Methods and Analysis

Yanru Liao ,Jina Yang, Breanna Richards

2022-11-13

Recall that the aim of this study is to assess if we can predict future isolate's performance against certain antibiotics where 'performance' is a binary discretion of either susceptible or resistant. In other words, can we learn from past isolates to develop a system to recommend antibiotics to new isolates? We are also generally interested in the kinds of patterns that we can extract from our current data about behaviors of drug susceptibility and resistance including how responses to these antibiotics shift over time.

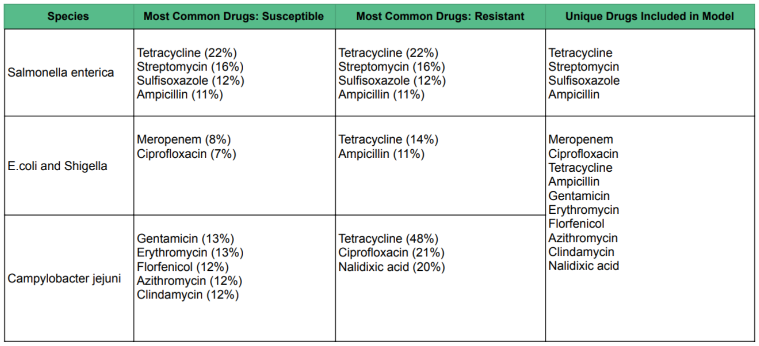
The analysis focuses on isolate data collected from the Salmonella enterica, E.coli and Shigella, and Campylobacter jejuni species in the National Center for Biotechnology Information's (NCBI) Pathogen database. Initially, we had decided to evaluate isolates from these three bacteria within joint modelling and analysis procedures. However, through our data exploration process, we found that our Salmonella isolates had retained two additional pieces of information compared to the E.coli and Campylobacter isolates. This information are antigen formula, which gives the detected presence of a specific viral antigen indicating viral infection, and serovar/serotype which gives the distinct variation within a certain bacteria. Additionally, while isolate data for Salmonella were retained from 2002 - 2021, E.coli and Campylobacter records didn't begin presenting themselves in our data until much later (2009 for E.coli and Shigella, and 2016 for Campylobacter). For these reasons, we decided to address the Salmonella isolates separate from the E.coli and Campylobacter isolates, i.e., we'll be building two separate models to address our main research question of interest: one for Salmonella isolates, and another for both E.coli and Campylobacter.

The full scope of our data contains isolates that are resistant and/or susceptible to 79 unique antibiotics, 76 of which these isolates have shown resistance to, and 62 of which these isolates have shown susceptibility to. Evidently, whereas some isolates have shown resistance to a certain durg, other isolates have shown susceptibility to the same drug. The idea is to create a system or function that will take information about isolates as input values and then within a subset of most common drugs we've found the isolates to show the most response to, reccomend the one that we've predicted with the most evidence that new isolate is most susceptible to, along with the level of confidence that we have in this recommendation (i.e. the probability that we obtain from our classification model). Therefore, since this is a binary classification problem, on the data that we'll use to develop our models, we'll code '1' as drug susceptiblity, '0' as drug resistance, and NA if we do not retain any information about that particular drug on a given particular isolate.

As it's not entirely feasible for us to predict drug response to each and every one of these 79 drugs, we decided to first focus on predicting responses to the most common drugs isolates represented in each of our now two separate datasets, one set for Salmonella, and a second joint set for Campylobacter and E.coli. Antibiotics up for consideration in our model fulfill one of two criteria: 1) represents at least 10% of all drugs within a given bacteria that were reported for isolate susceptibility or resistance, or 2) in the case that no drug represents at least 10% of all drugs reported for susceptibility or resistance within a given bacteria, be within the top two most common drugs that isolates of a given bacteria have shown susceptibility or resistance to. **Table 1** shows the most common antibiotics isolates within each bacteria showed resistance and susceptibility to. We combine the unique set of these antibiotics as the antibiotics that we will include in our model.

The final piece of pre-processing that our data has undergone involves the reformatting of the prevalence of the antimicrobrial resistant genes found in the isolates. There are 499 unique resistant genes found in the isolates within our data. In order to include some level of gene resistant presence as predictors in our

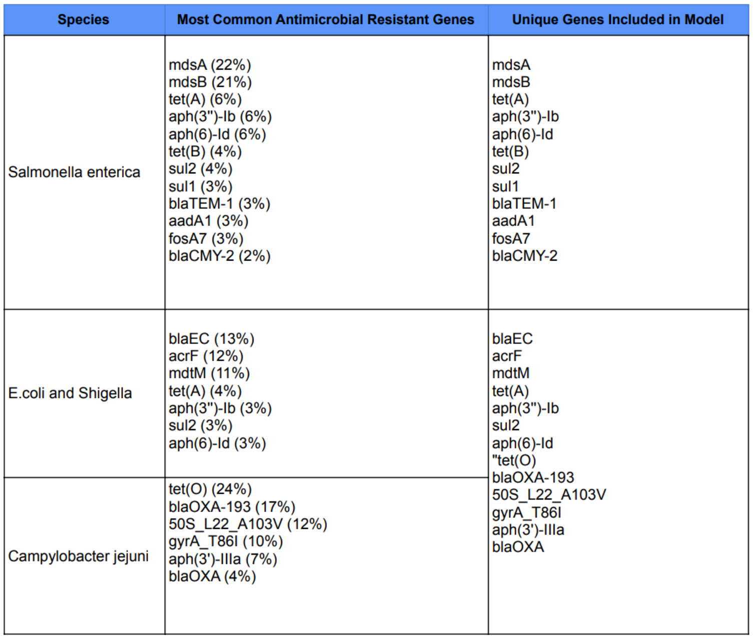
Table 1: Antibiotics Considered in Modeling



Species	Most Common Drugs: Susceptible	Most Common Drugs: Resistant	Unique Drugs Included in Model
Salmonella enterica	Tetracycline (22%) Streptomycin (16%) Sulfisoxazole (12%) Ampicillin (11%)	Tetracycline (22%) Streptomycin (16%) Sulfisoxazole (12%) Ampicillin (11%)	Tetracycline Streptomycin Sulfisoxazole Ampicillin
E.coli and Shigella	Meropenem (8%) Ciprofloxacin (7%)	Tetracycline (14%) Ampicillin (11%)	Meropenem Ciprofloxacin Tetracycline Ampicillin Gentamicin Erythromycin Florfenicol Azithromycin Clindamycin Nalidixic acid
Campylobacter jejuni	Gentamicin (13%) Erythromycin (13%) Florfenicol (12%) Azithromycin (12%) Clindamycin (12%)	Tetracycline (48%) Ciprofloxacin (21%) Nalidixic acid (20%)	

model, we chose to include binary variables for the most common resistant genes observed in our data. These variables took a value of 1 if found within a given isolate, and a value of 0 is not found within a given isolate. The genes that we decided to include as predictors in each of the two separate models (one for Salmonella, and another jointly for E.coli and Campylobacter) are displayed in **Table 2**. We decided to retain the genes with over 2% prevalence in the Salmonella isolates, and the genes with over 3% prevalence in either the E.coli isolates or the Campylobacter isolates. A higher prevalence threshold of 3% was chosen for E.coli and Campylobacter as to not overpopulate the joint E.coli and Campylobacter model with these genetic predictors.

Table 2: Antimicrobial Resistant Genes Considered in Modeling



Species	Most Common Antimicrobial Resistant Genes	Unique Genes Included in Model
Salmonella enterica	mdsA (22%) mdsB (21%) tet(A) (6%) aph(3'')-Ib (6%) aph(6)-Id (6%) tet(B) (4%) sul2 (4%) sul1 (3%) blaTEM-1 (3%) aadA1 (3%) fosA7 (3%) blaCMY-2 (2%)	mdsA mdsB tet(A) aph(3'')-Ib aph(6)-Id tet(B) sul2 sul1 blaTEM-1 aadA1 fosA7 blaCMY-2
E.coli and Shigella	blaEC (13%) acrF (12%) mdtM (11%) tet(A) (4%) aph(3'')-Ib (3%) sul2 (3%) aph(6)-Id (3%)	blaEC acrF mdtM tet(A) aph(3'')-Ib sul2 aph(6)-Id "tet(O) blaOXA-193 50S_L22_A103V gyrA_T86I aph(3')-IIIa blaOXA
Campylobacter jejuni	tet(O) (24%) blaOXA-193 (17%) 50S_L22_A103V (12%) gyrA_T86I (10%) aph(3')-IIIa (7%) blaOXA (4%)	

Justification for why this plan will answer your question

The ultimate goal is to build a model or a series of models, rather, that will, out of the antibiotics that we chose to consider, recommend to new isolates the antibiotic predicted with the highest level of confidence that that new isolate will show susceptibility to. Approaching this task with the random forest classifier model is appropriate as we want to predict either a susceptible or resistant response for our isolates on each of the antibiotics and we can assess our levels of confidence as random forest implementation in R doesn't only return the predicted class of each observation, but also the probabilities associated with those predictions. Additionally, random forests is a type of non-parametric supervised machine learning model so we don't have to be held to stringent assumptions of our data like we would with parametric techniques like logistic regression. We are also interested generally in the kinds of patterns that we can extract from our current data, especially when it comes to the most pertinent covariates in the classification processes. We want to find which antimicrobial genes as well as which baseline covaraites are most helpful in deciding how to treat cases of foodborne illness in Salmonella, E.coli, and Campylobacter. And so, what is especially handy about random forests is its emphasis on feature selection.

As feature selection is a built-in mechanism on random forests, the output of our models are able to provide a simple measure of how integral each variable of interest is to classification. This process will be paramount in extracting tangible insights about associations between drug susceptibility and resistance, and the supporting characteristic information logged in the NCBI database about each new case. Even pertaining to our goal of assessing how time plays a roll in predicting drug response, the mechanisms of the random forest algorithm in R makes it easy to assess the significance of time comparative to the other variables considered in modeling due to this variable importance feature.

The biggest limitation of implementing this model on our data is that it assumes our data are complete and non-missing. As the nature of the NCBI database is generally incomplete, this is a hurdle that we have to thoughtfully address. While we can impute these missing values in a pretty straight-forward process, when we have larger amounts of missingness, these imputed values become less and less reliable.

Initial implementation(s) - do a brief use case as a sanity check to make sure you understand how to use the methods and to catch any potential problems.

We'll demonstrate the utilities of the `randomForest` function from the `randomForest` package in R on our data with a simple use case (Liaw and Wiener, 2002). For this example, we used our dataset for Salmonella enterica isolates and focused on the isoaltes' resistant/susceptible response to the antibiotic, tetracycline.

However, as aforementioned, before we ran this implemention, we first utilized the `Boruta` function from the `Boruta` package in order to perform variable selection (Kursa and Rudnicki, 2010). In action, the Boruta algorithm first adds randomness to the dataset of interest by creating shuffled copies of all of the features up for consideration. These new shuffled copies are called shadow features. Next, the function applies the random forest on the data and evaluates the importance of each of the features in the classification process. With each iteration of the classifier, the Boruta algorithm checks if the un-shuffled, original feature has a higher importance than the most important of its shuffled shadow features. If an original covariate is not deemed more important to classification than the best of that covariate's associated shadow features, then it is deemed unimportant and removed as a variable worth keeping in the modeling process. The Boruta algorithm stops when either all features are either confirmed or denied as being important, or alternatively when the maximum number of a specified number of random forest runs is reached.

A hurdle of utilizing this algorithm is that it only accepts non-missing data. As our current data currently obtains missing values, for the sake of this demonstration, as mentioned in data-preprocessing, we imputed missing values by utilizing the `mice` function from the `MICE` package in R.

To recap the algorithm, Multivariate Imputatoin via Chained Equations (MICE) assumes Missing at Random (MAR) data and imputes on a variable by variable basis, meaning that a new imputation model is specified with each new variable with missingness. Linear regression is used to predict missing values for continuous variables, and logistic regression is used to predict missing values for categorical variables.

After ensuring that all of the categorical variables in our data are expressed as factors, we imputed our missing data.

Our use of this imputation algorithm will be more heavily scrutinized for the actual implementation of our methods on our data, but for now, we will impute values for any column without much criticism towards the values that are actually being imputed.

We will only work with 1 generated imputed dataset for the sake of this example.

```r
imputed_dat <- mice(data = tetra_test, m = 1,
     method = "pmm", maxit = 50,
     seed = 500)
```

Table 3: Tetracycline RF Performance Metrics

Metric	Value
Accuracy	0.9758994
Sensitivity	0.9618741
Specificity	0.9900417
Pos Pred Value	0.9898369
Neg Pred Value	0.9626210
AUC	0.9951389

After imputing our data, we then move on to the `Boruta` feature selection process.

```r
boruta <- Boruta(tetracycline ~ .,
data = imp_data, doTrace = 2, maxRuns = 500)
```

From 12 iterations, the Boruta algorithm decided that none of the 23 predictors of consideration were unimportant, so we will will impute them all into the example classification algorithm.

We fit a model with drug susceptibility to the antibiotic, tetracycline, as our outcome of interest (a value of 1 for susceptible and a value of 0 for resistant), and the variables collection date, isolation type, minimum SNP distance to an isolate of the same isolation type, minimum SNP distance to an isolate of a different isolation type, serovar, country, isolation source, number of isolates "close" to that isolate (being close to another isolate being defined as having a minimum SNP distance less than or equal to 7 to an isolate of either the same or different isolation type), antigen formula, average similarity scores between isolates within the same SNP cluster of antibiotics that isolates show resistance and susceptibility to, and finally a set of 12 binary variables indicating the genetic presence of 12 of the most common antimicrobial resistant genes in Salmonella.

Note that our outcome of interest, susceptibility on tetracycline is roughly balanced with 51.3% of the isolates showing resistance to tetracycline, and the remaining 48.8% showing susceptibility.

```r
set.seed(1)

rand_forest_test <- randomForest(tetracycline ~.,
data = imp_data)
```

The output of the random forest command gives the type of random forest, which is classification in this case because our outcome is binary, the number of trees that were grown as a part of the process which in this case was the default 500 trees, and the number of variables tried at each split, which was 4, i.e. the value found by taking the largest integer value less than or equal to, otherwise known as the *floor* value of the square root of the number of columns in the dataset (floor of the $\sqrt{24}$).

The random forest function also provides a measure of the out of bag estimate of the error rate. While this is reported as 2.41% in the model output, we can also plot the out of bag error against the number of trees as well as plot this by class (resistant (0) vs. susceptible (1)).

Notably, **Figure 1** shows that our model is stronger at predicting the tetracycline susceptibility class than the resistant class, a plot that can be supplemented with other performance related visuals like ROC curve.

We also can extract the statistics of interest based on the confusion matrix outputted from predicted classification. We just collected this on our full dataset for now just for demonstration purposes (**Table 3**).

We will also use importance measures to extract information about the most important predictors in these cases of predicting drug response. Ultimately, we'll be comparing the results and important predictors of

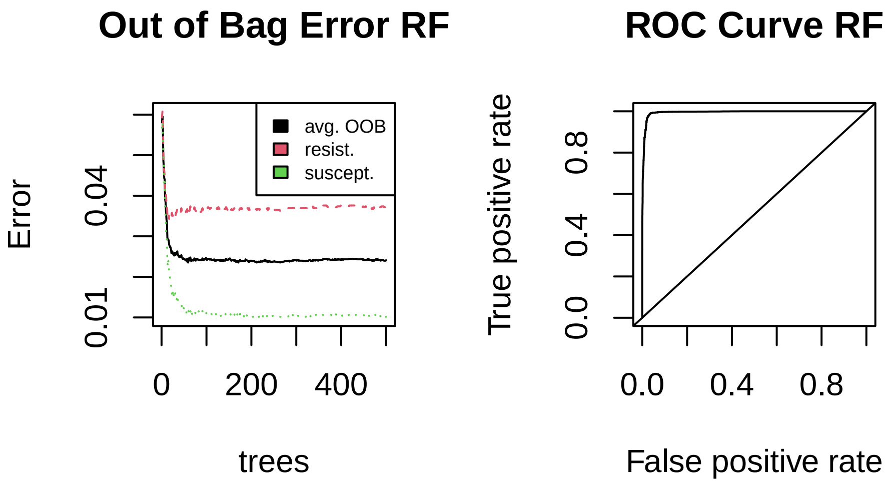


Figure 1: Out of Bag Error Plot and ROC Curve for Tetracycline RF

multiple models with susceptibility/resistance responses to each of the antibiotics of interest, in order to recommend antibiotics that we predict new isolates will show susceptibility to. In this simple case though, we show that we can extract the most important variables integral to the random forest classification of our data (Figure 2).

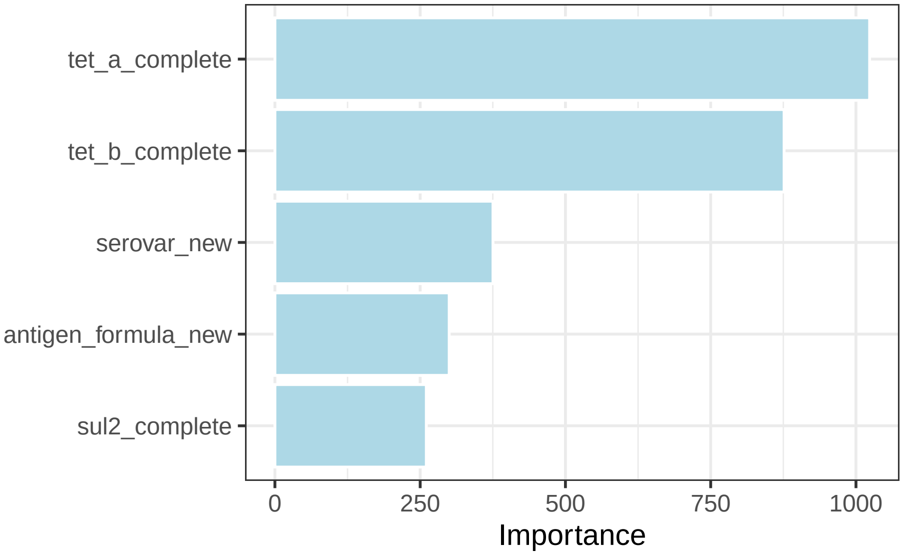


Figure 2: Top 5 Most Important Covariates

Moving forward, we'll work to properly tune the hyperparameters within our random forest models and systematically expand this toy example to predict isolates' performances on larger sets of antibiotics which will hopefully retain usefulness in the assessment of future isolates to come.

References

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

Miron B. Kursa, Witold R. Rudnicki (2010). Feature Selection with the Boruta Package. Journal of Statistical Software, 36(11), 1-13. URL http://www.jstatsoft.org/v36/i11/.