,79e-122
kmer 4	0,005858	0,005844	0,01066	1,81e-55	0,2203	4,33e-72
kmer 5	0,004716	0,004176	0,005464	5,17e-40	7,13e-15	3,56e-10
kmer 6	0,003913	0,003338	0,004016	1,13e-05	6,65e-06	0,2993
kmer 7	0,003339	0,002832	0,002986	6,73e-02	6,91e-14	3,92e-09
kmer 8	0,002866	0,002073	0,002358	9,17e-24	2,27e-47	2,35e-16

For the method maximum value, the trends of the median and p-values were similar to the two previously described methods. The median values of the shorter kmer lengths differed and the p-values were low between all groups, though the differences between the groups were smaller. For kmer of length 4 the comparison between the core genome genes and the genes present in *E. coli* the p-value of 0,2203 indicated no statistically significant difference between the groups. Already from kmer of length 4 the groups started to have similar values. All values from kmer of lengths 3 and 4 can be seen in Figure 4.3.



(a) Maximum value, kmer of length 3. **(b)** Maximum value, kmer of length 4.

Figure 4.3: Maximum values for *E. coli*, kmer of lengths 3 and 4. On the x-axis there are the values from the calculation with the method maximum value between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The values from antibiotic resistant genes present in *E. coli* respectively not present in *E. coli* were more spread than previous methods, with higher peaks at values

up to the frequency of 100. Despite the spread, the median values of the genes present in *E. coli* can be seen in the figures to be more similar to the core genome, than the ones not present. The p-value of 0,2203 for kmer of length 4 between the core genome genes and the genes present can be seen to be explained by the similar distributions of the values.

4.1.2 *Klebsiella pneumoniae*

In this section the most interesting results for *K. pneumoniae* are presented. Additional plots can be seen in Appendix A.2.

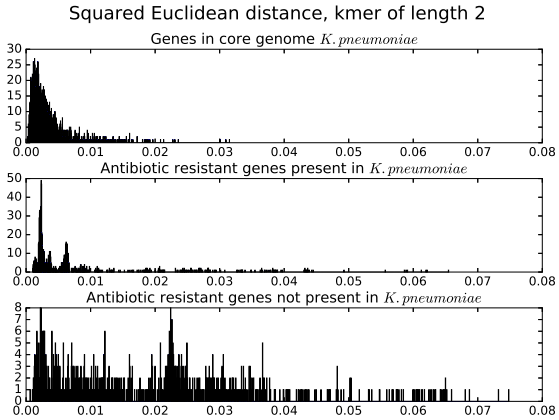
4.1.2.1 Squared Euclidean distance

Same as for *E. coli*, the method squared Euclidean distance took all kmers into consideration and the distribution were compared to the average core genome gene to calculate a value for each gene in the three different groups. The median and p-values for *K. pneumoniae* for kmer of lengths 1 to 8 are presented in Table 4.4.

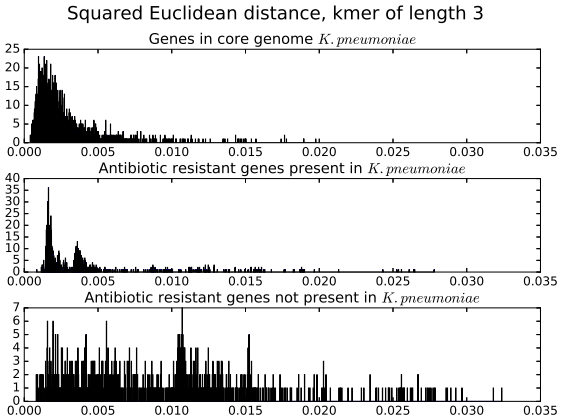
Table 4.4: The median values from the calculation with the method squared Euclidean distance between the average core genome gene and the genes from three groups, core genome genes, antibiotic resistant genes present and not present in *K. pneumoniae* and the p-values for the comparisons between the groups.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,002429	0,003782	0,0326	1,42e-92	4,93e-73	3,12e-262
kmer 2	0,002385	0,003456	0,01904	2,91e-103	3,89e-75	1,04e-283
kmer 3	0,001967	0,0026681	0,008813	1,37e-115	6,34e-50	1,61e-272
kmer 4	0,007209	0,009918	0,01503	1,37e-116	8,61e-25	3,96e-210
kmer 5	0,001437	0,001461	0,002417	1,66e-103	9,28e-03	5,78e-91
kmer 6	0,001350	0,001265	0,001604	7,14e-53	2,92e-02	9,57e-15
kmer 7	0,001315	0,001204	0,001361	8,88e-22	9,91e-06	8,96e-01
kmer 8	0,001304	0,001184	0,001279	5,41e-12	1,08e-07	5,41e-04

The patterns of the median values and p-values could be recognized from *E. coli*. For the shorter kmer lengths, the difference of median values between the three groups was larger and with increasing kmer length the difference decreased. But while the values of core genome genes and genes present in *K. pneumoniae* were similar, the difference to the genes not present in the species was almost 10-fold for kmer of length 1. Not until the kmer of length 6 the median values for all three groups started to be similar. This was also supported by the p-values, there was a larger difference between the genes not present in the species and the core genome genes than the difference between the genes present in the species and the core genome genes. Plots of the values for kmer of lengths 2 and 3 can be seen in Figure 4.4.



(a) Squared Euclidean distance, kmer of length 2.



(b) Squared Euclidean distance, kmer of length 3.

Figure 4.4: Squared Euclidean distance values for *K. pneumoniae*, kmer of lengths 2 and 3. On the x-axis there are the values from the calculation with the method square Euclidean distance between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The distribution of the core genome genes looked similar in the plots for kmer of lengths 2 and 3, where there can be see that the antibiotic resistant genes not present in the species had a large spread in values. Even though the spread of antibiotic resistant genes present in *K. pneumoniae* had a larger spread than the core genome genes, the peaks were around the same values.

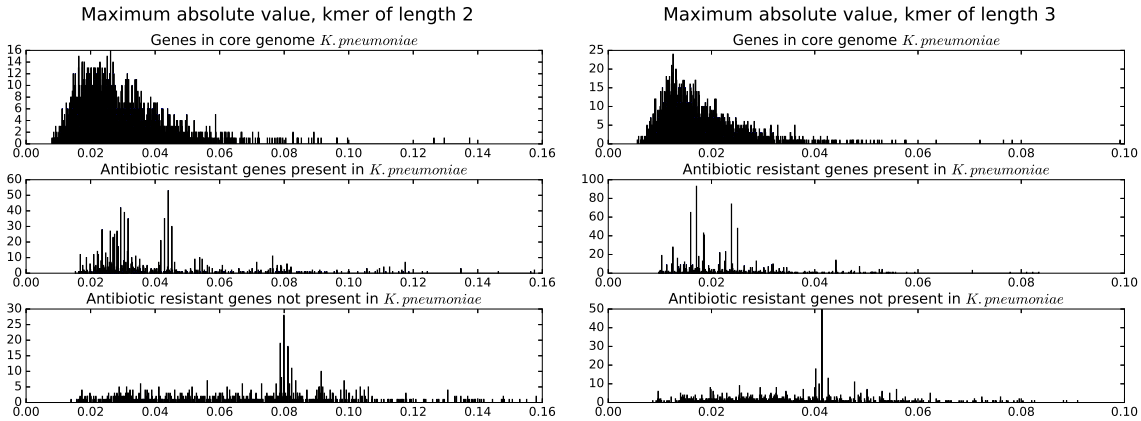
4.1.2.2 Maximum absolute value

In the method maximum absolute value, the values were calculated for each gene as the maximum absolute value of the difference of the average core genome gene and for each gene for the three different groups. The median and p-values for kmer of lengths 1 to 8 are presented in Table 4.5.

Table 4.5: The median values from the calculation with the method maximum absolute value between the average core genome gene and the genes from three groups, core genome genes, antibiotic resistant genes present and not present in *K. pneumoniae* and the p-values for the comparisons between the groups.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,03654	0,04404	0,11675	8,72e-101	4,08e-47	2,64e-246
kmer 2	0,02579	0,03258	0,06902	2,14e-101	1,06e-88	1,72e-281
kmer 3	0,01601	0,01974	0,03328	8,57e-105	4,86e-72	1,01e-258
kmer 4	0,01062	0,01160	0,01605	8,20e-110	2,28e-16	1,26e-164
kmer 5	0,007116	0,005838	0,008603	2,16e-119	4,19e-28	2,03e-42
kmer 6	0,005108	0,004565	0,005246	2,91e-13	2,08e-09	2,54e-01
kmer 7	0,003824	0,003423	0,003655	5,24e-05	1,30e-16	2,78e-06
kmer 8	0,003010	0,002325	0,002583	2,45e-18	1,95e-54	4,99e-16

For the method maximum absolute value, the trends were similar to the squared Euclidean distance. The median values for shorter kmer lengths differed and the p-values were low between all groups. From kmer of length 6 the groups started to have similar median values. The values from kmer of lengths 2 and 3 can be seen in Figure 4.5.



(a) Maximum absolute value, kmer of length 2.

(b) Maximum absolute value, kmer of length 3.

Figure 4.5: Maximum absolute values for *K. pneumoniae*, kmer of lengths 2 and 3. On the x-axis there are the values from the calculation with the method maximum absolute value between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The distribution of the core genome genes was similar in both figures. The values of the antibiotic resistant genes present in *K. pneumoniae* respectively not present in *K. pneumoniae* were more spread and values consisted of many higher peaks. It could be seen that the genes the core genome values were more similar to the genes present in *K. pneumoniae* than to the genes not present in *K. pneumoniae*.

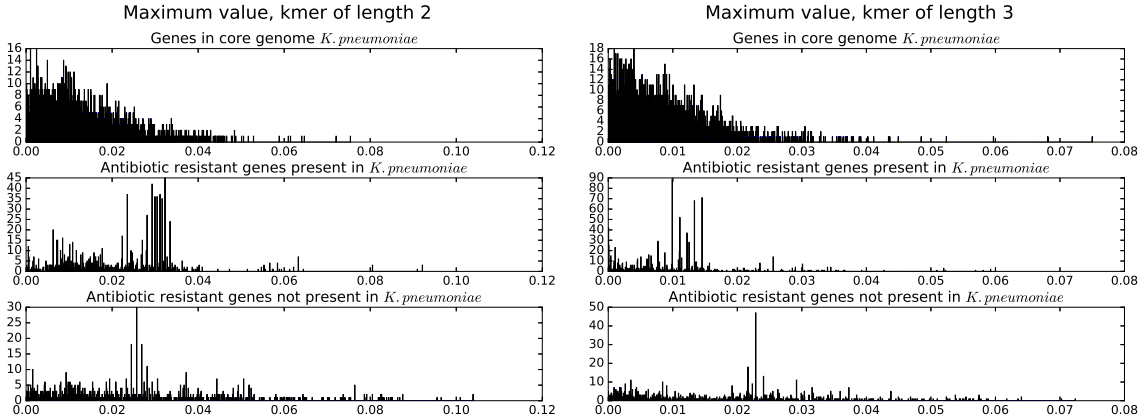
4.1.2.3 Maximum value

In the method maximum value, the values were calculated for each gene as positive difference of the most common kmer in average core genome gene and the genes in the three different groups. The median and p-values for kmer of lengths 1 to 8 are presented in Table 4.6.

Table 4.6: Median and p-values for antibiotic resistant genes present respectively not present in *K. pneumoniae* and core genome genes from the method maximum value.

	Median values			P-values		
	Core genome	Present	Not present	Present vs not present	Present vs core genome	Not present vs core genome
kmer 1	0,01528	0,02242	0,03288	2,77e-30	2,86e-41	1,18e-104
kmer 2	0,01059	0,02236	0,02328	4,00e-03	2,01e-94	9,93e-82
kmer 3	0,007209	0,009918	0,01503	4,56e-31	2,69e-06	1,14e-60
kmer 4	0,004737	0,003178	0,006494	2,07e-51	9,68e-34	1,79e-16
kmer 5	0,003822	0,002195	0,003794	1,16e-47	1,17e-91	1,28e-01
kmer 6	0,003532	0,00246	0,003097	8,79e-06	7,58e-41	2,19e-14
kmer 7	0,002876	0,002231	0,002496	9,29e-04	2,55e-40	3,46e-19
kmer 8	0,002594	0,001831	0,002102	1,66e-18	1,43e-65	4,03e-21

For the method maximum value, the values had the least differences of the methods described. The similar trends could still be seen for some groups, the core genome and genes not present were different. For kmer of length 2 the genes present and not present had similar median value and p-value, for kmer of length 3 the values again differed and for the rest of kmers the values did not fully stabilize around the same median as the previous methods. The values from kmer of lengths 2 and 3 can be seen in Figure 4.6.



(a) Maximum value, kmer of length 2. **(b)** Maximum value, kmer of length 3.

Figure 4.6: Maximum values for *K. pneumoniae*, kmer of lengths 2 and 3. On the x-axis there are the values from the calculation with the method maximum value between the average core genome gene and the genes from the three groups and on the y-axis the frequency.

The values consisted of many high peaks, the antibiotic resistant genes present in *K. pneumoniae* had more peaks at several values while the genes not present had values where the peaks were more distinct. There could not be clearly seen that the core genome values were more similar to the genes present in the species than to the genes not present in the specie, compared to in the other methods and in *E. coli*.

4.2 Frequency analysis

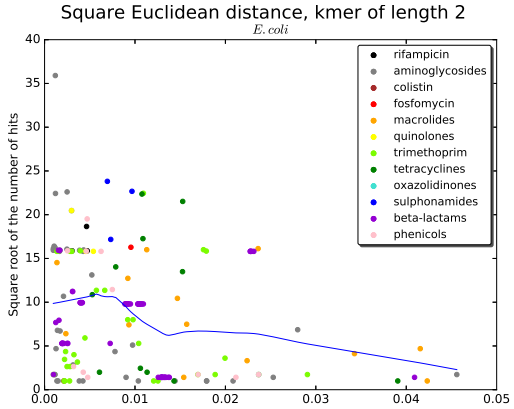
In the frequency analysis the values from the antibiotic resistant genes present in the species from the kmer analysis were used. This to see whether there could be some correlations between the values and the number of hits for the gene groups. The results are presented first for *E. coli* and then for *K. pneumoniae*.

4.2.1 *Escherichia coli*

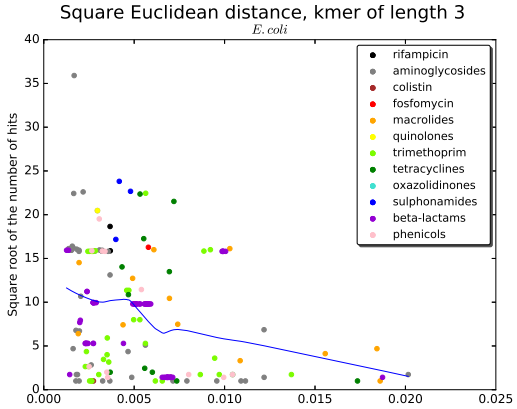
In this section the most interesting results for *E. coli* are presented. Additional plots can be seen in Appendix A.2.

4.2.1.1 Squared Euclidean distance

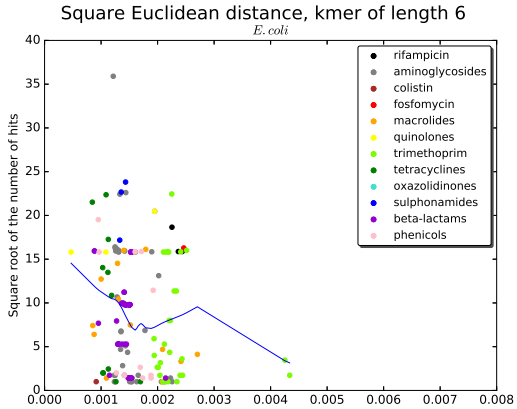
The square root of the number of hits was plotted against the values from the method squared Euclidean distance. The plots for kmer of lengths 2, 3 and 6 can be seen in Figure 4.7.



(a) Squared Euclidean distance, kmer of length 2, median value 0,0043.



(b) Squared Euclidean distance, kmer of length 3, median value 0,0031.



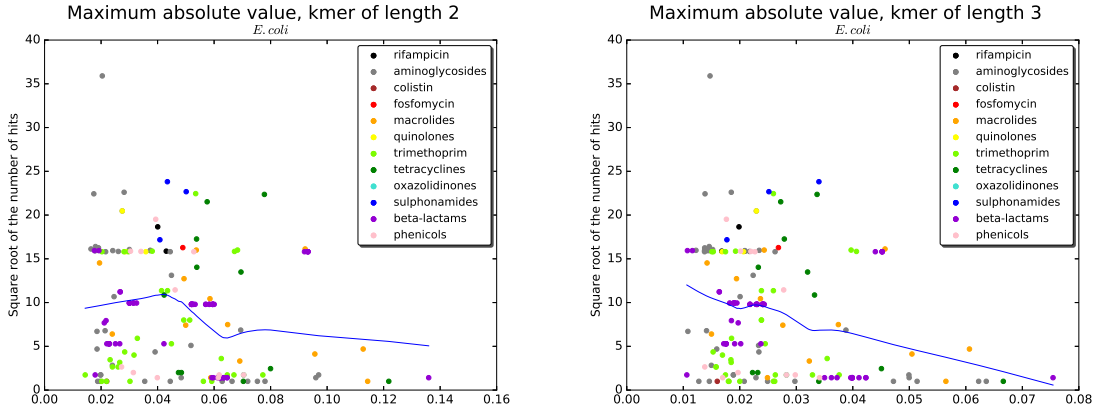
(c) Squared Euclidean distance, kmer of length 6, median value 0,001335.

Figure 4.7: Frequency analysis, squared Euclidean distance for *E. coli*, kmer of lengths 2, 3 and 6. On the x-axis there are the values from the kmer analysis and the method squared Euclidean distance for gene groups from the antibiotic resistant genes present in *E. coli* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

For kmer of lengths 2 and 3 the trend line went down away from the values around the median values. The values around the median values had large spread in hits but the values with higher values had fewer hits. There could not be seen any specific trends of the antibiotic classes. For kmer of length 6 the values were more compact and here the values for the same classes were more clustered, especially for beta-lactams. Though, there was still a large spread in the number of hits within each antibiotic class.

4.2.1.2 Maximum absolute value

The square root of the number of hits was plotted against the values from the method maximum absolute value. The plots for kmer of lengths 2 and 3 can be seen in Figure 4.8.



(a) Maximum absolute value, kmer of length 2, median value 0,03769.

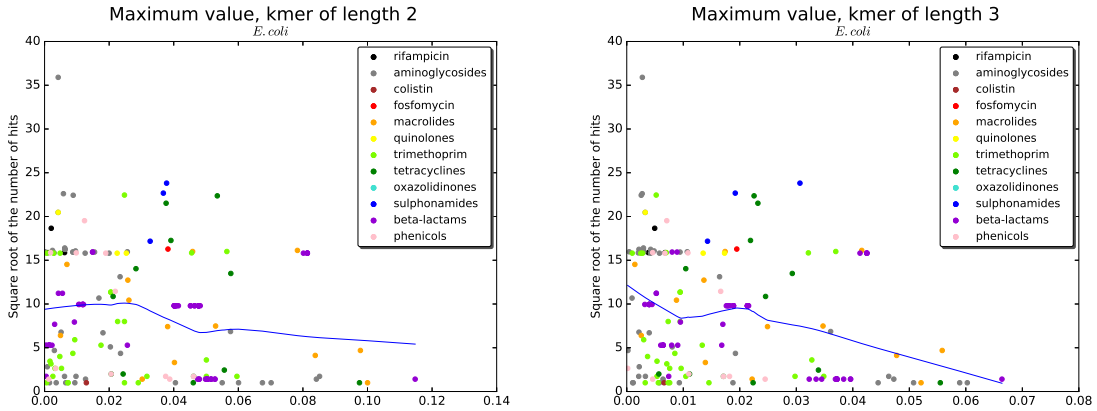
(b) Maximum absolute value, kmer of length 3, median value 0,0209.

Figure 4.8: Frequency analysis, maximum absolute values for *E. coli*, kmer of lengths 2 and 3. On the x-axis there are the values from the kmer analysis and the method maximum absolute value for gene groups from the antibiotic resistant genes present in *E. coli* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

Similar to squared Euclidean distance, kmer of lengths 2 and 3, the trend line went down away from the values around the median values. The values around the median values had large spread in hits but the values with higher values had fewer hits. There could not be seen any specific trends of the antibiotic classes.

4.2.1.3 Maximum value

The square root of the number of hits was plotted against the values from the method maximum value. The plots for kmer of lengths 2 and 3 can be seen in Figure 4.9.



(a) Maximum value, kmer of length 2, median value 0,02005.

(b) Maximum value, kmer of length 3, median value 0,01308.

Figure 4.9: Frequency analysis, maximum value for *E. coli*, kmer of lengths 2 and 3. On the x-axis there are the values from the kmer analysis and the method maximum value for gene groups from the antibiotic resistant genes present in *E. coli* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

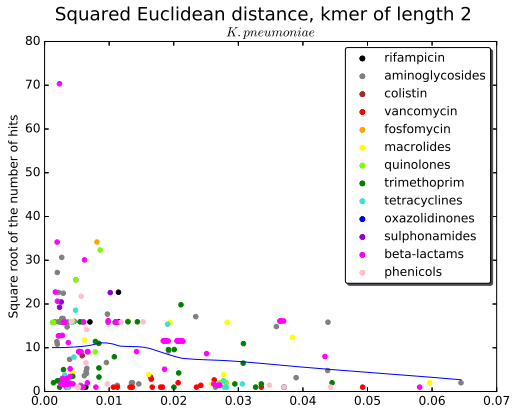
The plots looked similar to the previously methods with the trend line went down away from the values around the median values. The values around the median values had large spread in hits but the values with higher values had fewer hits. There could not be seen any specific trends of the antibiotic classes.

4.2.2 *Klebsiella pneumoniae*

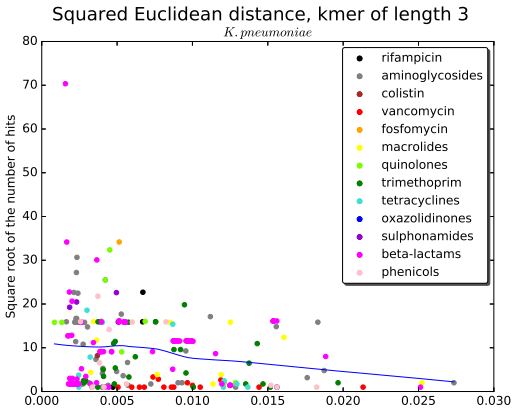
In this section the most interesting results for *K. pneumoniae* are presented. Additional plots can be seen in Appendix A.2.

4.2.2.1 Squared Euclidean distance

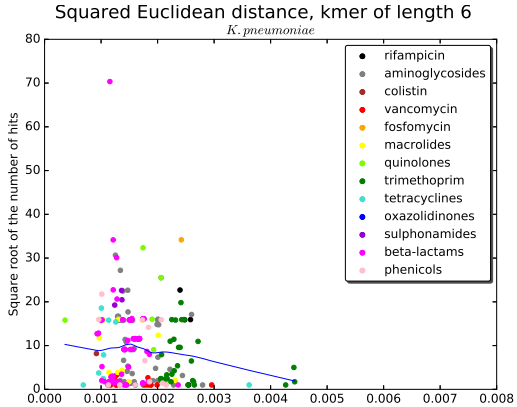
Similar as for *E. coli*, for *K. pneumoniae* the square root of the number of hits was plotted against the values fromthe method squared Euclidean distance, the plots for kmer of lengths 2, 3 and 6 can be seen in Figure 4.10.



(a) Squared Euclidean distance, kmer of length 2, median value 0,003456.



(b) Squared Euclidean distance, kmer of length 3, median value 0,002668.



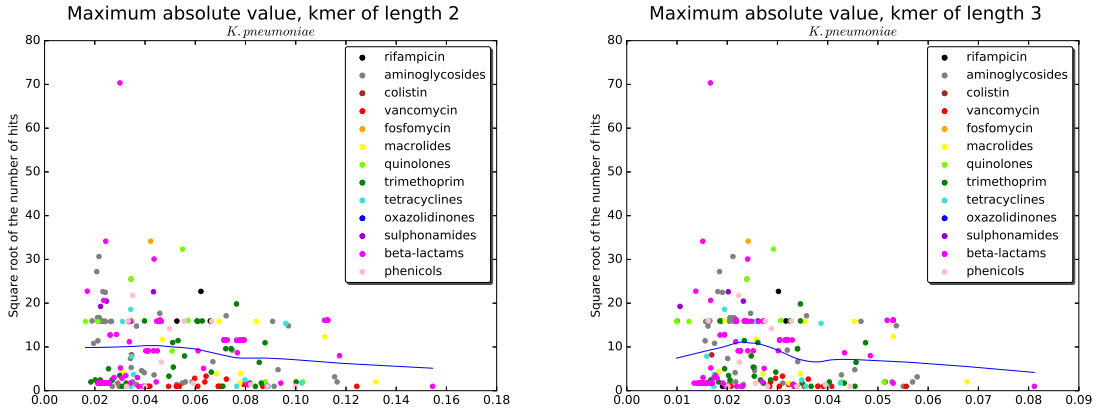
(c) Squared Euclidean distance, kmer of length 6, median value 0,001265

Figure 4.10: Frequency analysis, squared Euclidean distance for *K. pneumoniae*, kmer of lengths 2, 3 and 6. On the x-axis there are the values from the kmer analysis and the method squared Euclidean distance for gene groups from the antibiotic resistant genes present in *K. pneumoniae* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

In the plots the slope of the trend line decreased with increased value. The values around the median values had large spread in hits but the values with higher values had fewer hits. Compared to *E. coli* there were more groups over 25 at the y-axis and one group around 70. For the shorter kmers there could not be seen any specific trends of the antibiotic classes, except for vancomycin that seemed to all have low number of hits. For kmer of length 6 the values were more compact and here the values of the same classes were more clustered, especially for the resistance genes of beta-lactams, tetracyclines and most trimethoprimes. Though, there was still a large spread in the number of hits in those groups.

4.2.2.2 Maximum absolute value

The square root of the number of hits was plotted against the values from the method maximum absolute value. The plots for kmer of lengths 2 and 3 can be seen in Figure 4.11.



(a) Maximum absolute value, kmer of length 2, median value 0,03258.

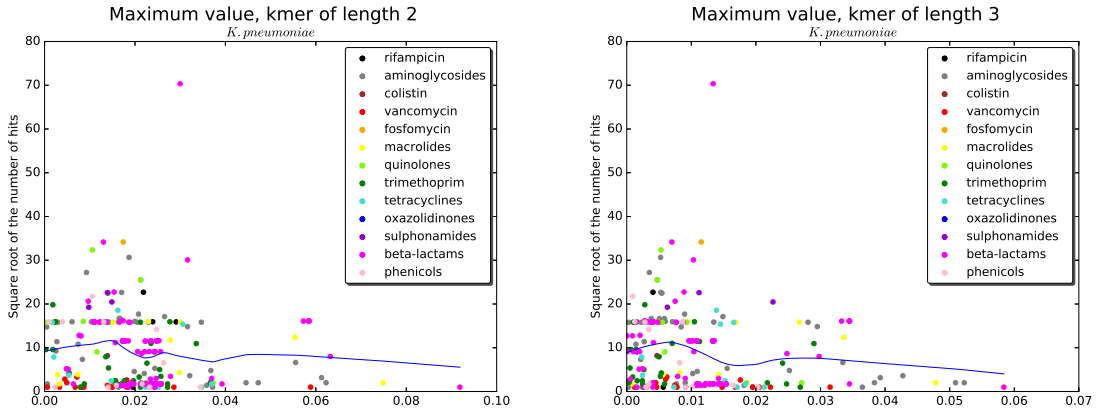
(b) Maximum absolute value, kmer of length 3, median value 0,01974.

Figure 4.11: Frequency analysis, maximum absolute value for *K. pneumoniae*, kmer of lengths 2 and 3. On the x-axis there are the values from the kmer analysis and the method maximum absolute value for gene groups from the antibiotic resistant genes present in *K. pneumoniae* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

Similar to the squared Euclidean distance, the slope of the trend line decreased with increased value. The values around the median values had large spread in hits but the values with higher values had fewer hits. For the shorter kmers there could not be seen any specific trends of the antibiotic classes, except again the class vancomycin that seems to all genes had low number of hits.

4.2.2.3 Maximum value

The square root of the number of hits was plotted against the values from squared Euclidean distance, the plots for kmer of lengths 2 and 3 can be seen in Figure 4.12.



(a) Maximum value, kmer of length 2, median value 0,02236

(b) Maximum value, kmer of length 3, median value 0,009918

Figure 4.12: Frequency analysis, maximum value for *K. pneumoniae*, kmer of lengths 2 and 3. On the x-axis there are the values from the kmer analysis and the method maximum value for gene groups from the antibiotic resistant genes present in *K. pneumoniae* and on the y-axis the square root of the number of hits for the gene groups. The lowess trend line can be seen in blue.

Similar to the previous methods, the slope of the trend line decreased with increased value. The values around the median values had large spread in hits but the values with higher values had fewer hits. For the shorter kmers there could not be seen any specific trends of the antibiotic classes, except for vancomycin where all genes had low number of hits. The values for maximum values were also closer to zero than for the other methods.

4.3 Analysis of regions surrounding the gene

In the analysis of regions surrounding the gene, the genes were divided into three groups, genes that only were present in *E. coli* (called *E. coli* unique), genes that only were present in *K. pneumoniae* (called *K. pneumoniae* unique) and genes that were present in both two species (called shared). The groups were used in comparisons between the upstream and downstream sides, comparisons of gene groups and comparison of kmers. The results from the analysis of regions surrounding the gene are presented for the species *E. coli* and *K. pneumoniae* simultaneously.

4.3.1 Upstream and downstream

In the first analysis of regions surrounding the gene the difference between the kmer distribution values of the sequences of the upstream and downstream sides were investigated for kmer of lengths 1 to 7 with the Wilcoxon sign-rank test. This to see if there could be seen a statistically difference between the two sides, the p-values for kmer of lengths 1 to 7 can be seen in Table 4.7.

Table 4.7: P-values of the comparisons between the values from kmer distribution of the upstream and downstream sequences of three groups of genes

	E. coli unique n=95	*K. pneumoniae* unique n=146	Shared n=93+143
kmer 1 (4 kmers)	too few values	too few values	too few values
kmer 2 (16)	1.0	0,8361	0,7960
kmer 3 (64)	0,743	0,7891	0,9520
kmer 4 (256)	0,5906	0,6715	0,9315
kmer 5 (1024)	0,8690	0,9321	0,9192
kmer 6 (4096)	0,9592	0,9921	0,6098
kmer 7 (16 384)	0,02897	0,001446	0,7431

From kmer of lengths 2 to 6 the p-values were high, the difference between the groups of upstream and downstream sides of the genes could not be statistically explained to be different. From kmer of length 7 there could be seen some difference between the two sides, but the difference was not large. The other analyses were done not separating those groups.

4.3.2 Comparisons of gene groups

In the next analysis the unique groups from before were used together with the shared group, which was split into two, one each for the species. Then comparison values were calculated with the squared Euclidean distance of the kmer distributions between the four groups in four combinations. If there would not exist any differences between the species and genes, the hypothesis of the values would be;

E. coli unique vs *K. pneumoniae* unique = difference of unique genes (δu)
E. coli unique vs *E. coli* shared$\approx$0
E. coli shared vs *K. pneumoniae* shared=difference of shared genes (δs)
K. pneumoniae unique vs *K. pneumoniae* shared$\approx$0

The differences between the two species δs and δu would be equal. The calculated values can be seen in Table 4.8.

Table 4.8: Values from comparison of two groups of genes with the method squared Euclidean distance for kmer of lengths 1 to 7.

	E. coli unique vs *K. pneumoniae* unique	*E. coli* unique vs *E. coli* shared	*E. coli* shared vs *K. pneumoniae* shared	*K. pneumoniae* unique vs *K. pneumoniae* shared
kmer 1	0,006336	0,001024	0,0006482	0,001891
kmer 2	0,003910	0,0008370	0,0003919	0,001077
kmer 3	0,002674	0,0006822	0,0002590	0,001116
kmer 4	0,001741	0,0004803	0,0001882	0,0008875
kmer 5	0,001273	0,0004197	0,0001511	0,0007186
kmer 6	0,001081	0,0003933	0,0001399	0,0006456
kmer 7	0,0009981	0,0003890	0,0001342	0,0006334

The values between the shared genes were low for all kmers. For *E. coli* unique and shared genes, the values were also low but not as low as the comparison of the two shared groups. The values for comparison of unique and shared genes in *K. pneumoniae* were also slightly higher than the same comparisons for *E. coli*. Then the values between the two groups of unique genes were distinctly higher for all kmers.

4.3.3 Comparisons of kmers

There could be seen a larger difference between the unique groups of the two species then between the shared groups and the next step was to look which single kmers contributed to the difference. First the difference of kmer distribution value for each kmer was calculated between the unique group (*E. coli* unique vs *K. pneumoniae* unique) and shared group (*E. coli* shared vs *K. pneumoniae* shared). A positive value means a higher kmer distribution value in *E .coli*. From the two differences a third dataset was produced as the difference of the two previously data sets ((*E. coli* unique vs *K. pneumoniae* unique) vs (*E. coli* shared vs *K. pneumoniae* shared)), this to scale the numbers by remove the differences from the shared genes. A positive value means a higher kmer distribution value in *E .coli* The values from the third data set were sorted by absolute values and the top kmers were plot. The kmer comparisons of kmer of length 1 can be seen in Figure 4.13.

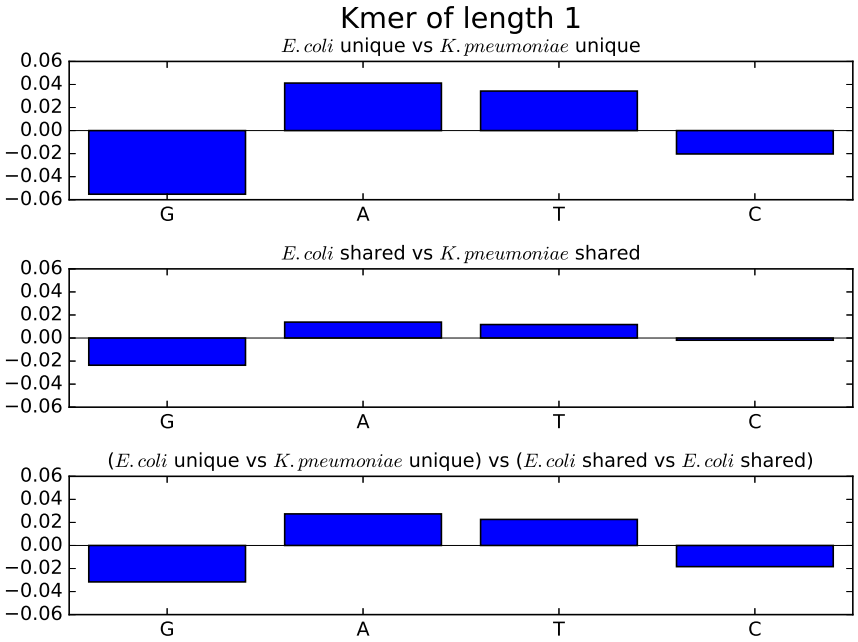


Figure 4.13: Comparison of kmers, kmer of length 1. The upper bars are the difference between the unique groups, the middle bars the difference of the shared groups and the lower bars the difference between the two previous differences. On the x-axis are the kmers and on the y-axis are the values from the difference of the kmer distributions, for the two upper bars a positive value means a higher kmer distribution value in *E .coli*. All values are sorted by the absolute value of the kmers in the third dataset.

Kmer of length 1, represents the GC content, from the figure there was a higher GC content in *K. pneumoniae* in both the comparisons with unique and shared genes, though the difference was not as large in the shared. In the total plot, the was still seen that the GC level was higher in *K. pneumoniae* than *E. coli*.

The kmer comparison of kmer of length 3 can be seen in Figure 4.13.

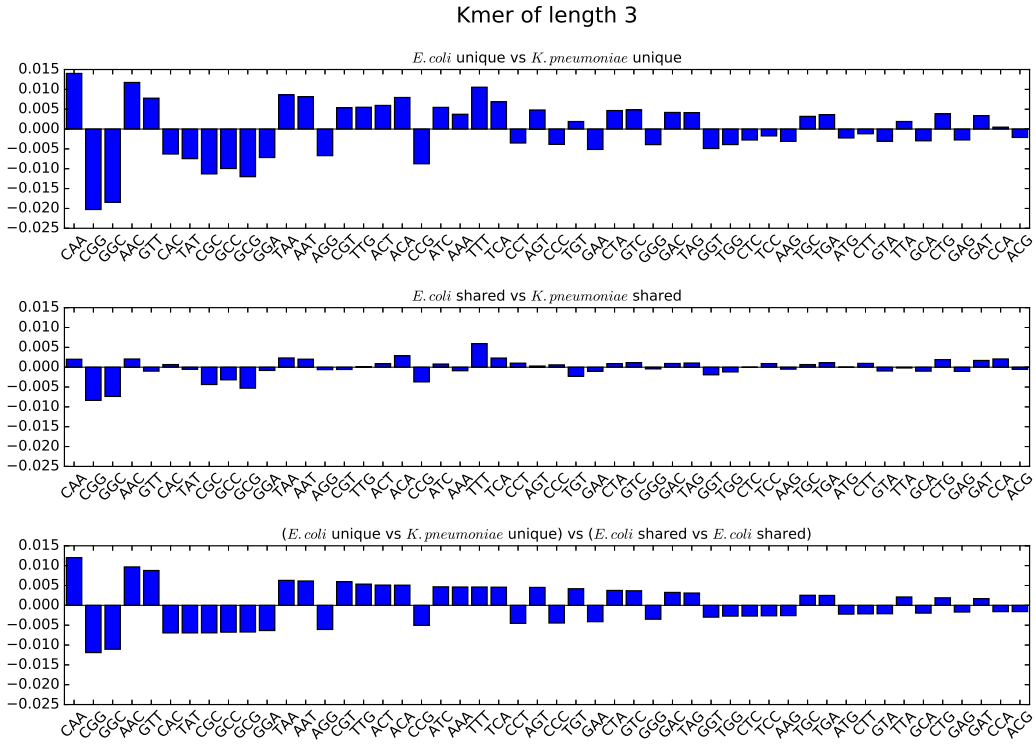


Figure 4.14: Comparison of kmers, kmer of length 3. The upper bars are the difference between the unique groups, the middle bars the difference of the shared groups and the lower bars the difference between the two previous differences. On the x-axis are the kmers and on the y-axis are the values from the difference of the kmer distributions, for the two upper bars a positive value means a higher kmer distribution value in *E .coli*. All values are sorted by the absolute value of the kmers in the third dataset.

Kmer of length 3, represents the codon level. The top kmers, CAA, CGG and GGC had all the same direction of the bar in all three plots, with higher absolute values for the unique genes. Many of the bars for the shared comparison were around zero, while the unique groups differed more.

The kmer comparison of kmer of length 6 can be seen in Figure 4.15.

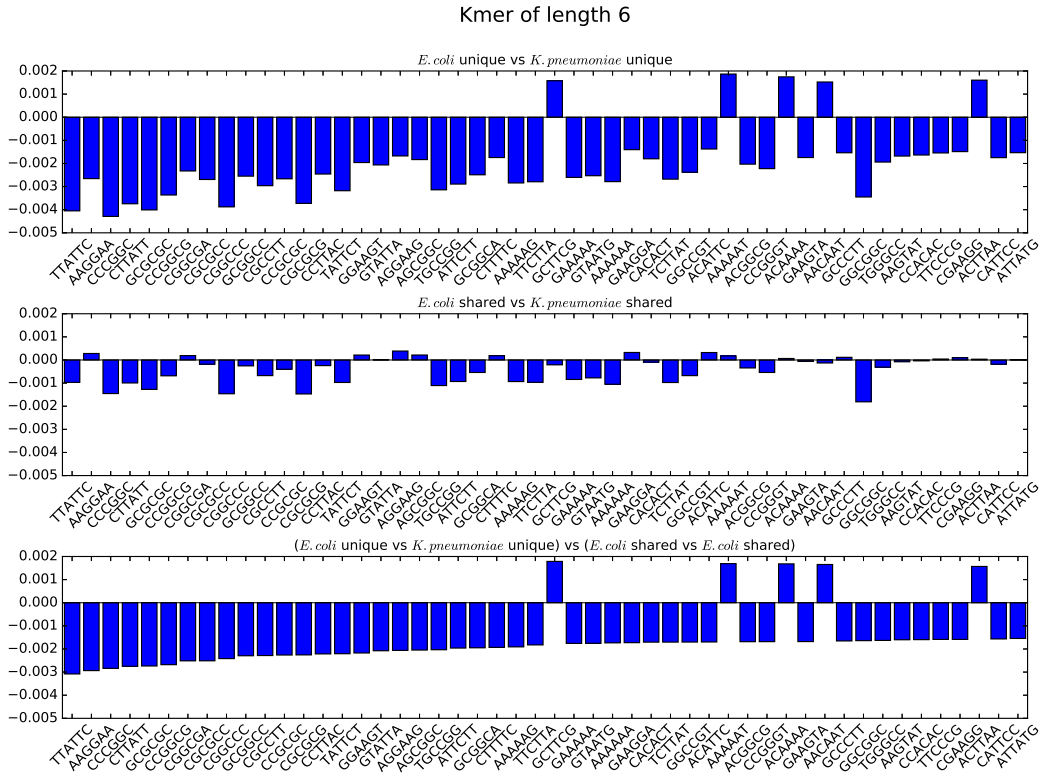


Figure 4.15: Comparison of kmers, kmer of length 6. The upper bars are the difference between the unique groups, the middle bars the difference of the shared groups and the lower bars the difference between the two previous differences. On the x-axis are the kmers and on the y-axis are the values from the difference of the kmer distributions, for the two upper bars a positive value means a higher kmer distribution value in *E .coli*. All values are sorted by the absolute value of the kmers in the third dataset.

The kmers of length 6 could be related to transcription factors, here the most of the bars were negative, with higher fractions in the *K. pneumoniae*, while the bars in the shared genes had small differences. The kmers with the highest (absolute) values were TTATTC, AAGGAA and CCCGGC.

4.4 The predictive model

To create the models the response variable (present/not present in *E. coli*) and the length were fixed in a loop while the kmer of length 3 with lowest p-value for each round was added into the model until a threshold was exceeded. To get the specificity and sensitivity values the data was split in half training and half testing for the different cutoff classification values and was run for 100 times. The generated specificity and sensitivity values were plotted with a ROC curve.

4.4.1 All genes

For all genes, out of 63 possible kmers, the final model included 21 kmers. Using the selected kmers on all data, the p-values and estimates for the model can be seen in Table 4.9. The p-values and estimates when each kmer were added can be seen in Appendix A.3.

Table 4.9: Model parameters, estimates and p-values for all genes, with the kmers in the order they were added to the model.

Parameter	Estimate	P-value
Intercept	-13,73	1,87e-20
Length	4,16e-04	6,87e-02
AAG	-111,59	7,94e-16
ACT	173,07	3,30e-26
CAG	85,72	7,73e-10
GAG	172,06	2,12e-35
AGG	-80,85	2,03e-06
ACC	-21,83	1,34e-01
GGA	40,48	1,10e-02
TGT	-104,98	6,44e-10
CCC	-24,03	8,82e-02
GCT	118,78	1,87e-16
CGG	-70,30	9,10e-08
CGC	69,34	1,34e-12
TAA	124,10	3,37e-22
TTC	112,37	1,63e-13
TTG	23,94	1,16e-01
GCA	79,81	9,41e-08
GTC	106,41	2,12e-10
TGA	64,38	2,30e-05
ATC	65,41	7,34e-06
CTC	-59,09	3,59e-04
GTT	59,08	5,18e-04

In the final model all kmers still had an impact of the result, with the kmers GAG, ACT and TAA with the largest influences.

4.4.2 Antibiotic classes

4.4.2.1 Beta-lactams

For beta-lactams, the final model included 16 kmers. The p-values and estimates for the model can be seen in Table 4.10.

Table 4.10: Model parameters, estimates and p-values for the antibiotic class beta-lactam, with the kmers in the order they were added to the model.

Parameter	Estimate	P-value
Intercept	-15,75	4,13e-11
Length	2,62e-03	4,65e-04
AAG	-274,88	5,92e-18
AGA	-105,37	4,10e-03
ACT	379,39	2,82e-22
GCA	218,36	1,37e-11
CCT	-462,29	8,16e-26
ACA	-318,25	8,44e-17
CAG	269,40	4,90e-13
GTA	197,92	4,90e-13
GAG	474,86	2,33e-22
GAA	366,80	8,13e-17
GTG	-161,86	3,58e-06
TCG	380,73	9,28e-17
CGA	-281,24	1,01e-12
TCT	-174,37	9,12e-08
TTA	152,56	5,25e-08
CAC	137,25	6,75e-04

In the model for beta-lactams, the kmers CCT, GAG and ACT were of importance.

4.4.2.2 Aminoglycosides

For aminoglycosides, the final model included three kmers, CCT, TAA and TTG. The p-values and estimates for the model can be seen in Table 4.11.

Table 4.11: Model parameters, estimates and p-values for the antibiotic class aminoglycosides, with the kmers in the order they were added to the model

Parameter	Estimate	P-value
Intercept	-6,50	6,77e-09
Length	4,29e-03	1,57e-04
CCT	157,32	8,49e-04
TAA	-95,19	8,94e-05
TTG	89,64	5,89e-03

4.4.2.3 Macrolides

For macrolides, the final model included three kmers, GGT, CAG and TTC. The p-values and estimate for the model can be seen in Table 4.12.

Table 4.12: Model parameters for the antibiotic class macrolides, with the kmers in the order they were added to the model

Parameters	Estimate	P-value
Intercept	6,98e-01	6,37e-01
Length	-3,94e-03	1,12e-03
GGT	-316,57	6,41e-06
CAG	233,21	1,43e-04
TTC	167,95	1,12e-03

4.4.3 ROC curves

The sensitivity and specificity values were obtained from the four models all genes and three antibiotic classes at different cutoff values for the classification, generating ROC-curves seen in Figure 4.16.

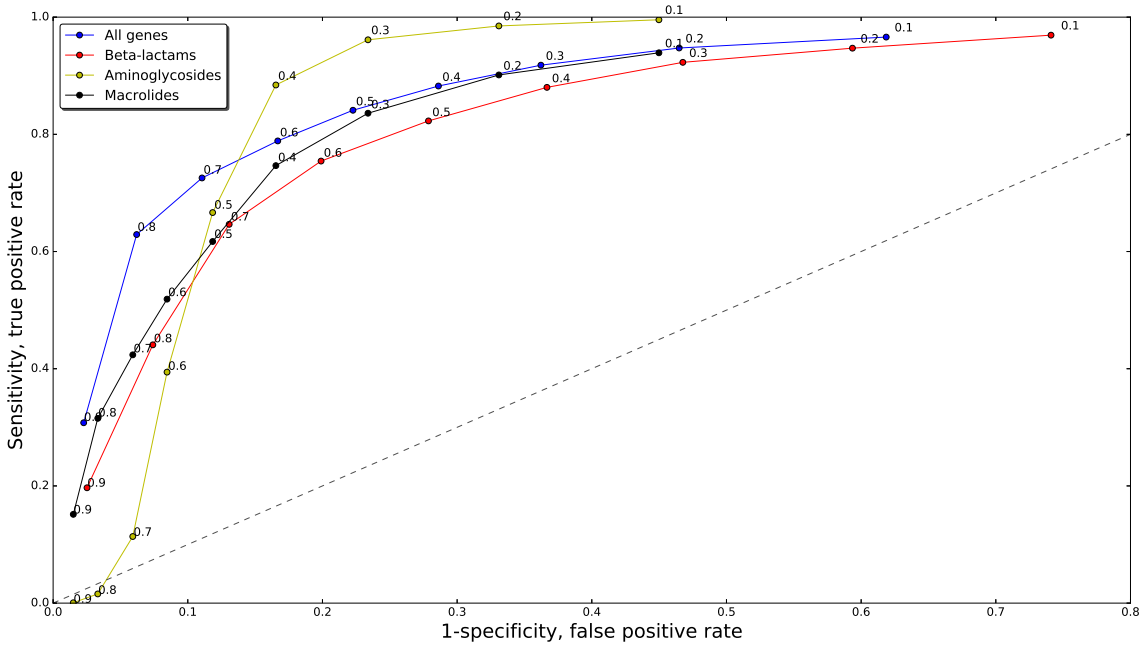


Figure 4.16: ROC curves for the four models, all genes and three antibiotic classes. The numbers next to the dots represents the cutoff value for the classification and the dashed line (y=x) represents a random classification. On the x-axis is the false positive rate or (1-specificity) and on the y-axis the true positive rate or the sensitivity.

5

Discussion

In the results for the kmer analysis, tables of median and p-values were presented for kmer of lengths 1 to 8 together with plots of all values for some kmer lengths. The results from the three methods squared Euclidean distance, maximum absolute value and maximum value showed differences for the shorter kmers for all methods. In the frequency analysis the correlation between number of hits was plotted against the values from the kmer analysis in scatter plots for some kmer lengths, where the genes with the highest values had fewer hits. For the analysis of regions surrounding the gene the upstream and downstream sides of the genes were compared without seeing any differences of the two sides. Also, comparisons between unique and shared gene on both group and kmer level was done and differences could be seen. Finally, logistic regression predictive models were created to predict gene compatibility for all genes and three antibiotic classes. The most important kmers of the models were shown as well as ROC-curves for the models, where the most classifiers performed better than a random classification.

5.1 Kmer analysis

In the kmer analysis three different methods were used to compare core genome genes and antibiotic resistant genes present respectively not present in *E. coli* and *K. pneumoniae*. In the method squared Euclidean distance all kmers of the gene had impact on the value, to describe the distance from the kmers to the average core genome gene and similar patterns could be seen in both *E. coli* and *K. pneumoniae*. The distribution of the genes present in the species had, for the shorter kmer lengths, a distribution similar to the core genome genes, while the genes not present in species were more spread. Also, the p-values of the comparisons between the groups were low. The differences decreased around kmer of length 5 for *E. coli* and kmer of length 4 for *K. pneumoniae*, so the method was interesting for the shorter kmer lengths.

In the method maximum absolute value, the largest absolute difference of kmer distribution values in both core genome and genes present respectively not present in the species were calculated, only involving maximum two kmers in each value. The patterns were similar to the ones from squared Euclidean distance for both species. The values were not that spread but there were differences for the shorter kmer lengths between the groups and also for the median values for the shorter kmer lengths up to kmer of length 5 for *E. coli* and kmer of length 4 for *K. pneumoniae*.

The method maximum value compared only the value of the most frequent kmers from the average core genome gene and the antibiotic resistant genes. From the average core genome gene, it was the same value every time, while the value from the antibiotic resistant gene varied. The plots had fewer values with higher frequency but there could still be seen differences similar to the other methods, though the differences were not that large. For *K. pneumoniae* the values were not following exactly the same patterns as for the previous methods but the median values stabilized for the longer kmers.

5.2 Frequency analysis

The frequency analysis was done to see if there were any correlations between the frequency of the antibiotic resistant genes and the values from the kmer analysis. The values showed differences of the genes to the average core genome genes kmer distribution and with this analysis it was interesting to look if the number of hits could relate to be more likely to be compatible in the species.

From the plots there could be seen that *K. pneumoniae* had more hits for the gene groups with highest number of hits, even though there were fewer genomes in the database. The number of hits did not tell how many of the strains the gene was present in as hits at the same positions in multiple species only were counted as one hit.

There could not be seen any specific differences between the three used methods, squared Euclidean distance, maximum absolute value and maximum value for this analysis. Most gene group values clustered around the median values and had various number of hits, there the genes with most hits also could be found. The values far away from the median value did not have as many hits, which also could be seen by the decreasing lowess trend line for higher values. The more hits the more probable the value was around the median value, but if the value was around the median value did not mean there was many hits. From these results there could be concluded that the more the antibiotic resistant genes differed compared to the core genome genes, the more rare they were in the species.

To give the antibiotic classes different colors was a reason to see if a specific class, where the genes should have similar mechanisms, had trends between or within the class. No clear patterns could be seen, for all kmer lengths the number of hits varied within the classes, while for higher kmers the values where more similar and could be seen to group in the plots. This showed on a more distinct distribution of longer kmers within the classes to distinguish them from each other. The only antibiotic class had similar number of hits for all genes was vancomycin in *K. pneumoniae*, which was an interesting result as the vancomycin affects gram-negative bacteria and is ineffective against gram-negative bacteria as *K. pneumoniae* [16].

5.3 Analysis of regions surrounding the gene

The first step was to see if there were any differences of the upstream and downstream regions of the genes. There could not be seen a statistically significant difference between the two sides and the rest of the analyses did not separate the groups. If there would have been a difference, it could have been necessary to separate the two sides in the analyses, for example are the promoters at the upstream side.

Comparing the groups and following the hypothesis that if there would not exist any difference between the species and genes, within the species the values would have been close to zero and between the species the values would have been equal for the shared and unique comparisons. When looking at the values, within the same species the values were low, maybe little higher than expected. The values for the unique groups were higher, suggesting a difference of the surrounding of the genes. The more surprisingly results were the values for the shared genes, that were lower than the values for the genes within the same species. This low difference could been explained by taking the same genes for both groups, but the analysis was done multiple times, taking different random genes in each group with various sizes of the groups, in all runs giving low values.

Next analysis was to look at the kmers of different lengths that contributed to the difference between the groups. The GC content was represented by the kmer of length 1, with more G and C in *K. pneumoniae* than *E. coli*, this result was following the full genome GC content percentages in the species *K. pneumoniae* (57%) and *E. coli* (51%) [17].

On the codon level, kmer of length 3, the top three codons were explored more. CAA, was more common in the unique genes of *E. coli* and CGG and GGC were more common in the unique genes of *K. pneumoniae*. No clear differences of the frequency of these codons in the full genomes could be seen, possibly CGG could be related to the higher frequency in *K. pneumoniae* of 10,6 (per thousand codon) compared to 5,0 (per thousand codons) in *E. coli* [17].

For kmer of length 6, the kmers could be transcription factors, here the most of the kmers were more common in the unique genes of *K. pneumoniae*. No specific transcription factors of interest could be found, though it was noted that many of the kmers included a lot of C and G, which could be connected to the higher GC content in *K. pneumoniae*. It could have been interesting to look at longer kmer lengths to further investigate if specific transcription factors could be found.

5.4 The predictive model

Predictive models were created for all genes and three antibiotic classes to see the predictive ability of the classification, how well the models could perform. When looking at the ROC curves, the models performed well, for most point better than

a random classifier.

For all genes 21 kmers were used in the model, when comparing the kmers to a codon frequency table, the estimates for many of the kmer followed the usage frequency in the full genome of *E. coli*. For example, AAG was the least used codon for K (lysine), CAG was the most used codon for Q (glutamine) and TAA was the most used stop codon. If a gene should be compatible in *E. coli*, it would be reasonable the gene would include the preferred codons of the species [17].

The models were also done for antibiotic classes, to see how well the model could perform on a smaller set of data with more similar genes. The model for beta-lactams included 16 kmers and except from the first added kmer AAG, the most kmers were different from the model with all genes.

The models for aminoglycosides and macrolides contained only three kmers each and they were able to perform good. The ROC-curve of the model for macrolides was similar to the model for beta-lactams and did not differ much from the model for all genes. For the cutoff values 0,9 and 0,8 the model for aminoglycosides performed worse than a random classification but was then able to have the highest true positive rate out of all models.

In the desired outcome for this type of classification, new genes would be identified to be transferred to and be resistant in *E. coli* after input of the data of the genes. The model should rather avoid false positives than false negatives and it would be more favourably to have a low false positive rate. Preferably the true positive rate would have been 100% and the false positive rate 0%, but that is often not the case.

To give an example, from the ROC curve in Figure 4.16, for the point 0,7 for all genes, the true positive rate was 0,72 and the false positive rate was 0,11. Of the 2115 genes the values would mean, out of 1117 genes present or positive in this case, 804 would be correctly classified while 313 would be false negatives. For the 998 not present or negative genes, 888 genes would be correctly classified while 110 genes would be classified as false positives.

To get an even better model more input parameters could have been used to test the predictive ability, now only the length and distribution values of kmer of length 3 were used. Also, other methods as random forest could have been used to create models.

6
Conclusion

Through different analyses some patterns of gene compatibility have been found. In the kmer analysis differences could be seen for both the species *E. coli* and *K. pneumoniae*. For the shorter kmer lengths the median values differed and was smaller between the core genome and antibiotic resistant present in the species than difference between the core genome and the genes not present in the species. Also, the p-values were small for the comparisons between the groups for the shorter kmer lengths and the differences decreased with increasing kmer length. The method squared Euclidean distance took all kmer into consideration and seemed to be the method with the most differences between the antibiotic genes. On the other hand, the method maximum value had the lowest differences and also for the *K. pneumoniae* the values were not following the rest of the trends. The kmer distributions can be used to see differences for the genes between the different groups, primarily for smaller kmer lengths.

In the frequency analysis many values clustered around the median values, with various number of hits. The gene groups with the most hits were close to the median values and the values far away from the median values, did not have many hits. No clear conclusion about differences between or within classes could be done, for all kmer lengths the number of hits varies within the classes, while for higher kmers the values were more similar and could be seen to group in the plots. No clear trends of hits could be seen within the classes. The analysis was able to see how high values, and the more the antibiotic resistant genes differed from the core genome, the more rare they were in the species.

For analysis of regions surrounding the gene the genes were divided into groups if they were present in both or only one specie and when comparing the gene groups of unique and shared genes. There could be seen differences between the unique genes of the species, while the shared gene values were surprisingly low. On kmer level, the kmers that differed most between the species had no clear correlation, potentially they could be related to the higher GC content in *K. pneumoniae*. The analysis found differences between the unique and shared groups and were able to tell which kmers had the biggest influence on that differences.

Lastly, looking at the four created predictive models, the model for all genes included with the length 21 kmers out of 63 possible. The model was able to find some of the least and most frequent used codons in *E. coli*. The models for the antibiotics classes did perform similar than the model for all genes, but for aminoglycosides the low

cutoff values gave high true positive rates. In this kind of classification, a low false positive rate would be of interest and with the model for all genes the best values of true positive rate of 72% and false e positive rate of 11%. From the gene sequence it was possible to predict gene compatibility from a logistic regression predictive model, both for all genes and when dividing the genes into antibiotic classes.

To summarize, in this project analyses have developed and from the results it has been found compatibility patterns of the antibiotic resistant genes by looking at the gene sequences and the regions surrounding the genes. From the gene sequences it has also been possible to predict the gene compatibility in a predictive model.

Bibliography

1. WHO. *Antibiotic resistance* https://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance (2020). [Accessed: 2020-10-08].
2. Acquired antibiotic resistance genes: An overview. *Frontiers in Microbiology* **2,** 1–27 (2011).
3. C Reygaert, W. An overview of the antimicrobial resistance mechanisms of bacteria. *AIMS Microbiology* **4,** 482–501 (2018).
4. Idalia, V.-M. N. & Bernardo, F. in *Escherichia coli - Recent Advances on Physiology, Pathogenesis and Biotechnological Applications* 253–274 (Intech, 2017).
5. Kamran Taj, M. & Samreen, Z. Escherichia coli as a model organism. *Internation Journal of Engineering Research and Science & Technology* **3,** 80–88 (2014).
6. Serres, M. H., Gopal, S., Nahum, L. A., Liang, P. & Riley, M. A functional update of the, 1–7 (2001).
7. Li, B., Zhao, Y., Liu, C., Chen, Z. & Zhou, D. Molecular pathogenesis of Klebsiella pneumoniae. *Future Microbiology* **9,** 1071–1081 (2014).
8. Liu, P. *et al.* Complete genome sequence of Klebsiella pneumoniae subsp. pneumoniae HS11286, a multidrug-resistant strain isolated from human sputum. *Journal of Bacteriology* **194,** 1841–1842 (2012).
9. Sheskin, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures* 5th edition (CRC Press, 2011).
10. Norton, E. C. & Dowd, B. E. Log Odds and the Interpretation of Logit Models. *Health Services Research* **53,** 859–878 (2018).
11. Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters* **27,** 861–874 (2006).
12. Zankari, E. *et al.* Identification of acquired antimicrobial resistance genes. *Journal of Antimicrobial Chemotherapy* **67,** 2640–2644 (2012).
13. Bortolaia, V. *et al.* ResFinder 4.0 for predictions of phenotypes from genotypes. *The Journal of antimicrobial chemotherapy* **75,** 3491–3500 (2020).
14. Jordan, I. K., Rogozin, I. B., Wolf, Y. I. & Koonin, E. V. Essential Genes Are More Evolutionarily Conserved Than Are Nonessential Genes in Bacteria. *Genome Research* **12,** 962–968 (2002).
15. Cleveland, W. S. LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression. *The American Statistician* **35,** 54–54 (1981).
16. Miller, S. I. Antibiotic resistance and regulation of the Gram-negative bacterial outer membrane barrier by host innate immune molecules. *mBio* **7,** 5–7 (2016).

17. Nakamura, Y., Gojobori, T. & Ikemura, T. Codon usage tabulated from international DNA sequence databases: Status for the year 2000. *Nucleic Acids Research* **28,** 292 (2000).

A

Appendix

A.1 Example scripts

A.1.1 Python code

Python version 2.7.11.

A.1.1.1 Kmer analysis

```
# Kmer analyis for E. coli
import matplotlib.pyplot as plt
from numpy import median
import extract_blast_results
import function_scripts, see section below
from scipy.stats import ranksums

ecoli="core_genomes/ecoli/ecoli_core_genome.fsa" # fasta file of
    the core genome
anti="resfinder/all_from_resfinder.fsa" # fasta file with all
   the antibiotic resistant genes
blast="blast_ecoli.txt" # the lines from the BLAST results

for k in range(1,9):
        print("kmer= "+str(k))
        blastresult=extract_blast_results.blast_results(blast) #
            extract only the genes fulfilling the hits
           conditions 95% identity, 75% length
        # the core genome and antibiotic resistant genes into
           nested dictionaries, gene name and kmers as keys and
           kmer distribution values as values
        n_ecoli=function_scripts.ecoli_to_nested(ecoli,k)
        n_anti=function_scripts.resfinder_to_nested(anti,k)

        medel_ecoli=function_scripts.medel_kmer(n_ecoli) # the
           average core genome kmer distribution

        dist_anti=function_scripts.distance(medel_ecoli, n_anti)
            # squared Euclidean distance between average core
           genome gene and antibiotic resistant genes
```

```
dist_ecoli=function_scripts.distance(medel_ecoli,
    n_ecoli) # squared Euclidean distance between average
     core genome gene and core genome genes

max_abs_anti=function_scripts.max_abs_value(medel_ecoli,
     n_anti)
max_abs_ecoli=function_scripts.max_abs_value(medel_ecoli
    , n_ecoli)

max_anti=function_scripts.max_value(medel_ecoli, n_anti)
max_ecoli=function_scripts.max_value(medel_ecoli,
    n_ecoli)

items=[[dist_anti, dist_ecoli], [max_abs_anti,
    max_abs_ecoli], [max_anti, max_ecoli]]
names=["Squared␣Euclidean␣distance", "Maximum␣absolute␣
    value", "Maximum␣value"]
filenames=["distance", "max_absolute_value", "max_value"
    ]
for i in range(len(items)):
        present=[]
        not_present=[]
        for key, value in items[i][0].items():
                if key in blastresult:
                        present.append(value)
                else:
                        not_present.append(value)
        # Wilcoxon rank-sum test between the groups
        print(ranksums(present, not_present))
        print(ranksums(present, items[i][1].values()))
        print(ranksums(not_present, items[i][1].values()
            ))

        x1=[float(j) for j in present]
        x2=[float(j) for j in not_present]

        # Median values for the three groups
        print(median(items[i][1].values()))
        print(median(present))
        print(median(not_present))

        fig, (ax1, ax2, ax3)=plt.subplots(3)
        fig.suptitle(str(names[i])+",␣kmer␣of␣length␣"+
            str(k), fontsize=18)
        fig.tight_layout(pad=1.4)

        ax1.hist(items[i][1].values(), bins=1500)
        ax1.set_title("Genes␣in␣core␣genome␣$\it{E.␣coli
            }$")
```

```
ax2.hist(x1, bins=1500)
ax2.set_title("Antibiotic resistant genes
    present in $\it{E. coli}$")
ax3.hist(x2, bins=1500)
ax3.set_title("Antibiotic resistant genes not
    present in $\it{E. coli}$")

xmin, xmax1=ax1.get_xlim()
xmin, xmax2=ax2.get_xlim()
xmin, xmax3=ax3.get_xlim()
xmax=max([xmax1, xmax2, xmax3])

ax1.set_xlim([0, xmax])
ax2.set_xlim([0, xmax])
ax3.set_xlim([0, xmax])
plt.subplots_adjust(top=0.88)
plt.savefig("histogram_ecoli_"+filenames[i]+"_k"
    +str(k)+".eps", format='eps', dpi=1200)
plt.show()
plt.close()
```

A.1.1.2 Frequency analysis

```
# Frequency analysis for E. coli
import matplotlib.pyplot as plt
import function_scripts, see section below
import resfinder_classes
import math
import statsmodels.api as sm

lowess=sm.nonparametric.lowess

ecoli="core_genomes/ecoli/ecoli_core_genome.fsa"
anti="resfinder/all_from_resfinder.fsa"
blast="blast_ecoli.txt"

antibiotic_groups=resfinder_classes.antibiotic_classes() #
    script to extract dictionaries with the different
    antibiotic classes as keys and the genes belonging to
    each class as values

frequency={}
with open("hits_per_gene_group_ecoli.txt", 'r') as f: # file
     with the gene groups and the number of hits per gene
    group
        for line in f:
                lines=line.split('\t')
                frequency[lines[0]]=lines[1]
```

```
for k in range(1,9):
        n_ecoli=function_scripts.ecoli_to_nested(ecoli,k)
        n_anti=function_scripts.resfinder_to_nested(anti,k)

        medel_ecoli=function_scripts.medel_kmer(n_ecoli)

        dist_anti=function_scripts.distance(medel_ecoli,
           n_anti)
        average_gene_group_dist_anti=function_scripts.
           average_gene_group_value(dist_anti)

        max_abs_anti=function_scripts.max_abs_value(
           medel_ecoli, n_anti)
        average_gene_group_max_abs_anti=function_scripts.
           average_gene_group_value(max_abs_anti)

        max_anti=function_scripts.max_value(medel_ecoli,
           n_anti)
        average_gene_group_max_anti=function_scripts.
           average_gene_group_value(max_anti)

        items=[average_gene_group_dist_anti,
           average_gene_group_max_abs_anti,
           average_gene_group_max_anti]
        names=["Square␣Euclidean␣distance", "Maximum␣
           absolute␣value", "Maximum␣value"]
        filenames=["distance", "max_absolute_value", "
           max_value"]
        for i in range(len(items)):
        frequency_dict={}
        values_dict={}
        frequency_values=[]
        values_values=[]

        for key in antibiotic_classes:
            if key in ["nitroimidazole", "vancomycin","
               fusidicacid"]: # the classes not existing for
                E. coli
                continue
            frequency_dict[key]=[]
            values_dict[key]=[]
            for j in antibiotic_classes[key]:
                if j in frequency.keys():
                    frequency_dict[key].append((math.sqrt(
                        float(frequency[j].split('\n')[0]))))
```

```
                    values_dict[key].append(items[i][j])

                    frequency_values.append((math.sqrt(float
                        (frequency[j].split('\n')[0]))))
                    values_values.append(items[i][j])

        color_list=['black','grey', 'brown', 'red','orange',
            'yellow','lawngreen', 'green', 'turquoise','blue
           ','darkviolet', 'pink']
        counter=0
        anti_keys= ["rifampicin", "aminoglycosides", "
           colistin", "fosfomycin", "macrolides", "
           quinolones", "trimethoprim", "tetracyclines", "
           oxazolidinones", "sulphonamides", "beta-lactams",
            "phenicols"]
        for keys in values_dict:
            plt.scatter(values_dict[keys], frequency_dict[
               keys], color=(color_list[counter]),label=
               anti_keys[counter])
            counter+=1

        # Lowess line created
        z=lowess(frequency_values,values_values, frac=0.66,
           it=0)
        d=list(list(zip(*z))[0])
        c=list(list(zip(*z))[1])
        plt.plot(d,c)

        plt.legend(loc='best', shadow=True, fancybox=True,
           scatterpoints=1, fontsize=11)
        plt.xlim(left=0)
        plt.ylim(bottom=0)
        plt.suptitle(str(names[i])+",␣kmer␣of␣length␣"+str(k
           ), fontsize=18)
        plt.title("$\it{E.␣coli}$", fontsize=13)
        plt.ylabel('Square␣root␣of␣the␣number␣of␣hits',
           fontsize=12)
        plt.savefig("scatter_ecoli_"+filenames[i]+"_k"+str(k
           )+".eps", format='eps', dpi=1200)
        plt.show()
        plt.close()
```

A.1.1.3 Analysis of the regions surrounding the gene

```
import cPickle
import function_scripts, see section below
import random
```

```
import matplotlib.pyplot as plt
from scipy import stats

ecoli_blast_file="fulfill_lines_in_ecoli.txt" # textfiles
    with the lines of hits from BLAST
klebsiella_blast_file="fulfill_lines_in_klebsiella.txt"

ecoli_file=open("pickle_dict_ecoli_genomes") # storage of
    all E. coli genomes from the database
ecoli_dict=cPickle.load(ecoli_file)
ecoli_file.close()

kleb_file=open("pickle_dict_klebsiella")
klebsiella_dict=cPickle.load(kleb_file)
kleb_file.close()

# Add the genes of the species into lists
ecoli_genes=[]
klebsiella_genes=[]

with open (ecoli_blast_file, 'r') as f1, open (
    klebsiella_blast_file, 'r') as f2 :
        for line in f1:
                gene=line.split("\t")[0]
                if gene not in ecoli_genes:
                        ecoli_genes.append(gene)
        for line in f2:
                gene=line.split("\t")[0]
                if gene not in klebsiella_genes:
                        klebsiella_genes.append(gene)

# Split the genes in groups of shared and unique
unique_ecoli=[]
unique_klebsiella=[]
all_shared_genes=[]

for gene in ecoli_genes:
    if gene not in klebsiella_genes:
        unique_ecoli.append(gene)
    else:
        all_shared_genes.append(gene)

for gene in klebsiella_genes:
    if gene not in ecoli_genes:
        unique_klebsiella.append(gene)
```

```
## Pick some of the shared genes and remove the genes in
   common of the two groups
random.seed(3)
shared_genes1=random.sample(all_shared_genes,105)
random.seed(23)
shared_genes2=random.sample(all_shared_genes,155)

for i in shared_genes1:
    if i in shared_genes2:
        shared_genes1.remove(i)
        shared_genes2.remove(i)

upstreams_shared=[]
downstreams_shared=[]

# Get the upstream and downstream sequences of E. coli for
   the unique genes and also the sequences for the shared
   genes

upstreams_ecoli=[]
downstreams_ecoli=[]
shared_ecoli=[]

with open(ecoli_blast_file, 'r') as f:
        for line in f:
                gene=line.split("\t")[0]
                genome=line.split("\t")[1]
                start=line.split("\t")[8]
                end=line.split("\t")[9]

                if gene in unique_ecoli: #1
                        surrounding=function_scripts.
                           gene_surrounding([start, end],
                           ecoli_dict[genome], 100)
                        upstreams_ecoli.append(surrounding
                           [0])
                        downstreams_ecoli.append(surrounding
                           [1])

                elif gene in shared_genes1:

                        surrounding=function_scripts.
                           gene_surrounding([start, end],
                           ecoli_dict[genome], 100)
                        upstreams_shared.append(surrounding
                           [0])
```

```
                        downstreams_shared.append(
                            surrounding[1])
                        shared_ecoli.extend([surrounding[0],
                            surrounding[1]])

# same for K. pneumoniae
upstreams_klebsiella=[]
downstreams_klebsiella=[]
shared_klebsiella=[]

with open(klebsiella_blast_file, 'r') as f:
        for line in f:
                gene=line.split("\t")[0]
                genome=line.split("\t")[1]
                start=line.split("\t")[8]
                end=line.split("\t")[9]

                if gene in unique_klebsiella:
                        surrounding=function_scripts.
                            gene_surrounding([start, end],
                            klebsiella_dict[genome], 100)
                        upstreams_klebsiella.append(
                            surrounding[0])
                        downstreams_klebsiella.append(
                            surrounding[1])

                elif gene in shared_genes2:
                         surrounding=function_scripts.
                             gene_surrounding([start, end],
                             klebsiella_dict[genome], 100)
                         upstreams_shared.append(surrounding
                             [0])
                         downstreams_shared.append(
                             surrounding[1])
                         shared_klebsiella.extend([
                             surrounding[0],surrounding[1]])

for i in range(1,8):
        kmer_length=int(i)

        kmers_up_ecoli=function_scripts.list_to_kmers(
            upstreams_ecoli, kmer_length)
        kmers_down_ecoli=function_scripts.list_to_kmers(
            downstreams_ecoli, kmer_length)
        kmers_up_klebsiella=function_scripts.list_to_kmers(
            upstreams_klebsiella, kmer_length)
```

```
kmers_down_klebsiella=function_scripts.list_to_kmers
    (downstreams_klebsiella, kmer_length)
kmers_up_shared=function_scripts.list_to_kmers(
    upstreams_shared, kmer_length)
kmers_down_shared=function_scripts.list_to_kmers(
    downstreams_shared, kmer_length)


# P-values of the upstream vs downstream sides, need
    to match in the order of kmers in lsits
all_kmers=[''.join(x) for x in itertools.product('
    ACGT', repeat=kmer_length)]
x=list(range(len(all_kmers)))

upvalues_ecoli=[]
downvalues_ecoli=[]

upvalues_klebsiella=[]
downvalues_klebsiella=[]

upvalues_shared=[]
downvalues_shared=[]

for kmer in all_kmers:
    if kmer in kmers_up_ecoli.keys():
        upvalues_ecoli.append(kmer_up_ecoli[kmer])
    else:
        upvalues_ecoli.append(int(0))

    if kmer in kmers_down_ecoli.keys():
        downvalues_ecoli.append(kmers_down_ecoli[
            kmer])
    else:
        downvalues_ecoli.append(int(0))
    if kmer in kmers_up_klebsiella.keys():
        upvalues_klebsiella.append(
            kmers_up_klebsiella[kmer])
    else:
        upvalues_klebsiella.append(int(0))

    if kmer in kmers_down_klebsiella.keys():
        downvalues_klebsiella.append(
            kmers_down_klebsiella[kmer])
    else:
        downvalues_klebsiella.append(int(0))
```

```
        if kmer in kmers_up_shared.keys():
            upvalues_shared.append(kmers_up_shared[kmer])
        else:
            upvalues_shared.append(int(0))

        if kmer in kmers_down_shared.keys():
            downvalues_shared.append(kmers_down_shared[
                kmer])
        else:
            downvalues_shared.append(int(0))

print("E. coli upstream vs downstream")
print(stats.wilcoxon(upvalues_ecoli,
    downvalues_ecoli))

print("K. pneumoniae upstream vs downstream")
print(stats.wilcoxon(upvalues_klebsiella,
    downvalues_klebsiella))

print("Shared genes upstream vs downstream")
print(stats.wilcoxon(upvalues_shared,
    downvalues_shared))


# Comparisons of gene groups
all_kmers_unique_ecoli=function_scripts.
    list_to_kmers(upstreams_ecoli+downstreams_ecoli,
    kmer_length)
all_kmers_unique_klebsiella=function_scripts.
    list_to_kmers(upstreams_klebsiella+
    downstreams_klebsiella, kmer_length)
all_kmers_shared_ecoli=function_scripts.
    list_to_kmers(shared_ecoli, kmer_length)
all_kmers_shared_klebsiella=function_scripts.
    list_to_kmers(shared_klebsiella, kmer_length)

# Values
print("E. coli unique vs K. pneumoniae unique")
print(function_scripts.distance_two_dicts(
    all_kmers_unique_ecoli,
    all_kmers_unique_klebsiella))

print("E. coli shared vs K. pneumoniae shared")
print(function_scripts.distance_two_dicts(
    all_kmers_shared_ecoli,
    all_kmers_shared_klebsiella ))
```

```
list_unique=function_scripts.
    abs_difference_two_dicts(all_kmers_unique_ecoli,
    all_kmers_unique_klebsiella)
list_shared=function_scripts.
    abs_difference_two_dicts(all_kmers_shared_ecoli,
    all_kmers_shared_klebsiella)

# Difference of unique and shared groups
list_diff=function_scripts.abs_difference_two_dicts(
    list_unique[3], list_shared[3])
nr_of_hits=[4, 16, 50, 50, 50, 50, 50] # number of
    kmers will be printed in bar plot

# Sort the values by the third dataset
unique_values=[]
unique_kmers=[]
shared_values=[]
shared_kmers=[]
for kmer in list_diff[1][:nr_of_hits[i-1]]:
    if kmer not in list_unique[3]:
        unique_values.append(float(0))
    else:
        unique_values.append(list_unique[3][kmer])
    if kmer not in list_shared[3]:
        shared_values.append(float(0))
    else:
        shared_values.append(list_shared[3][kmer])
        unique_kmers.append(kmer)
        shared_kmers.append(kmer)

fig, (ax1, ax2, ax3)=plt.subplots(3)
fig.suptitle("Kmer␣of␣length␣" + str(kmer_length),
    fontsize=18)

ax1.bar(range(nr_of_hits[i-1]), unique_values, align
    ='center', tick_label=unique_kmers)
ax1.set_title("$\it{E.␣coli}$␣unique␣vs␣$\it{K.␣
    pneumoniae}$␣unique",fontsize=12)
ax2.bar(range(nr_of_hits[i-1]), shared_values, align
    ='center', tick_label=shared_kmers)
ax2.set_title("$\it{E.␣coli}$␣shared␣vs␣$\it{K.␣
    pneumoniae}$␣shared", fontsize=12)
ax3.bar(range(nr_of_hits[i-1]), list_diff[2][:
    nr_of_hits[i-1]], align='center', tick_label=
    list_diff[1][:nr_of_hits[i-1]])
```

```
ax3.set_title("$\it{E. coli}$ unique vs $\it{K.
    pneumoniae}$ unique) vs ($\it{E. coli}$ shared vs
    $\it{E. coli}$ shared)", fontsize=12)

for ax in [ax1, ax2 ,ax3]:
    ax.axhline(y=0.0, c="black",linewidth=0.5)
if int(kmer_length) >=3:
    plt.setp(ax1.get_xticklabels(), rotation=45)
    plt.setp(ax2.get_xticklabels(), rotation=45)
    plt.setp(ax3.get_xticklabels(), rotation=45)
ymin1, ymax1=ax1.get_ylim()
ymin2, ymax2=ax2.get_ylim()
ymin3, ymax3=ax3.get_ylim()
ymin=min([ymin1, ymin2, ymin3])
ymax=max([ymax1, ymax2, ymax3])

plt.subplots_adjust(left=0.075, right=0.925, bottom
    =0.1, top=0.9, wspace=0, hspace=0.5)
plt.setp((ax1,ax2, ax3),xlim=[-0.5, nr_of_hits[i
    -1]-0.5], ylim=[ymin, ymax])
plt.savefig("gene_surrounding_k"+str(i)+".eps",
    format='eps', dpi=1200)
plt.show()
plt.close()
```

A.1.1.4 Function scripts

```
# Function scripts
import copy

def resfinder_to_nested(file,k):  # resfinder file into a
    nested dictionary with kmers and values
    nested={}
    counter=0
    seq_list=[]
    with open(file, 'r') as f:
        for line in f:
            line=line.strip()
            if line[0]==">":
                if not seq_list: # first time
                    current_dict=line[1:]
                    nested[current_dict]={}
                    continue
                else:
                    for seq in seq_list:
                        for i in range(0, len(seq)-k+1):
                            kmer=seq[i:i+k]
```

```
                                if "N" in kmer:
                                    continue
                                if kmer in nested[current_dict]:
                                    nested[current_dict][kmer
                                        ]+=1
                                else:
                                    nested[current_dict][kmer]=1
                            nested[current_dict] = {k: v / float(
                                total) for total in (sum(nested[
                                current_dict].values()),) for k, v in
                                 nested[current_dict].items()}

                            current_dict=line[1:]
                            nested[current_dict]={}
                            seq_list=[]
                            continue
                        seq_list.append(line)
                        seq_list = [''.join(seq_list)]

                # the last one
                for seq in seq_list:
                    for i in range(0, len(seq)-k+1):
                        kmer=seq[i:i+k]
                        if kmer in nested[current_dict]:
                            nested[current_dict][kmer]+=1
                        else:
                            nested[current_dict][kmer]=1
                    nested[current_dict] = {k: v / float(total) for
                        total in (sum(nested[current_dict].values())
                        ,) for k, v in nested[current_dict].items()}

            for key in nested.keys():
                if "N" in key:
                    nested[key]
            return nested

def max_value(dict1, nested2): # find the max value for
    each gene and the max value for the E. coli, the
    difference and then create a new dictionary with the (abs
    ) distance.
                max_dict = 0
                max_nested={}
                max_value_dict={}

                for k, v in nested2.items():
                        # print(k,v)
```

```
                max_nested[k] = max(v.values())

        max_dict=max(dict1.values())
        for key, value in max_nested.items():
                max_value_dict[key]=1.0*abs(max_dict-value)


        return max_value_dict

def medel_kmer(nested): # get the average kmer distribution
        dict={}
        sum_kmer = 0

        # iterating key value pair
        for key, value in nested.items():
                sum_kmer+=1
                for kmer in value:
                        if kmer in dict:
                                dict[kmer]+=value[kmer]
                        else:
                                dict[kmer]=value[kmer]

        dict = {k: v /float(sum_kmer) for k,v in dict.
           iteritems()}

        return dict

def distance(dict1, nested2): # compare the distance for the
    same kmer for all genes
        distance_dict=copy.deepcopy(nested2)

        for key, value in distance_dict.items():
                for kmer in value:
                        if kmer in dict1:
                                distance_dict[key][kmer
                                   ]=1.0*abs(value[kmer]-
                                   dict1[kmer])**2
                        else:
                                distance_dict[key][kmer
                                   ]=1.0*abs(value[kmer])**2

                for key2, value2 in dict1.items():
                        if key2 not in value:
                                distance_dict[key][key2
                                   ]=1.0*abs(value2)**2
```

```
        distance_dict = {k: sum(v.values()) for k, v in
            distance_dict.items()}

        return distance_dict


def max_abs_value(dict1, nested2): # find the maxiumum abs
    value between kmers for each gene
        abs_dict=copy.deepcopy(nested2)

        for key, value in abs_dict.items():
                for kmer in value:
                        if kmer in dict1:
                                abs_dict[key][kmer]=1.0*abs(
                                    value[kmer]-dict1[kmer])
                        else:
                                abs_dict[key][kmer]=1.0*abs(
                                    value[kmer])

                for key2, value2 in dict1.items():
                        if key2 not in value:
                                abs_dict[key][key2]=1.0*abs(
                                    value2)

        max_abs_dict={}
        for k, v in abs_dict.items():
                max_abs_dict[k]=max(v.values())

        return max_abs_dict

def ecoli_to_nested(file,k): # E. coli file into a nested
    dictionary with kmers and values
     nested={}
     seq_list=[]
     with open(file, 'r') as f:
         for line in f:
             line=line.strip()
             if line[0]==">":
                 if not seq_list: # first time
                     current_dict=line[1:]
                     nested[current_dict]={}
                     continue
                 else:
                     for seq in seq_list:
                         for i in range(0, len(seq)-k+1):
                             kmer=seq[i:i+k]
```

```
                                if "N" in kmer:
                                    continue
                                if kmer in nested[current_dict]:
                                    nested[current_dict][kmer
                                        ]+=1
                                else:
                                    nested[current_dict][kmer]=1
                            nested[current_dict] = {k: v / float(
                                total) for total in (sum(nested[
                                current_dict].values()),) for k, v in
                                 nested[current_dict].items()}

                            current_dict=line[1:]
                            nested[current_dict]={}
                            seq_list=[]
                            continue
                        seq_list.append(line)
                        seq_list = [''.join(seq_list)]
                    #the last one
                    for seq in seq_list:
                        for i in range(0, len(seq)-k+1):
                            kmer=seq[i:i+k]
                            if kmer in nested[current_dict]:
                                nested[current_dict][kmer]+=1
                            else:
                                nested[current_dict][kmer]=1
                        nested[current_dict] = {k: v / float(total) for
                            total in (sum(nested[current_dict].values())
                            ,) for k, v in nested[current_dict].items()}

    # if empty keys, delete
    empty_keys = [k for k,v in nested.iteritems() if not v]
    for k in empty_keys:
        del nested[k]

    return nested

def average_gene_group_value(dict): # input a dictionary
    with gene names and values, output a new dict with the
    average value for each of the gene group
     new_dict={}
     for key in dict:
         gene=key.split('_')
         gene_group=gene[0]
         if gene_group in new_dict:
             new_dict[gene_group].append(dict[key])
```

```
        else:
            new_dict[gene_group]=[dict[key]]

    average_dict={}
    for key in new_dict:
        average_dict[key]=sum(new_dict[key])/len(new_dict[
            key])
    return average_dict


def list_to_kmers(list, kmer_length): # have a list of
    sequences and make them into kmer distributions of kmer
    of length k
    kmer_dict={}
    k=int(kmer_length)
    sum_kmer=0
    for seq in list:
        for i in range(0, len(seq)-k+1):
            kmer=seq[i:i+k]
            lst=["K", "M", "N", "Y", "R"]
            f = any(f in kmer for f in lst)
            if f:
                continue
            sum_kmer+=1

            if kmer in kmer_dict.keys():
                kmer_dict[kmer]+=1
            else:
                kmer_dict[kmer]=1
    kmer_dict={k:v/float(sum_kmer) for k,v in kmer_dict.
        iteritems()}
    return kmer_dict

def distance_two_dicts(dict1, dict2): # compare the distance
    for the same kmer for all genes to get a value
    distance_dict={}
    for kmer, value in dict1.items():
        if kmer in dict2:
            distance_dict[kmer]=1.0*abs(dict1[kmer]-dict2[
                kmer])**2
        else:
            distance_dict[kmer]=1.0*abs(dict1[kmer])**2

    for kmer2, value2 in dict2.items():
        if kmer2 not in distance_dict.keys():
            distance_dict[kmer2]=1.0*abs(value2)**2
```

```
    distance_value = sum(distance_dict.values())
    return distance_value

def abs_difference_two_dicts(dict1, dict2): # compare the
   distance for the same kmer for all genes to get a value
   and sort the values
    new_dict={}
    for kmer, value in dict1.items():
        if kmer in dict2:
            new_dict[kmer]=dict1[kmer]-dict2[kmer]
        else:
            new_dict[kmer]=value

    for kmer2, value2 in dict2.items():
        if kmer2 not in new_dict.keys() :
            new_dict[kmer2]=0-value2

    dictlist=[]
    for key, value in new_dict.iteritems():
        dictlist.append([key, value])

    abs_sorted_list=sorted(dictlist, key=lambda sublist: abs
       (sublist[1]), reverse=True)

    key_abs=[]
    value_abs=[]
    for i in abs_sorted_list:
        key_abs.append(i[0])
        value_abs.append(i[1])

        return abs_sorted_list, key_abs, value_abs, new_dict

def gene_surrounding(positions, genome, length): # to use in
    loop, input one genome, the positions and the length.
   Output the sequences of upstream and downstream
    start_value, end_value=positions

    if start_value < end_value: # first strand
        if int(start_value)-int(length)<=0:
            length=int(start_value)-1

        start=int(start_value)
        end=int(end_value)

        upstream=genome[start-int(length)-1:start-1]
        downstream=genome[end:end+int(length)]
```

```
    else: # other strand
        if int(end_value)-int(length)<=0:
            length=int(end_value)-1

        start=int(start_value)
        end=int(end_value)
        trans_table=string.maketrans("ATCG", "TAGC")

        upstream_rev=genome[end-int(length)-1:end-1]
        upstream=upstream_rev.translate(trans_table)

        downstream_rev=genome[start:start+int(length)]
        downstream=downstream_rev.translate(trans_table)

    return upstream, downstream
```

A.1.2 R kod

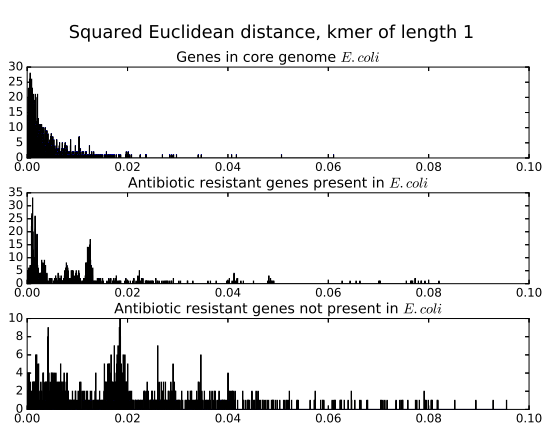
```
# R version 4.0.0
# Creating the model from input text file with present/not
    present, length and values for kmer of length 3
all_data <- read.table(
  "model_file.txt",
  sep="\t", header=TRUE)
all_values<-all_data[,c(-length(all_values))] # remove the
    kmer TTT

kmer_columns<-all_values[3:65] # the columns with kmer

model_list<-all_values[,1:2] # the fixed parameters
kmer_list<-list()
coeff_list<-list()
continue=TRUE
while (continue) # add the kmer with the lowest p-value
{
  for(i in 1:length(kmer_columns)){
  sub_values<-c(model_list, kmer_columns[i])
  fit <- glm(in_ecoli~.,data=sub_values,family=binomial)
  pvalues<-summary(fit)$coefficients[,4] # all p-values
  pvalue_kmer<-pvalues[length(pvalues)] # the last p-value
  # (the added one)
  kmer_list<-append(kmer_list, pvalue_kmer) # append to list

  coeff<-summary(fit)$coefficients[,1] # same for estimates
  coeff_kmer<-coeff[length(coeff)]
  coeff_list<-append(coeff_list, coeff_kmer)
```

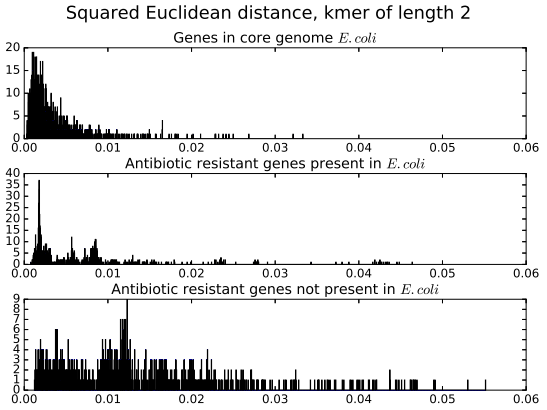
```
}
  low<-kmer_list[which.min(kmer_list)] # find the lowest p-
      value
  if (low>0.001) { ## stop when excedd the p-value cutoff
    continue=FALSE
    print("DONE")
    break
  }
 # save the parameters before zeroing and continue with the
     next round
  model_list<-c(
    model_list, kmer_columns[match(names(low),names(kmer_
        columns))])
  kmer_columns<-kmer_columns[-match(names(low),names(kmer_
      columns))]
  kmer_list<-list()
  coeff_list<-list()
}

model_names<-names(model_list)

# training and testing data, run 100 times
set.seed(33)
random_nr<-floor(runif(100, min=0, max=1000))
smp_size <- floor(0.5* nrow(all_values))
model_names_columns<-all_values[model_names]

specificity_list=list()
sensitivity_list=list()
for(i in 1:100){
  set.seed(random_nr[i])
  train_id <- sample(seq_len(nrow(all_values)), size = smp_
      size)
  train_data<-model_names_columns[train_id,]
  test_data <- model_names_columns[-train_id,]
  glm.fit <- glm(in_ecoli~.,data=train_data,family=binomial)
  glm.probs <- predict(glm.fit,test_data,
                       type = "response")
  glm.pred <- ifelse(glm.probs > 0.5, 1, 0) # cut off for
      the classification

 model_table<-table(glm.pred,test_data$in_ecoli)
  if (dim(model_table)[1]==1) {  # if all values exceed or
      fall below the cutoff
    if(rownames(model_table)==0)
    {sensitivity<-0
```

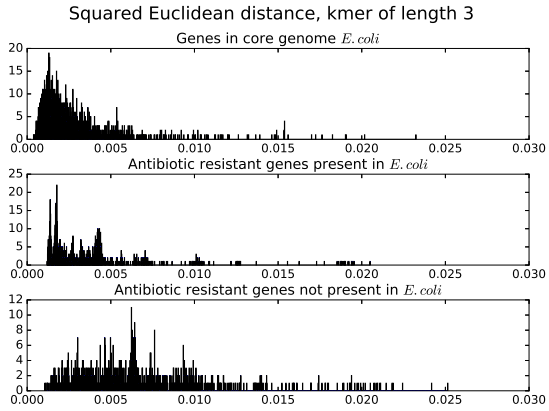
```
    specificity<-1}
    else{sensitivity<-1
    specificity<-0}
  } else{
    specificity<-(model_table[1,1]/(model_table[1,1]+model_
      table[2,1]))
    sensitivity<-(model_table[2,2]/(model_table[1,2]+model_
      table[2,2]))
  }
  specificity_list<-c(specificity_list, specificity) # save
    the specificity into the list
  sensitivity_list<-c(sensitivity_list, sensitivity) # same
    for sensitivity
}
# Print the specificity and sensitivity values
print("specificity")
mean(unlist(specificity_list))
print("sensitivity")
mean(unlist(sensitivity_list))
```

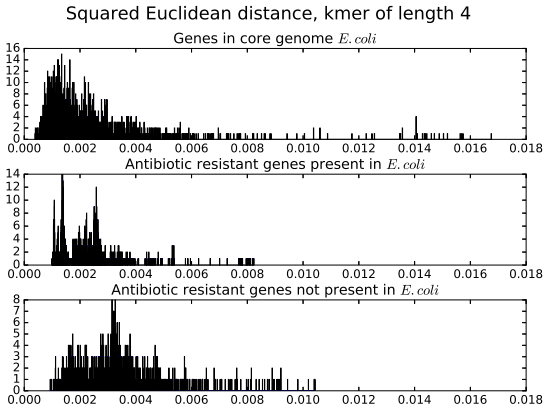
A.2 Plots

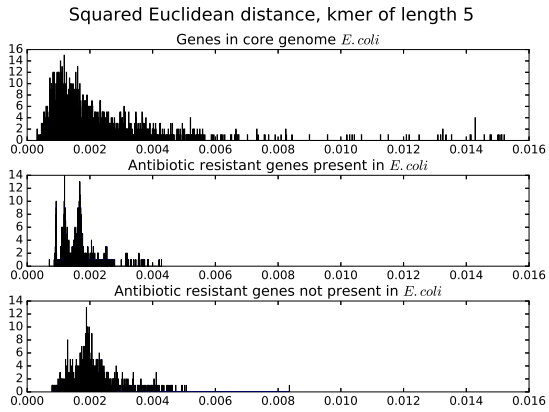
A.2.1 All plots kmer analysis for *E. coli*, squared Euclidean distance

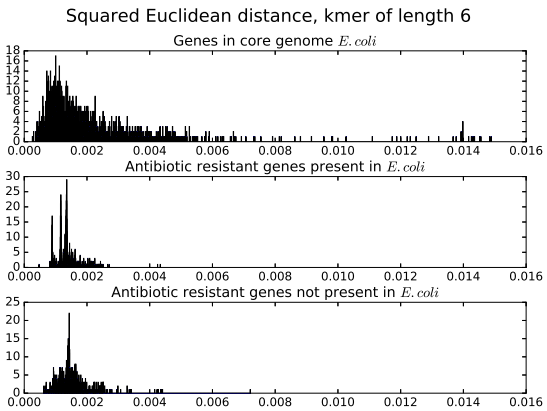


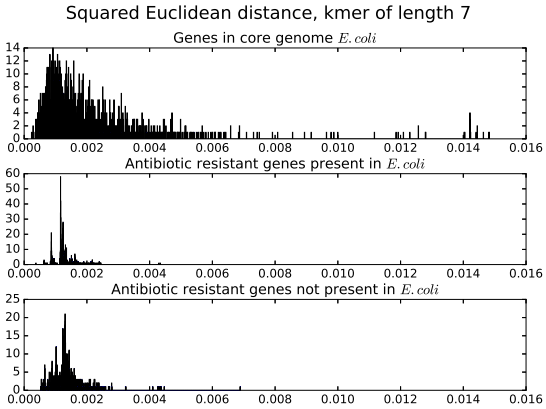


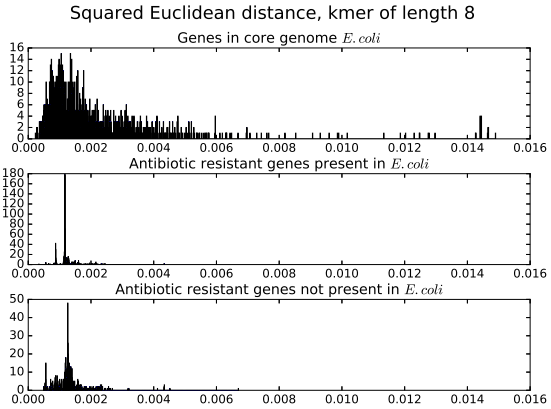




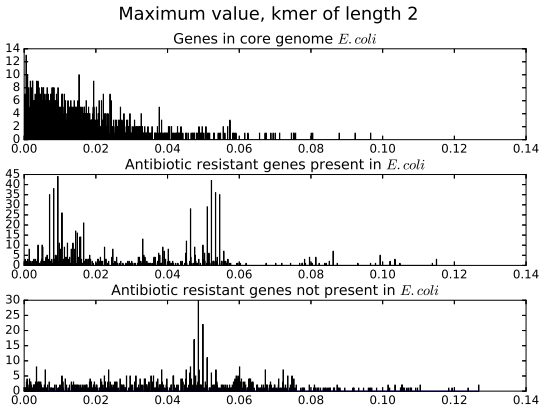




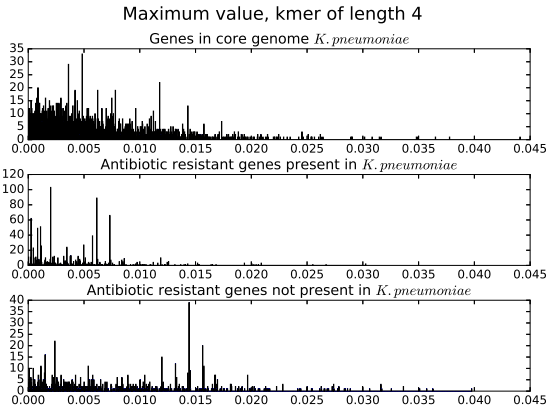




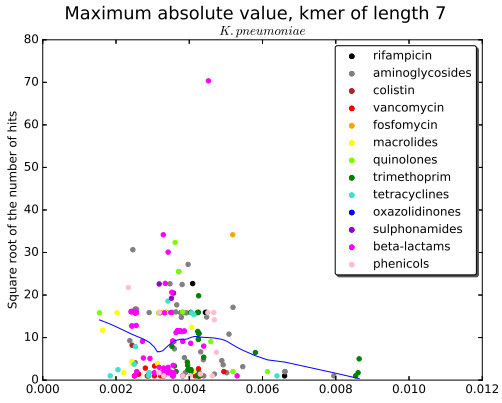
A.2.2 Additional plots



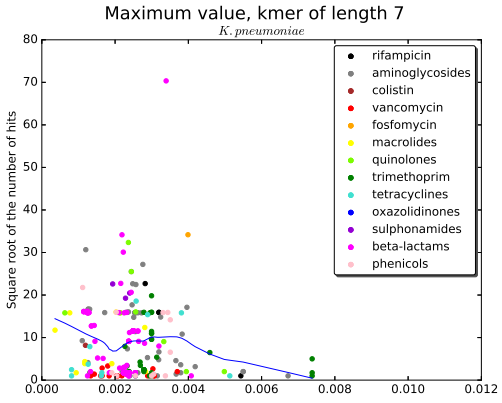
(a) Maximum value, kmer of length 2.



(b) Maximum value, kmer of length 4.



(c) Maximum absolute value, kmer of length 7, median value 0,003423



(d) Maximum value, kmer of length 7, median value 0,002231

A.3 Predictive model

For all genes, the final model included 21 kmers. The values for the kmers, when added to the full model.

Table A.1: Model parameters for all genes for the kmers in the order they were added.

Kmer added	Estimate	P-value
AAG	-98,23	2,05e-58
ACT	129,46	7,50e-35
CAG	97,24	5,42e-27
GAG	73,97	2,23e-15
AGG	-114,80	1,89e-20
ACC	-85,46	3,52e-16
GGA	-75,97	1,40e-10
TGT	-54,05	5,91e-06
CCC	-67,99	2,48e-09
GCT	48,86	6,41e-07
CGG	-50,63	9,57e-08
CGC	40,28	6,48e-07
TAA	42,68	1,29e-06
TTC	77,91	2,77e-09
TTG	60,80	2,18e-06
GCA	59,73	2,16e-05
GTC	56,72	1,52e-04
TGA	61,84	2,61e-05
ATC	56,71	6,96e-05
CTC	-57,04	5,29e-04
GTT	59,06	5,18e-04

A.3.1 Antibiotic classes

A.3.1.1 Beta-lactams

For beta-lactams, the final model included 16 kmers. Values for kmers when added to the full model.

Table A.2: Model parameters for beta-lactams for the kmers in the order they were added.

Kmer added	Estimate	P-value
AAG	-136,29	5,88e-56
AGA	168,84	1,91e-28
ACT	160,81	2,29e-18
GCA	125,31	6,73e-16
CCT	-220,54	7,31e-21
ACA	-165,17	2,57e-13
CAG	157,48	6,28e-12
GTA	189,58	2,07e-09
GAG	120,76	5,64e-07
GAA	226,79	1,02e-13
GTG	-139,69	1,17e-07
TCG	105,77	1,20e-05
CGA	-169,50	2,30e-09
TCT	-140,86	9,61e-06
TTA	120,55	2,12e-06
CAC	137,25	6,75e-04

A.3.1.2 Aminoglycosides

For aminoglycosides, the final model included three kmers, CCT, TAA and TTG. Values for kmers when added to the full model.

Table A.3: Model parameters fo aminoglycosides for the kmers in the order they were added.

Kmer added	Estimate	P-value
CCT	194,92	2,97e-08
TAA	-73,48	1,74e-04
TTG	89,64	5,89e-03

A.3.1.3 Macrolides

For macrolides, the final model included three kmers, GGT, CAG and TTC. Values for kmers when added to the full model.

Table A.4: Model parameters for macrolides for the kmers in the order they were added.

Kmer added	Estimate	P-value
GGT	-197,46	6,74e-05
CAG	-73,48	1,74e-04
TTC	167,95	1,12e-03

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS