Dean Gordon Placement Report

MSc Data Analytics, Queen's University, Belfast

Dean Gordon

9/13/2019

Abstract

Data on pupils' preferences for post-primary schools are available from cohorts who transferred in 2017, 2018 and 2019. These have been combined with other data, including pupil-level and school-level attributes to attempt to identify:

1. How many applications schools are likely to receive in the coming years
2. Where one school undergoes a significant change, which other schools are likely to be impacted, in terms of applications

A number of methods are examined for predicting the number of first preference applications that a school will receive in future years, including machine learning models (random forests and neural networks) and a transition matrix that takes a simple proportion (by gender) of pupils from each primary school to predict which post-primary school is their first preference. A number of these methods were able to make useful predictions of the number of first preference applications at each school, although none were able to make accurate predictions on a pupil level, and indeed we will show that the pupil data available is not adequate for making these predictions accurately.

The transition matrix, as well as performing best in making predictions of first preference applications, can be adapted as one way to identify other schools that will be impacted by significant changes to a school. Second preference data offers another way to identify other schools whose first preference applications could be impacted by significant changes to a school. The most salient insights from the methods above are gathered together to inform decision making for a particular school or set of schools, with this information presented in an interactive application using Shiny.

This report has been redacted to remove any references to individual pupils or schools.

1. Introduction

Pupils in Northern Ireland are unconstrained in their selection of preferences for post-primary schools. However, allocations are constrained by external limits applied to each school, limiting the number of admissions permitted from primary schools each year. Selection criteria, including academic selection, are set individually by schools and used to rank applicants: pupils will not be denied a place for failing to meet a standard, but rather because another pupil has been ranked higher based on the selection criteria. The Education Authority (EA) is responsible for ensuring sufficient places are available in appropriate schools for each pupil transferring to post primary school through their area planning responsibilities.

Forecasting future demand for post-primary school places, taking into account the various categories of schools (secondary or grammar; Roman Catholic, Non-denominational or integrated; co-educational or single-sex) is vital for the EA's area planning goal, which is "to establish a network of viable schools that are of the right type, the right size, located in the right place, and have a focus on raising standards."[1] Enrolment data are available, showing the schools to which pupils have been allocated and recently preference data have been collected, recording first and subsequent preferences for primary 7 pupils transferring to post-primary schools. The goal of this paper is to combine the preference data now available with pupil-level and school-level attributes, to gain insights and make predictions pertinent to strategic area planning that meets the stated objectives of the EA.

In addition to making predictions about first preference applications to individual schools, this paper seeks to understand the relationships between schools so as to understand the impact that changes in one school will have on another. For example, if one school was to close, where would we expect the pupils who chose it as their first preference to go instead? Or, if a school was to be allowed to accept more pupils, would that lead to other schools not getting enough applications to fill their available places? These questions cannot be answered with certainty and there is much local context and domain expertise that should be applied when looking into them. However, drawing together the relevant data can be very informative when considering possible answers to these questions.

Therefore, this paper looks at a number of models to make predictions, at a school level, of future first preference applications. Secondly, it uses second preference data and data about feeder primary schools for each post-primary school to indicate the relationships between post-primary schools, using a variety of means to identify which schools are likely to be impacted by changes to a selected school or schools.

[1] https://www.eani.org.uk/school-management/area-planning/what-is-area-planning

2 Methods

2.1 Preliminary Investigation

Before exploring methods for predicting first preference applications, we will examine whether the pupil attributes available are discriminative of school choice at the individual level (although the purpose of our models is to make accurate predictions for first choice applications at a school level). The variables available are:

Variable	**Contains**
Post code	Post code of pupil
Other Location Variables	Less precise locations variables, such as Super Output Area[1] and Small Area[2] have been inferred from the post code
Primary school	Department of Education Number
Dayboard	yes or no
Special Educational Needs	stages 0-5
Attends a special unit	yes or no
Free school meals	entitled or not entitled
Gender	male or female
Date of birth month	Number
Date of birth year	Number
Religion	Catholic, Protestant, Other Christian, Other non-Christian, none/not known
newcomer	yes or no
In care	yes or no
Irish medium unit	yes or no

A subset of the data, looking at the 2017 transferring cohort, is created containing only pupils in the majority set for each characteristic: that is, those who were not entitled to free school meals, did not have any special needs indicated, were not in care etc. Having created this subset, the only thing to distinguish pupils was their religion, date of birth, where they lived and primary school. Also included was whether their first preference was a grammar, secondary or integrated school (even though this will not be available for making predictions) but ignored the date of birth data, as it may have reduced the number of 'identical' pupils to compare (birthdays earlier in the school year have been shown to correlate to preference for grammar over secondary schools[2], which has been included explicitly). From this, ten subsets of data were created, consisting of male and female sets of Catholic, Non-denominational and Integrated schools, with Catholic and Non-denominational schools further divided into secondary and grammar schools. For each Super Output Area, the number of distinct schools chosen by pupils with similar attributes (including primary school attended) and identical preferences for the characteristics of their first preference school is calculated. This will indicate whether it is possible, with the data available, to determine which post-primary schools will be selected as first preference by individual pupils.

2.2 Machine Learning Models

To facilitate building models, pupil data and preference data were merged from different sources in R[3] using *tidyr*[4] and *dplyr*[5], and the *caret*[6] package was used with the whole of the 2017 dataset used for training and 2018 used for testing. The random seed was set to 1 before each model was created.

[2] According to unpublished analysis by the team at SIB

2.2.1 Random Forest

A random forest model was created in R (using *ranger*[7]) and 10-fold cross validation was used in model training, with the random forest consisting of 500 trees. In creating the model, class probabilities were saved so that the probabilities of first preference applications to each school could be aggregated to estimate first preferences for each school, rather than counting the most likely first preference of individual pupils.

This method was used as, although not a high dimensionality of data, the algorithms used to combine the various decision trees and weight them accordingly could be expected to work well to provide class probabilities given variables that could interact in complex ways.

2.2.2 Neural Network

A neural network was also created in R using the *nnet*[8] package. A 5-fold cross-validation was used with maximum of 100 iterations for optimising the weights. 200, 1,000 and 5,000 iterations were also used, but without the cross-validation, due to time considerations. In each case a feed-forward neural network with a single hidden layer was created. A neural network lends itself to outputting probabilities for outputs based on a set of inputs. The identical inputs leading to different outputs will not be helpful for optimising weights, but the model should not become too confident on any of those outputs. A neural network was also suitable for dealing with the large number of factors in the primary school and SOA inputs to the model.

2.2.3 Transition Matrix

Instead of training a machine learning model, this approach simply took that proportion of pupils from each primary school (broken down by gender) who selected each post-primary school as first preference. Predictions are then made by looking at the number of transferring pupils in subsequent years and applying the same proportions from each primary school to post-primary schools. The transition matrix has also been adapted to operate based on the SOA, SA, parliamentary constituency, travel to work area and ward of a pupil, instead of their primary school. Too small an area will lead to unstable predictions, as individual pupils will have a large impact on the proportions. Areas that are too large will fail to capture differences in behaviour of pupils within the larger groups.

2.3 Additional Analysis

Three methods have been used to identify schools that may be affected by changes to another school or schools:

1. **Transition matrix**. A school is chosen and “closed”, meaning each pupil who selected it as first preference is “sent back” to their primary school and the proportions for the primary school are adjusted to allow for the “closing” of a post-primary school. Grammar or secondary preferences are maintained, so “closing” a secondary school will only show the other secondary schools that share the same feeder primary schools.
2. **Second preferences**. This shows the second preferences for those whose first preference was the school that was chosen to “close”. As the second preference data is incomplete, second preference data over the three years, 2017-2019, are considered together. These numbers are scaled so that the sum is equal to the number of pupils who went to the school which was “closed”.
3. **First preference, where the selected school is second preference**. As above, data from three years are included. However, the scaling takes into account the number of pupils who applied to each of the first preference schools.

These tables were built into a shiny app to allow for easy exploration.

2.3.1 First and Second Preferences

Missing second preferences are imputed by assigning second preferences in proportion to the second preferences recorded, for a given first preference. This will not be reliable in all cases (such as where zero or a small number of second preferences are recorded).

The second preference data will be able to capture the relationship between grammar and secondary schools that the transition matrix overlooks. Additionally, we expect to see some similarities between the schools identified by the transition matrix and those identified by both ways of looking at the second preference data.

We will compare the top six schools identified as likely to be most impacted by: the transition matrix; second preference where the selected school is first preference; and first preference where the selected school is second preference. The intersection of schools in each of these top six lists is computed where there are at least four results in each of the lists. More robust methods, such as calculating Spearman's rank correlation coefficient, are not appropriate given the lists, for various reasons, will have different members. The comparisons will allow an assessment of whether the transition matrix and second-preference lists are capturing the same relationships between schools.

3 Results

The preferences file for 2017-2019 contained 68,772 first preferences, although only 63,066 of these contained both a pupil identifier that could be linked to the pupil file and a valid post-primary school identifier. First preferences that contained an invalid school reference were retained to avoid systematically overestimating first preferences for schools. The 2017 training set contained 20,712 pupils with first preferences, and the test set from 2018 contained 21,011 pupils with first preferences. The data contains 818 primary schools and classified pupils into one of 197 post-primary schools.

3.1 Preliminary Investigation

Figure 1 shows the highest number of post-primary schools being selected as first preference by one of the ten groups of pupils from the dataset of similar pupils.

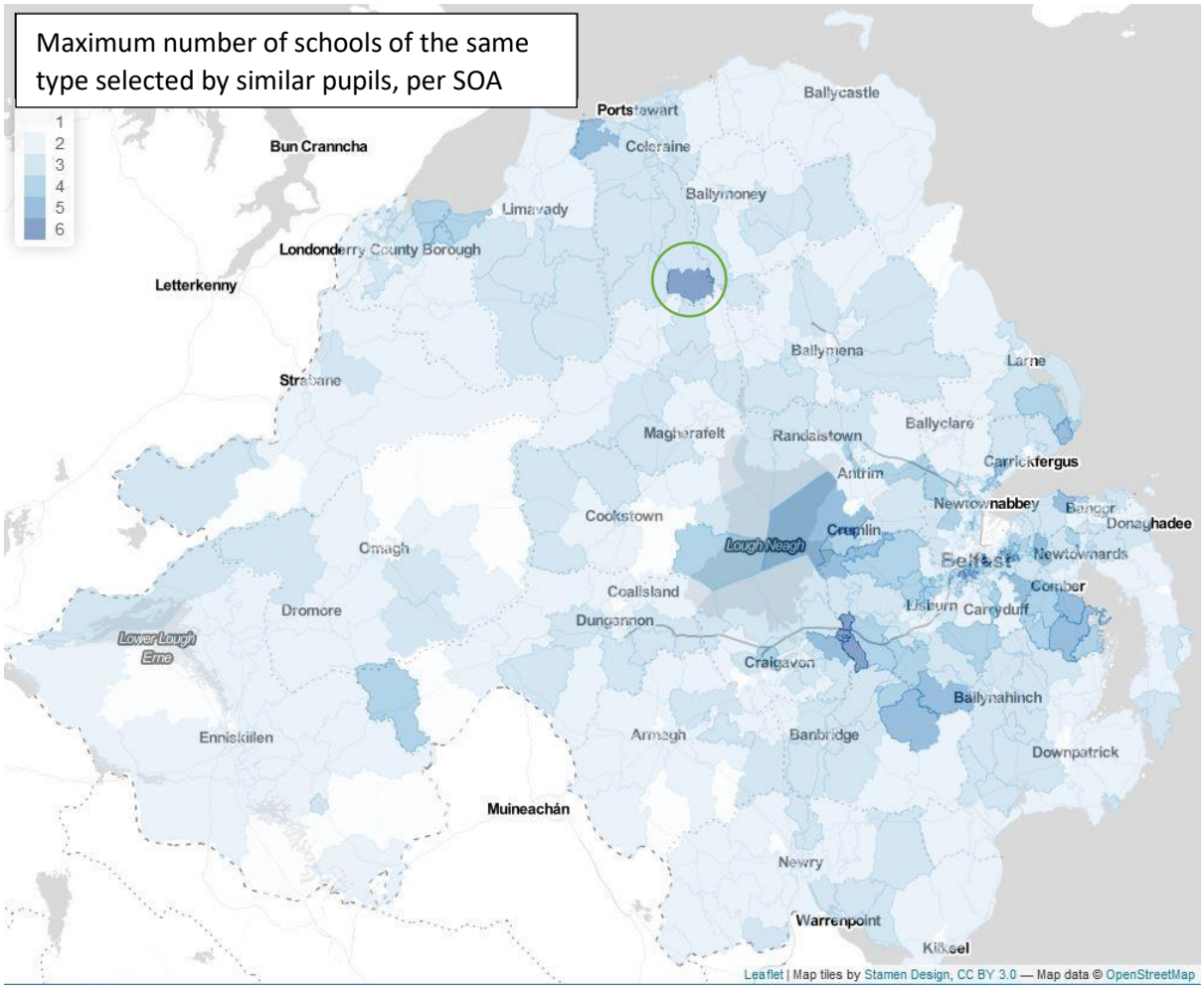

Figure 1

Each of the ten subsets contained at least one SOA with five similar pupils who selected at least four different schools of the same type. In two cases, primary schools have pupils, who are indistinguishable in our data, which have six different first preferences for post-primary schools of the same type. This indicates that the pupil attributes available are not, in many cases, adequate to make definitive predictions about school choices, and in many cases can only narrow preferences down to 3-4 schools.

Figure 2 shows lines linking the Kilrea SOA (green circle in figure 1) with the towns containing the first preference schools selected by pupils across the various categories (religious affiliation; secondary, grammar; gender).

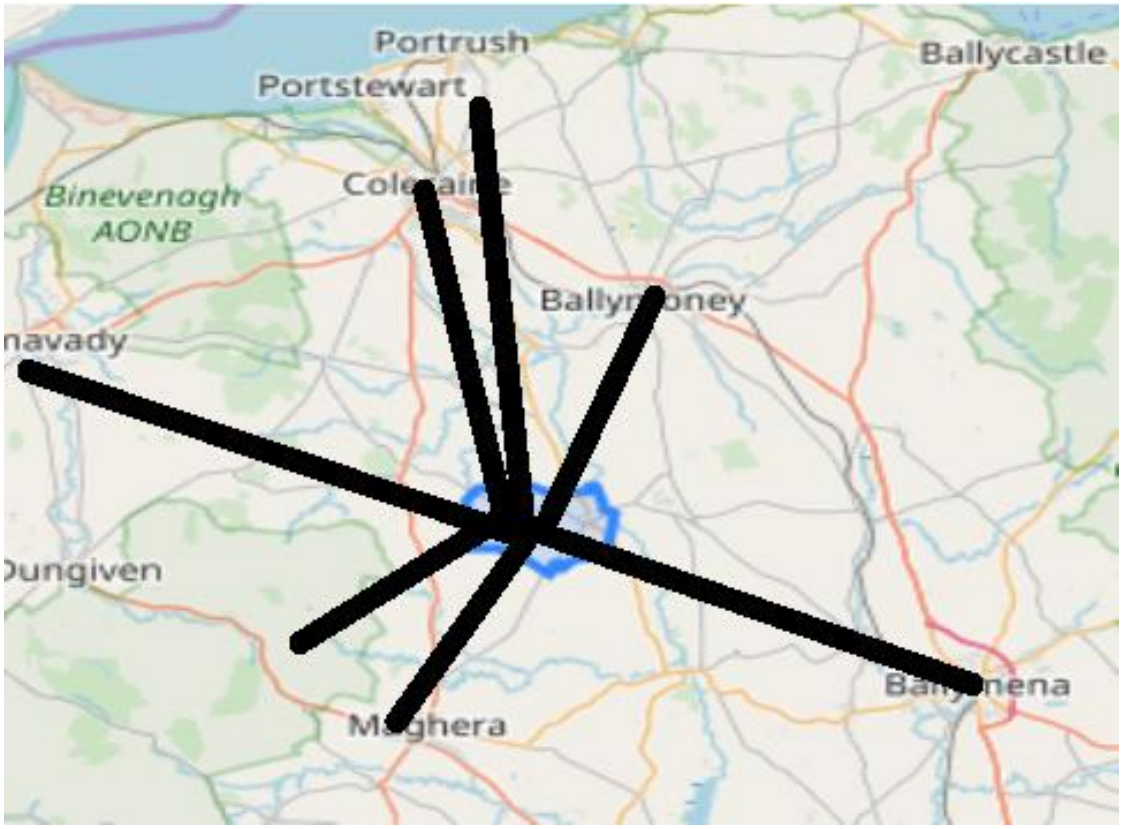

Figure 2

A number of schools are approximately equal distances from this SOA. Choices may be influenced by work location of parents or public transport links, but it appears that these pupils have a number of similar choices and make those choices independently. It is likely that many pupils themselves have difficulty choosing their first preference school, and so a model based on pupil and school attributes cannot be expected to perform well with predictions for individual pupils.

3.2 Machine Learning Model

3.2.1 Random Forest Model

The random forest model was able to correctly predict school choices in 36% of cases, taking the primary school of the pupil as the most important factor, only slightly more important than the Small Area of the pupil (100 and 97.5 respectively on a normalized measure of variable importance). Perhaps surprisingly, the SOA was the next most important variable (65.8) and the other location variable, Travel to Work Area (TTWA) made a smaller, but non-negligible contribution (15.1).

The predictions for first preference applications to each school showed a reasonable predictive value: the chart below shows the model created on 2017 data applied to 2018 inputs and has an R^2 value of 0.89. NAs were removed to calculate the R^2, but had been kept in up to this point to reflect that a valid first preference had not been recorded for all pupils.

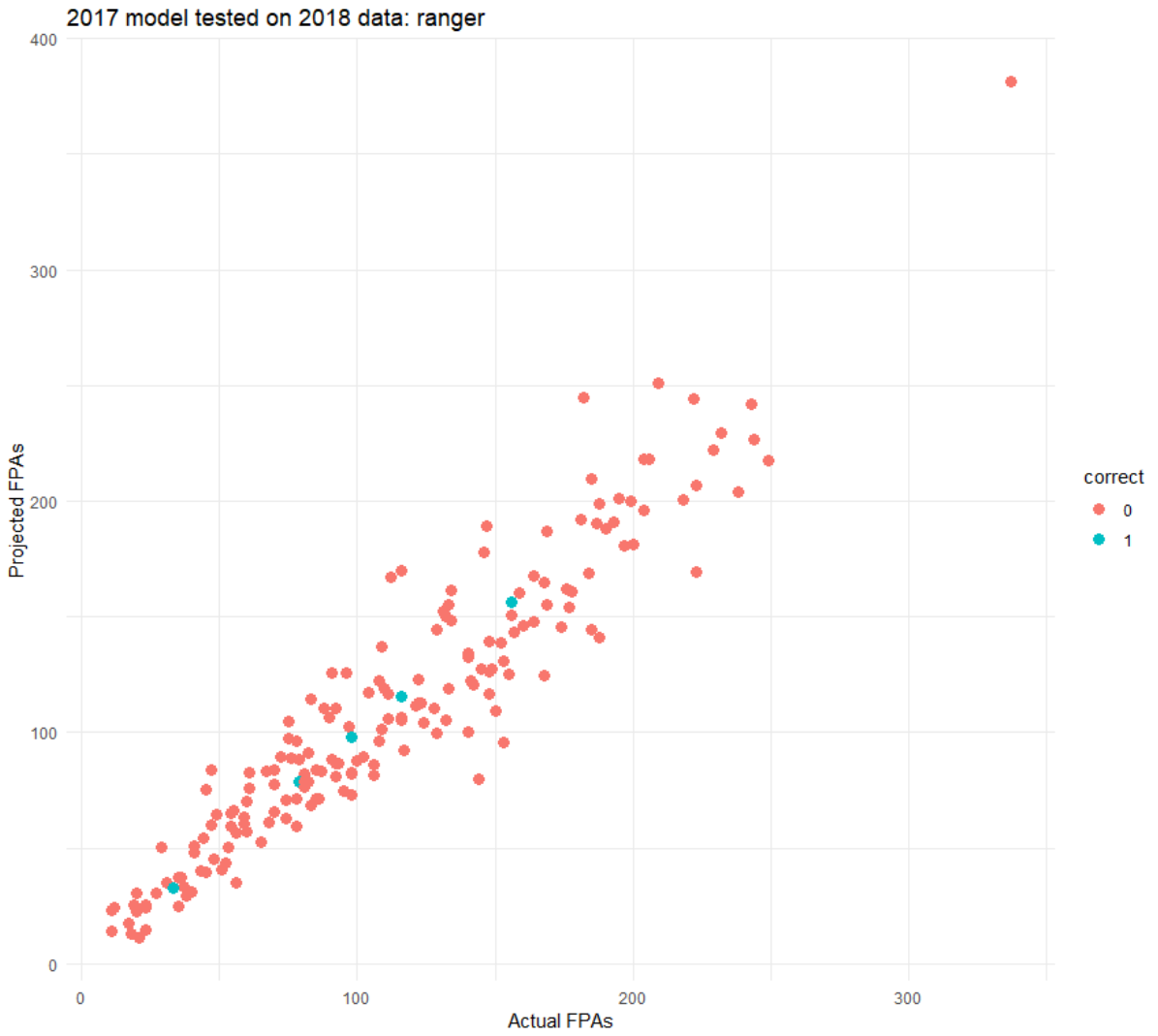

Figure 3

The blue dots in Figure 3 show where predicted first preferences, to the nearest integer, are equal to actual first preferences.

3.2.2 Neural Network

Various parameters were tried for the neural network, with results as shown in the table 1, with NAs removed as above.

Table 1

Summary of R2 values on training and test data of neural networks with various tuning parameters

Cross-Validation	Max Iterations	R^2 on training data	R^2 on test data
5-fold	100	0.9812	0.8720
None	200	0.9930	0.8693
None	1000	0.99996	0.8816
None	5000	0.999996	0.8829

Although for the fourth configuration max iterations was set to 5,000, convergence was reached before iteration 2,220. Although the R^2 value was similar when the probabilities were aggregated up and compared on a school level, the accuracy at an individual level was only 8.8%, suggesting that the model was assigning higher probabilities to more popular schools, but not taking all of the variables and their interactions into account.

3.2.3 Transition Matrix – Primary School

The transition matrix did not make predictions about individual pupils. Based on a transition matrix created on 2017 data, accounting for gender and primary school, an R^2 of 0.92 was achieved when applied to 2018 data and 0.90 when applied to 2019 data, with NAs removed as was the case with the other models.

2014 preference data (which was less complete than 2017-19) was used to create a matrix and this was applied to 2017-19 data, to check the stability of the matrix approach over time: this showed R^2 values of 0.83, 0.81 and 0.78 for 2017 to 2019 respectively. The decreasing values of R^2 for the matrices indicates that their predictive power is likely to reduce over time, although the model from 2014 is still informative, even when applied to 2019 input data. Because there were more missing or invalid first preferences in 2014, there is a systematic under-prediction of first preferences for 2017-2019. Some of the less reliable predictions have been circled in Figure 4.

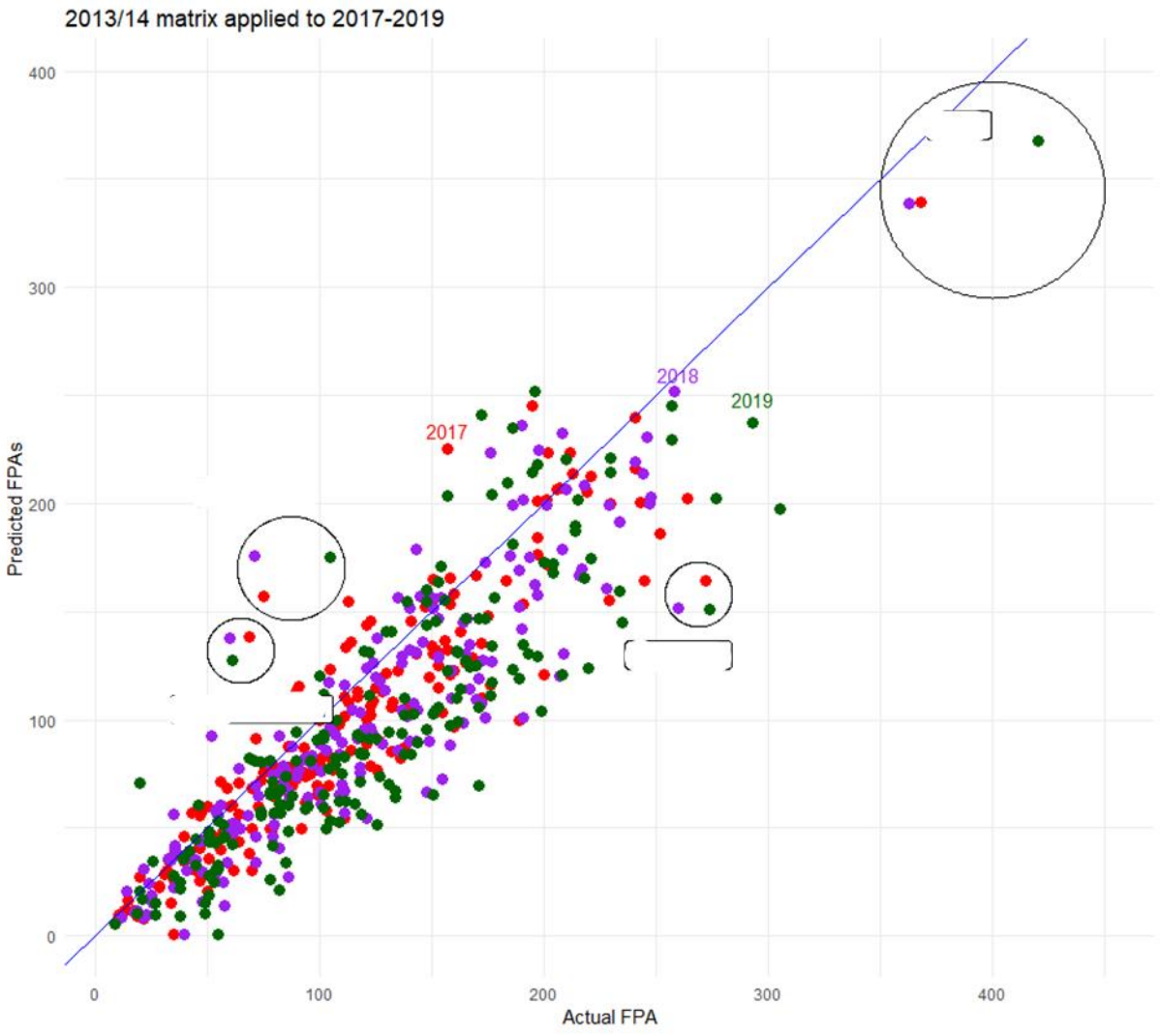

Figure 4

In each of the years 2017-2019, the transition matrix has predicted too many first preference applications for 2 schools, both of which had received negative publicity between 2014 and 2017. The identity of the schools has been redacted, although the information is in the public domain.

At the other end of the spectrum, predicted first preference applications for another school were consistently lower than actual first preferences. Estimates for 2017-2019 were based on 2014 first preferences for three schools that merged to become a single school, and it appears that the combined school is more appealing than at least one of the original three. Most of the other points furthest from the line do not apply to the three predictions made for the same school and so may reflect circumstances that only applied in one or two of the three years, or random variance.

3.2.4 Transition Matrix – Area-based

The SOA transition matrix performed equivalently with the primary school transition matrix, with R^2 values of 0.92 and 0.90 for 2018 and 2019 respectively. The formulae for the best fitting straight lines (ordinary least squares) did not help to choose a preference, with the primary school matrix returning a formula of *Predicted* = 1.0175 * *Actual* + 0.3436 compared to the SOA formula of *Predicted* = 1.0219 * *Actual* -0.05931. Although arguments could be made about the trade-off between the intercept and the slope, in both cases the differences are very small: this is equivalent to 1 predicted pupil for every 228 actual pupils, so it is much less than the mean absolute errors, which range from 14 to 17 across both models and both years (see table below).

A similar matrix was produced using a number of different geographies, none of which performed as well as primary schools or SOAs. Table 2 summarises all variables used in the transition matrices (all matrices built on 2017 data and tested on 2018 and 2019):

Table 2

Summary of R2 and Mean Absolute Errors for 2017 transition matrix created using primary school or geography

Geography	R^2 2018	R^2 2019	MAE 2018	MAE 2019
Primary School	0.92	0.90	13.8	17.1
SOA	0.92	0.90	15.1	17.5
Parliamentary Constituency	0.91	0.90	15.4	17.6
TTWA	0.91	0.89	15.4	17.5
Ward	0.91	0.89	15.4	18.0
SA	0.90	0.88	16.9	19.6

3.2.5 Comparison of prediction methods

Looking at the R^2 values of the models, the performances were similar, although even the worst performing transition matrix performed better than either of the machine learning models. However, it needs to be noted that predicting 2017 actual applications for 2018 and 2019 applications would achieve R^2 values of 0.91 and 0.87 respectively, also out-performing the machine learning models. Moreover, as the model is being trained at a pupil level but evaluated at a school level, we need to be wary of only looking at the R^2 values. Using 2017 first preferences as predictions for 2018 first preferences would result in a line with a slope of 0.95, which is much less desirable than the 1.02 achieved by the transition matrix, with MAE of 16.3 for 2018 and 19.8 for 2019, also indicating it is less reliable than any of the transition matrices, with the possible exception of the SA matrix.

3.3 Additional Analysis

The available pupil file is from 2016/17, allowing the transition matrix to be used to project first preference applications until 2023. The matrix is applied retrospectively so that the chart gives an indication of the accuracy of the predictions, as well as the predictions themselves. This has been included in a Shiny app for ease of use, as shown in Figure 5. The points show the actual number of first preference applications each year, the line shows the predicted number based on the transition matrix and a dashed line indicates the permitted intake for 2017.

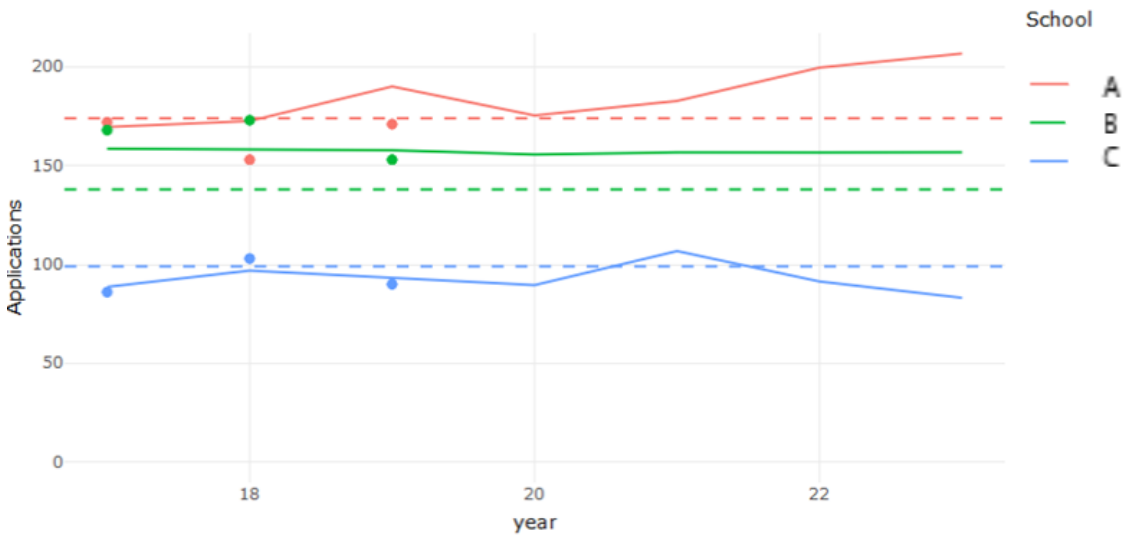

Figure 5

In Figure 5 (taken from the Shiny app), we can see that First Preference Applications to School A appear slightly difficult to predict, with approximately 20 less applications than predicted being received in 2018 and 2019. School B appears to have a very stable population in their feeder primary schools, although the actual number of FPAs has been less stable, whereas School C predictions were within 6 of actual applications in each of the 3 years. The predicted lines do not always go through the points showing the actual FPAs, as not all first preferences can be linked back to a pupil and therefore a primary school.

3.3.1 First and Second Preferences

There are major limitations with the second preference data, in that second preferences are recorded for less than 20% of pupils. There are likely to be a variety of reasons for this, with the EA suggesting that some parents may be wary of recording a second preference as they believe it may harm chances of getting into a first preference school, and also suggesting that even where second preferences are expressed they may not be recorded on the system as those entering details into the system are aware that the first preference school is likely to accept the pupil. This suggests that the data are missing in a non-random way.

Furthermore, looking at top four preferences, there are two instances where preferences for both Boys' and Girls' single-sex schools have been expressed by the same pupil. It is not possible to say how many other mistakes have been made in recording preferences. Of the 68,772 first preferences expressed, just 13,276 second preferences are recorded (19.3%) and 5,640 third preferences (8.2%).

Tables have been produced display, for a selected school:

- second preferences for that school,
- first preferences where that school is second preference and
- reallocations of pupils from that school, using the transition matrix

A Shiny app (see appendix section 3.2) was used to make it easy to compare these three tables, allowing intuitions about particular schools to be tested by knowledgeable users. These individual cases are the most effective way to assess the usefulness of the tables. However, the analysis below attempts to provide an overview without focusing in on individual cases.

There is only one instance where the transition matrix and the second preference list identified the same top 6 affected schools, and only two instances where first and second preference tables agree on the top 6. Table 3 summarises the correspondence between the different tables, with figures in parenthesis indicating the figures calculated using the SOA for the transition matrix.

Table 3

Summary of similarities between top 6 results in scenario planning for each post-primary school

Measure	Transition and 2nd preference	Transition and 1st preference	1st and 2nd preferences	All three tables
Median	3 (3)	3 (3)	3	2 (2)
Mean	3.1 (2.8)	2.9 (2.8)	3.3	2.1 (2.08)
% >= 4 matches	41.1 (37.6)	30.7 (31.6)	40.4	12.4 (11.4)
% >= 3 matches	64.5 (60.6)	60.6 (58.1)	70.8	36.0 (30.7)
% >= 2 matches	84.1 (82.6)	89.8 (87.5)	94.4	69.7 (69.3)

For more than four matches, there is a similar size of overlap between schools identified by the transition matrix and the 2nd preference table, and those identified by looking at the 1st and 2nd preference tables. As we reduce our threshold number of matches between the two tables, the size of the overlap between the transition matrix and 2nd preferences does not keep up with the size of the overlap between 1st and 2nd preferences. This analysis suggests that the transition matrix can be used to identify a real relationship between schools. This relationship may be similar to the relationships that can be identified by looking at 2nd preferences for a school. It is worth noting with the figures above that the transition matrix preserves preference for a secondary or grammar school based on first preference, while in 37% of cases this preference was not preserved between first and second preferences.

4. Discussion

4.1 Successes and Limitations

It has not been possible to quantify the usefulness of the methods used to make predictions about future first preference applications, as it is not possible to say in how many cases the estimates were close enough for practical purposes. The transition matrix is promising in that it out-performed the simplest method of predicting that each school would receive the same number of applications as they did in the previous year, as measured by R^2, MAE and the gradient of the line between predicted and actual. Although 86.75% of pupils were offered a place at their first preference school in 2019[9], predicting only first preferences will not allow for predictions to be made about enrolments, which is crucial to understanding future needs and identifying schools that may become unviable.

The transition matrix is also used to try to understand the links between schools, so that if there are changes to one school, the other schools likely to be impacted can be identified. There are a number of limitations with this approach:

1. It reflects first preferences, but not admissions. Post-primary schools often give priority to pupils from a primary school with a matching religious affiliation as one of their first admissions criteria. In one example, having attended an integrated primary school is the 3rd criterion, after having a sibling enrolled at the school and being an oldest child. In 2016 27% of applications were from pupils at integrated primary schools, but 39% of those admitted that year had attended an integrated primary school. Therefore it reflects where pupils, potentially from a number of schools, would have applied if a particular school was not available.
2. Comparing the outputs with the second preferences for particular schools, it appears that there are categories within the different schools of the same type. For example, the transition matrix tells us that pupils whose first preference is for School A attend primary school with pupils whose first preferences are mainly for School B, but also School C. The second preference data, despite its limitations, clearly indicates that, of those whose first choice is School A, School C is much more popular as a second preference than School B, which is the third most popular second preference.

Despite these limitations, the simplicity of the approach ensures that the tables produced are meaningful, if not precise. The transition matrix can be used to identify schools that share the same feeder primary schools as any school selected, with the estimates of displaced pupils giving a weighting of those primary schools (based on first preferences rather than admissions).

4.2 Future Enquiries

We hope to have subsequent preferences recorded more accurately in the near future. Reliable subsequent preference data would allow for predictions of enrolments, although there would be numerous possible methods to explore for assigning pupils to schools.

The continued collection of data would also allow us to understand how much less reliable estimates of first preference applications become over time: there is clearly a deterioration of the quality of estimates, but not enough years of data to quantify this deterioration. We have seen that the transition matrix performs worst where there are external circumstances impacting on first preference applications. However, it does account for demographic changes and it would be useful to be able to make predictions even with a ceteris paribus assumption.

The scenario planning uses the transition matrix as a simple model of the relationship between feeder primary schools and post-primary schools. However, with reliable subsequent preference data recorded, there would be no need to use the model of transfers between primary and post-primary schools: instead, data on pupils in a school, or who selected a school as first preference could be presented. If a scenario such as closing school X was explored, it would then be possible to state the other preferences (higher or lower) of every pupil in school X, giving an indication of where extra capacity could be used to facilitate a school closure. Likewise, a scenario of expanding a school's capacity could be explored by looking at the schools attended by pupils who would have preferred to attend the school whose capacity is expanded rather than the school to which they were allocated.

Despite the rich potential of second preference data, a number of schools continue to use having been selected as first preference as a criterion for selection, and we expect a number of parents to continue to record only a first preference. Along with these actions that may be attempts to game the system, recorded preferences reflect a perceived realistic preference, with factors such as scores in transfer tests causing actual first preferences not to be recorded.

5. Conclusion

This paper set out to understand how preference data could be used to predict future demand for schools and understand the impact that changes to one school may have on applications to another school. Some promise was shown in predicting future first preferences, although events outside the scope of a model will limit its predictive power over time. However, what is required for assessing the future viability of a school are predictions of enrolments, not merely first preferences. More data on subsequent preferences may allow for predictions of allocations of pupils, and an expansion of the transition matrix to meet the first objective of predicting future demand for schools.

The use of the transition matrix to reallocate pupils was promising, and its outputs are validated both by the fact that the transition matrix can make reasonable predictions of first preference applications to a school, and the correspondence with second preference data. This method provides a weighted summary of the post-primary schools that share the same feeder primary schools as the school selected. This could be further developed to account for the primary schools attended for those actually enrolled at the school, which will not be identical to the primary schools of those who applied for the school. Even without this development, it has been deemed worthwhile to add this information, in the form of a table and diagram (code in Appendix section 3.1, screenshots removed) to other information about schools made available to area planners in the EA.

There are some complex relationships between post-primary schools, even those of the same type. A transition matrix can never account for these relationships (such as sub-types and hierarchies), but if better second preference data becomes available, as expected, future scenario planning will be able to move away from the simple model provided by the transition matrix, to a more factual summary of actual preferences for specific pupils enrolled in schools of interest.

[1] A Super Output Area (SOA) is a statistical geography, dividing Northern Ireland into 890 SOAs

[2] A Small Area (SA) is a statistical geography containing at least 40 households or 100 people. They nest into SOAs and there are 4,537 SAs in Northern Ireland

[3] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[4] Hadley Wickham and Lionel Henry (2017). tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions. R package version 0.7.1. https://CRAN.R-project.org/package=tidyr

[5] Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2018). dplyr: A Grammar of Data Manipulation. R package version 0.7.5. https://CRAN.R-project.org/package=dplyr

[6] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2018). caret: Classification and Regression Training. R package version 6.0-81. https://CRAN.R-project.org/package=caret

[7] Marvin N. Wright, Andreas Ziegler (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, 77(1), 1-17. doi:10.18637/jss.v077.i01

[8] Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

[9] EA website, https://www.eani.org.uk/parents/admissions

Appendix

Table of Contents

1. Functions

1.1 Assess Models

1.1.1 Description

This function returns a dataframe containing the predicted number of first preference applications for each school in the dataset, along with the actual number of first preference applications. It takes as inputs the model for making predictions for individual pupils. It defaults to return actual and predicted applications with the training set, but can also be set to use a test set.

1.1.2 Code

```
assessModel <- function(model, test_x = NULL, test_y = NULL, data = "train"){

 probs <- extractProb(list(model))

 if(data == "train"){
 probsDf <- probs %>%
  select(-obs, -pred, -model, -dataType, -object) %>%
  colSums() %>%
  as.data.frame()

 probsDf$school <- gsub("X.","",rownames(probsDf))
 probsDf <- rename(probsDf, applications = .)

 temp <- table(gsub("X.","",probs$obs)) %>%
  as.data.frame()

 probsDf <- probsDf %>%
  left_join(temp, by = c("school" = "Var1"))

 returnDf <- probsDf
 }

## otherwise try it on test data

if(data == "test" & !is.null(test_x) & !is.null(test_y)){

 test_x <- test_pupils %>%
  select(-FPA)

 test_y <- test_pupils %>%
  select(FPA) %>%
  pull()

 test_y <- as.factor(test_y)

 probsTestDf <- predict(model, newdata = test_x, type="prob") %>%
  colSums() %>%
  as.data.frame()

 probsTestDf$school <- gsub("X.","",rownames(probsTestDf))
 probsTestDf <- rename(probsTestDf, applications = .)

 temp2 <- table(gsub("X.","",test_y)) %>%
  as.data.frame()

 probsTestDf <- probsTestDf %>%
  left_join(temp2, by = c("school" = "Var1"))
```

```
 probsTestDf$Freq[is.na(probsTestDf$Freq)] <- 0
 probsTestDf$correct <- as.factor(ifelse(round(probsTestDf$applications,0)==probsTestDf$Freq,1,0))

 returnDf <- probsTestDf[probsTestDf$Freq != 0 & probsTestDf$school != "NA",]
}

 if(!exists("returnDf")){
  stop("Function could not run. If you specified test, did you supply test_x and test_y?")
 }
 else{
  returnDf
 }
}
```

1.1 CreateMatrixProb

1.2.1 Description

This function returns the proportions of pupils of each gender going from each primary school to each post-primary school. It accepts parameters specifying which year's transferring cohort from the preference file should be used, allows for a list of schools to be excluded from the matrix and also uses a post_primary data frame to provide details of the post-primary schools along with the proportions.

1.2.2 Code

```
createMatrixProb <- function(multiyear, post_primary, simClosed = 1, year = 1){

 sY1PrimarySize <- multiyear %>%
  filter(year == !!year) %>%
  filter(!fpa2 %in% simClosed) %>%
  group_by(DENI.no., Gender) %>%
  summarise(primarySizeY1 = n())

 matrixProb <- multiyear %>%
  filter(year == !!year) %>%
  filter(!fpa2 %in% !!simClosed) %>%
  group_by(DENI.no., Gender, fpa2) %>%
  summarise(Freq = n()) %>%
  left_join(sY1PrimarySize, by = c("DENI.no.","Gender")) %>%
  mutate(percentOfSchool = Freq/primarySizeY1, fpa2 = as.numeric(as.character(fpa2))) %>%
  left_join(post_primary[,c(1,6)], by = c("fpa2" = "denino"))

 matrixProb
}
```

1.1 CreatePredByYear

1.3.1 Description

This function takes the pupil file and, for a specified year, applies the proportions from a probability matrix (as created by createMatrixProb above) to predict the number of applications to each post-primary school.

1.3.2 Code

```
createPredByYear <- function(multiyear, matrixProb, post_primary, year = 1){

  p7Pred <- multiyear %>%
    filter(year == !!year) %>%
    group_by(DENI.no., Gender) %>%
    summarise(yXSize = n()) %>%
    left_join(matrixProb, by = c("DENI.no.", "Gender")) %>%
    mutate(yXPred = percentOfSchool * yXSize)

  p7Pred
}
```

1.1 createPredictionChartDf

1.4.1 Description

This function creates a dataframe in a tidy format, ready to plot the predicted and actual first preference applications to each post-primary school. It takes as inputs (in this order) pupil data, preferences data, attributes data of post-primary schools, a post-code lookup and a string specifying whether primary school code or an area based variable should be used for the transition matrix.

1.4.2 Code

```
source("functions//createMatrixProb.R")
source("functions//createPredSummary.R")

createPredictionChartDf <- function(sixteen, firstPrefs, post_primary, postcodes, variable =
"denino") {
  allowedList <- c("denino","lsoa11","oa11","osward","pcon","nuts","ttwa" )
  if(!variable %in% allowedList){
    stop(paste0("Please choose a variable from the following list, or leave blank to default to denino:",
allowedList))
  }

  multi <- firstPrefs %>%
    filter(pref == 1) %>%
    left_join(sixteen, by = "UPN") %>%
    left_join(postcodes[,c(5,12,22,25,27,36,37)], by = c("postcode" = "pcd")) %>%
    mutate(fpa2 = as.numeric(as.character(school)), DENI.no. = eval(parse(text =variable)), Gender =
gender, year = year.x) %>%
    select(fpa2, DENI.no., Gender, UPN, year)

  pupils16 <- sixteen %>%
    left_join(postcodes, by = c("postcode" = "pcd")) %>%
```

```
  mutate(denino = eval(parse(text = variable)))


 # 2017  matrix
 matrixProb <- createMatrixProb(multi, post_primary, year = 2017)

 # actual fpas for each year, name columns a2017 etc.
 actualFpas <- firstPrefs %>%
  filter(pref == 1) %>%
  group_by(year) %>%
  select(school, year) %>%
  table() %>%
  as.data.frame() %>%
  spread(year,Freq, fill = 0) %>%
  mutate(school = as.numeric(as.character(school))) %>%
  rename(a2017 = `2017`, a2018 = `2018`, a2019 = `2019`)

 # apply matrix to primary school populations, by gender, as per available pupil file
 # Aggregate individual probabilities of going to each school to estimate FPAs
 predictionSummary <- pupils16 %>%
  mutate(year = ifelse(ccyear==7,17,
                  ifelse(ccyear==6,18,
                       ifelse(ccyear==5,19,
                            ifelse(ccyear==4,20,
                                 ifelse(ccyear==3,21,
                                      ifelse(ccyear==2,22,
                                           ifelse(ccyear == 1,23,-1)))))))) %>%
  filter(year != -1) %>%
  count(denino,gender,year) %>%
  as.data.frame() %>%
  inner_join(matrixProb[,c(1,2,3,6)], by = c("denino" = "DENI.no.", "gender" = "Gender")) %>%
  mutate(applications = percentOfSchool * n) %>%
  ungroup() %>%
  select(fpa2, year, applications) %>%
  rename(school = fpa2) %>%
  group_by(school,year) %>%
  summarise(Applications = sum(applications)) %>%
  as.data.frame()

 predictionChart <- predictionSummary %>%
  spread(year, Applications, fill = 0) %>%
  left_join(post_primary[,c(1:2)], by = c("school" = "denino")) %>%
  left_join(actualFpas, by = "school") %>%
  rename(p2017 = `17`, p2018 = `18`, p2019 = `19`)
}
```

2. Scripts

2.1 MajorityPupils

2.1.1 Description

The script below create a subset of the pupil data, containing only similar pupils, who are further divided into 10 categories by gender, and preference for type of school (grammar/secondary and religious affiliation). Each of these ten datasets are then analysed, by Small Area and Super Output Area, to see how many different schools of the same type are chosen as first preferences. This is visualised both as a map and as a chart.

2.1.2 Code

```
library(tidyverse)
library(leaflet)

#### import data ####

prefs <- read.csv("..//p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]
prefs <- prefs[!duplicated(prefs[,2:3]),] # Remove duplicates of Unique Pupil Number and preference number,
#e.g. remove cases where one pupil has recorded two first preferences
prefs$school <- as.numeric(as.character(prefs$school)) # Not treated as factor here to allow joining

schools <- read.csv("..//post_primary_school_attributes.csv")

pupils16 <- read.csv("..//2016-17 Pupils.csv", stringsAsFactors = F)

postcodes <- read.csv("..//postcodes.csv")

pupils <- pupils16 %>%
  inner_join(postcodes[,c(5,25,36,37)], by = c("postcode" = "pcd"))

soa <- geojsonio::geojson_read("..\\Super Output Areas.json",
                               what = "sp")
sa <- geojsonio::geojson_read("..\\Small Areas.json",
                              what = "sp")

#### join data together ####

# take first preference data and combine with pupil attributes
firstPref <- prefs %>%
  filter(pref == 1)

pupils <- pupils %>%
  inner_join(firstPref, by = "UPN") %>%
  rename(schoolyear = year.x, year = year.y)

pupils <- pupils %>% droplevels()
```

```
#### SOA ####

male_1_grammar_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11)

male_1_secondary_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11)

male_2_grammar_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11)

male_2_secondary_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11)

female_1_grammar_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11)

female_1_secondary_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
```

```
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11)

female_2_grammar_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11)

female_2_secondary_majority <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11)

categories <- c("male_1_grammar_majority", "male_2_grammar_majority",
"male_1_secondary_majority", "male_2_secondary_majority",
            "female_1_grammar_majority", "female_2_grammar_majority",
"female_1_secondary_majority", "female_2_secondary_majority")

for(i in 1:length(categories)){
  print(categories[i])

  matches <- eval(parse(text = categories[i])) %>%
    ungroup() %>%
    group_by(lsoa11) %>%
    summarise(matchCategory = n())

  tbl <- eval(parse(text = categories[i]))  %>%
    group_by(lsoa11, school) %>%
    summarise(count = n()) %>%
    arrange(lsoa11, school) %>%
    as.data.frame() %>%
    ungroup %>%
    group_by(lsoa11) %>%
    summarise(destinations = n()) %>%
    arrange(desc(destinations)) %>%
    left_join(matches, by = c("lsoa11"))

  df <- as.data.frame(tbl)
  maxmin <- df %>%
    group_by(matchCategory) %>%
    summarise(max = max(destinations), min = min(destinations), mean = mean(destinations)) %>%
    as.data.frame()
```

```
  print(ggplot(maxmin) +
        labs(x = "number of 'identical' pupils in SOA", y = "number of diffrent first preference schools",
             title = "Male pupils who selected a Non-denominational Grammar school as first
preference",
             subtitle = "Showing the number of first preference schools compared to the number of
transferring pupils in an SOA") +
        geom_line(aes(x=matchCategory, y=max),color="red") +
        geom_line(aes(x=matchCategory, y=min),color="blue") +
        geom_line(aes(x=matchCategory, y=mean),color="black") +
        theme_minimal() +
        scale_x_continuous(minor_breaks = seq(0 , 50, 2), breaks = seq(0, 50, 4)) +
        annotate("text", x = 5, y = 4, hjust = -0.2, label = "max", col = "red") +
        annotate("text", x = 5, y = 1.9, hjust = -0.2, label = "mean", col = "black") +
        annotate("text", x = 5, y = 1.1, hjust = -0.2, label = "min", col = "blue"))
}

ggplot(maxmin) +
  labs(x = "number of 'identical' pupils in SOA", y = "number of diffrent first preference schools",
       title = categories[i]) +
  geom_line(aes(x=matchCategory, y=max),color="red") +
  geom_line(aes(x=matchCategory, y=min),color="blue") +
  geom_line(aes(x=matchCategory, y=mean),color="black") +
  theme_minimal() +
  scale_x_continuous(minor_breaks = seq(0 , 50, 2), breaks = seq(0, 50, 4)) +
  annotate("text", x = 5, y = 4, hjust = -0.2, label = "max", col = "red") +
  annotate("text", x = 5, y = 1.9, hjust = -0.2, label = "mean", col = "black") +
  annotate("text", x = 5, y = 1.1, hjust = -0.2, label = "min", col = "blue")

#### SA ####

male_1_grammar_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, oa11)

male_1_secondary_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, oa11)

male_2_grammar_majority_sa <- pupils %>%
```

```
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, oa11)

male_2_secondary_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, oa11)

female_1_grammar_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, oa11)

female_1_secondary_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, oa11)

female_2_grammar_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, oa11)

female_2_secondary_majority_sa <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(oa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, oa11)
```

```
categories_sa <- c("male_1_grammar_majority_sa", "male_2_grammar_majority_sa",
"male_1_secondary_majority_sa", "male_2_secondary_majority_sa",
                "female_1_grammar_majority_sa", "female_2_grammar_majority_sa",
"female_1_secondary_majority_sa", "female_2_secondary_majority_sa")

for(i in 1:length(categories_sa)){
  print(categories_sa[i])

  matches <- eval(parse(text = categories_sa[i])) %>%
    ungroup() %>%
    group_by(oa11) %>%
    summarise(matchCategory = n())

  tbl <- eval(parse(text = categories_sa[i]))  %>%
    group_by(oa11, school) %>%
    summarise(count = n()) %>%
    arrange(oa11, school) %>%
    as.data.frame() %>%
    ungroup %>%
    group_by(oa11) %>%
    summarise(destinations = n()) %>%
    arrange(desc(destinations)) %>%
    left_join(matches, by = c("oa11"))

  df <- as.data.frame(tbl)
  maxmin <- df %>%
    group_by(matchCategory) %>%
    summarise(max = max(destinations), min = min(destinations), mean = mean(destinations)) %>%
    as.data.frame()

  print(ggplot(maxmin) +
        labs(title = categories_sa[i]) +
        geom_line(aes(x=matchCategory, y=max),color="red") +
        geom_line(aes(x=matchCategory, y=min),color="blue") +
        geom_line(aes(x=matchCategory, y=mean),color="black"))
}


## SOAs with highest number of schools chosen:

saCentres <- as.data.frame(cbind(as.character(sa$id), as.character(sa$name),
as.numeric(sa$center1), as.numeric(sa$center2)))
names(saCentres) <- c("saCode","saName","lng","lat")

dispersedSoa <- as.data.frame(c("95JJ12W1"))
names(dispersedSoa) <- "lsoa11"

dispersedPupils <- dispersedSoa %>%
```

```
 left_join(pupils, by = "lsoa11") %>%
 left_join(saCentres, by = c("oa11" = "saCode")) %>%
 left_join(prefs, by = "UPN") %>%
 select(lsoa11, lat,lng, school.x) %>%
 rename(pupil.lat = lat, pupil.lng = lng) %>%
 unique() %>%
 mutate(FPA = as.numeric(gsub("X.","",x=.$school.x))) %>%
 left_join(schools[,c(1,2,3)], by = c("FPA" = "denino")) %>%
 left_join(postcodes[,c(5,46,47)], by = c("school_postcode" = "pcd")) %>%
 rename(school.lat = lat, school.lng = long)

m <- leaflet(soa, options = leafletOptions(zoomSnap = 0.01)) %>%
 addProviderTiles(providers$Stamen.TonerLite) %>%
 addPolygons(stroke = TRUE, smoothFactor = 0.3, fillOpacity = 0, weight = 1)

# Create a colour palette
pal <-
colorFactor(c('#1b9e77','#d95f02','#7570b3','#e7298a','#66a61e','#e6ab02','#a6761d','#666666'),
            domain = unique(pupils$pref.schoolname))

# Add pupils to the map
for(i in 1:nrow(dispersedPupils)){
 m <- addPolylines(m,
lat=c(as.numeric(as.character(dispersedPupils[i,]$pupil.lat)),dispersedPupils[i,]$school.lat),
            lng = c(as.numeric(as.character(dispersedPupils[i,]$pupil.lng)),
dispersedPupils[i,]$school.lng),
            label = dispersedPupils[i,]$schoolname,
            color = pal(dispersedPupils[i,]$schoolname),
            fillOpacity = 0.1,
            weight = 2
 )
}

m

#### Other visualisation of dispersion ####

soaList <- list()

for(i in 1:length(categories)){
 print(categories[i])

 matches <- eval(parse(text = categories[i])) %>%
  ungroup() %>%
  group_by(lsoa11) %>%
  summarise(matchCategory = n())

 tbl <- eval(parse(text = categories[i]))  %>%
```

```
   group_by(lsoa11, school) %>%
   summarise(count = n()) %>%
   arrange(lsoa11, school) %>%
   as.data.frame() %>%
   ungroup %>%
   group_by(lsoa11) %>%
   summarise(destinations = n()) %>%
   arrange(desc(destinations)) %>%
   left_join(matches, by = c("lsoa11"))

  df <- as.data.frame(tbl)

  soaList[[i]] <- df

}

allDfs <- bind_rows(soaList)
df <- allDfs %>%
  group_by(lsoa11) %>%
  summarise(destinations = max(destinations))

ids <- as.data.frame(soa@data$id)
destinations <- ids %>%
  left_join(df, by = c("soa@data$id" = "lsoa11"))

soa$destinations <- destinations$destinations

factpal <-
colorFactor(c('#f7fbff','#deebf7','#c6dbef','#9ecae1','#6baed6','#4292c6','#2171b5','#084594'),
soa$destinations)

m <- leaflet(soa) %>%
  addProviderTiles(providers$Stamen.TonerLite) %>%
  addPolygons(stroke = TRUE, smoothFactor = 0.3, fillOpacity = 0.5, weight = 1,
              color = ~factpal(destinations))

print(m)


### min destinations ####
dfMin <- allDfs %>%
  group_by(lsoa11) %>%
  summarise(destinations = min(destinations))

ids <- as.data.frame(soa@data$id)
destinationsMin <- ids %>%
  left_join(dfMin, by = c("soa@data$id" = "lsoa11"))
```

```
soa$destinationsMin <- destinationsMin$destinations

factpal <-
colorFactor(c('#f7fbff','#deebf7','#c6dbef','#9ecae1','#6baed6','#4292c6','#2171b5','#084594'),
soa$destinationsMin)

m <- leaflet(soa) %>%
  addProviderTiles(providers$Stamen.TonerLite) %>%
  addPolygons(stroke = TRUE, smoothFactor = 0.3, fillOpacity = 0.5, weight = 1,
              color = ~factpal(destinationsMin))

print(m)

#### SOA plus primary school ####

male_1_grammar_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11, denino)

male_1_secondary_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11, denino)

male_2_grammar_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11, denino)

male_2_secondary_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 0) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11, denino)

female_1_grammar_majority_ps <- pupils %>%
```

```
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11, denino)

female_1_secondary_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 1, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11, denino)

female_2_grammar_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Grammar") %>%
  select(school, lsoa11, denino)

female_2_secondary_majority_ps <- pupils %>%
  filter(dayboard == 0, specialu == 0 , care == 0, freemeal == 0, stage == 0, newcomer == 0, imu == 0,
relig == 2, gender == 1) %>%
  arrange(lsoa11) %>%
  left_join(schools[,c(1,2,3,6,7)], by = c("school" = "denino")) %>%
  filter(Gram_Sec_label == "Secondary") %>%
  select(school, lsoa11, denino)

categories <- c("male_1_grammar_majority_ps", "male_2_grammar_majority_ps",
"male_1_secondary_majority_ps", "male_2_secondary_majority_ps",
            "female_1_grammar_majority_ps", "female_2_grammar_majority_ps",
"female_1_secondary_majority_ps", "female_2_secondary_majority_ps")

soaDeninoList <- list()

for(i in 1:length(categories)){
  print(categories[i])

  matches <- eval(parse(text = categories[i])) %>%
    ungroup() %>%
    group_by(lsoa11, denino) %>%
    summarise(matchCategory = n()) %>%
    group_by(lsoa11) %>%
    summarise(matchCategory = max(matchCategory))
```

```
  tbl <- eval(parse(text = categories[i]))  %>%
    group_by(lsoa11, school, denino) %>%
    summarise(count = n()) %>%
    arrange(lsoa11, school) %>%
    as.data.frame() %>%
    ungroup %>%
    group_by(lsoa11, denino) %>%
    summarise(destinations = n()) %>%
    arrange(desc(destinations)) %>%
    left_join(matches, by = c("lsoa11"))

  df <- as.data.frame(tbl)
  maxmin <- df %>%
    group_by(matchCategory) %>%
    summarise(max = max(destinations), min = min(destinations), mean = mean(destinations)) %>%
    as.data.frame()

  print(ggplot(maxmin) +
        labs(x = "number of 'identical' pupils in SOA", y = "number of different first preference
schools",
             title = "Male pupils who selected a Non-denominational Grammar school as first
preference",
             subtitle = "Showing the number of first preference schools compared to the number of
transferring pupils in an SOA") +
        geom_line(aes(x=matchCategory, y=max),color="red") +
        geom_line(aes(x=matchCategory, y=min),color="blue") +
        geom_line(aes(x=matchCategory, y=mean),color="black") +
        theme_minimal() +
        scale_x_continuous(minor_breaks = seq(0 , 50, 2), breaks = seq(0, 50, 4)) +
        annotate("text", x = 5, y = 4, hjust = -0.2, label = "max", col = "red") +
        annotate("text", x = 5, y = 1.9, hjust = -0.2, label = "mean", col = "black") +
        annotate("text", x = 5, y = 1.1, hjust = -0.2, label = "min", col = "blue"))

  df <- as.data.frame(tbl)

  soaDeninoList[[i]] <- df
}

allDfs <- bind_rows(soaDeninoList)
df <- allDfs %>%
  group_by(lsoa11) %>%
  summarise(destinations = max(destinations))

ids <- as.data.frame(soa@data$id)
destinations <- ids %>%
  left_join(df, by = c("soa@data$id" = "lsoa11"))

soa$destinations <- destinations$destinations
```

```
factpal <-
colorFactor(c('#f7fbff','#deebf7','#c6dbef','#9ecae1','#6baed6','#4292c6','#2171b5','#084594'),
soa$destinations)

m <- leaflet(soa) %>%
  addProviderTiles(providers$Stamen.TonerLite) %>%
  addPolygons(stroke = TRUE, smoothFactor = 0.3, fillOpacity = 0.5, weight = 1,
              color = ~factpal(destinations)) %>%
  addLegend(position = 'topleft',
            colors = ~factpal(c(1,2,3,4,5,6)),
            labels = ~c(1,2,3,4,5,6))

print(m)
```

2.2 randomForest

2.2.1 Description

This script prepares data for and creates a random forest model. It then uses functions described above to measure the accuracy of the predictive model.

2.2.2 Code

```
library(tidyverse)
library(caret)
library(plotly)

setwd("schools//Report")

#### Import data ####

prefs <- read.csv("..//p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]
prefs <- prefs[!duplicated(prefs[,2:3]),] # Remove duplicates of Unique Pupil Number and preference number,
#e.g. remove cases where one pupil has recorded two first preferences
prefs$school <- as.numeric(as.character(prefs$school)) # Not treated as factor here to allow joining

schools <- read.csv("..//post_primary_school_attributes.csv")

pupils16 <- read.csv("..//2016-17 Pupils.csv", stringsAsFactors = F)

postcodes <- read.csv("..//postcodes.csv")

pupils <- pupils16 %>%
  inner_join(postcodes[,c(5,25,36,37)], by = c("postcode" = "pcd"))

#### Prepare data for modelling ####

# take first preference data and combine with pupil attributes
```

```
firstPref <- prefs %>%
  filter(pref == 1) %>%
  rename(FPA = school) %>%
  mutate(FPA = paste("X",FPA,sep = "."))

pupils <- pupils %>%
  inner_join(firstPref, by = "UPN") %>%
  rename(schoolyear = year.x, year = year.y)

pupils_orig <- pupils %>% droplevels()

# select variables
pupils <- pupils_orig[,c(4,6,7,8,10,11,12,13,14,16,18,19,28,29,30,31,33)]

pupils$denino <- as.factor(pupils$denino)
pupils$relig <- as.factor(pupils$relig)

# create training and test sets
train_pupils <- pupils[pupils$year == 2017,]
test_pupils <- pupils[pupils$year == 2019,]

# separate response variable
pupils_x <- train_pupils %>%
  select(-FPA)

test_x <- test_pupils %>%
  select(-FPA)

pupils_y <- train_pupils %>%
  select(FPA) %>%
  pull()

test_y <- test_pupils %>%
  select(FPA) %>%
  pull()

pupils_y <- as.factor(pupils_y)

myFolds <- createFolds(pupils_y, k=10)

# Create trainControl

ctrl <- trainControl(
  method = "cv",
  number = 10,
  index = myFolds,
  summaryFunction = multiClassSummary,
  classProbs = TRUE,
```

```
  verboseIter = FALSE,
  savePredictions = TRUE
)

# Train a model

model_ranger <- train(
  x = pupils_x,
  y = pupils_y,
  method = "ranger",
  trControl = ctrl,
  importance = "impurity",
  preProcess = c("zv", "center", "scale"),
  tuneGrid = expand.grid(
    .mtry = c(7,8,9),
    .splitrule = "gini",
    .min.node.size = 5
  )
)

saveRDS(model_ranger, "models//ranger.RDS")

# Review model on training and test data

#### end ####

# Review models on training and test data
source("functions//assessModel.R")

model_ranger <- readRDS("models//ranger.RDS")

trainOutcome <- assessModel(model_ranger, data="train")
testOutcome <- assessModel(model_ranger, test_x, test_y, data="test")

cor(trainOutcome$Freq,trainOutcome$applications)^2
cor(testOutcome$Freq,testOutcome$applications)^2

t <- ggplot(testOutcome,aes(x=Freq,y=applications, text=school, col=correct)) +
  geom_point(size=3) +
  labs(title = "2017 model tested on 2018 data: ranger", x= "Actual FPAs", y = "Predicted FPAs") +
  theme_minimal()
ggplotly(t)


tree <- treeInfo(model_ranger$finalModel, tree = 5)
plot(tree$splitval)
```

2.3 nnet

2.3.1 Description

This script prepares data for and creates a neural network model. It then uses functions described above to measure the accuracy of the predictive model.

2.3.2 Code

```
library(tidyverse)
library(caret)

setwd("Report")

#### Import data ####

prefs <- read.csv("..//p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]
prefs <- prefs[!duplicated(prefs[,2:3]),] # Remove duplicates of Unique Pupil Number and preference number,
 #e.g. remove cases where one pupil has recorded two first preferences
prefs$school <- as.numeric(as.character(prefs$school)) # Not treated as factor here to allow joining

schools <- read.csv("..//post_primary_school_attributes.csv")

pupils16 <- read.csv("..//2016-17 Pupils.csv", stringsAsFactors = F)

postcodes <- read.csv("..//postcodes.csv")

pupils <- pupils16 %>%
 inner_join(postcodes[,c(5,25,36,37)], by = c("postcode" = "pcd"))

#### Prepare data for modelling ####

# take first preference data and combine with pupil attributes
firstPref <- prefs %>%
 filter(pref == 1) %>%
 rename(FPA = school) %>%
 mutate(FPA = paste("X",FPA,sep = "."))

pupils <- pupils %>%
 inner_join(firstPref, by = "UPN") %>%
 rename(schoolyear = year.x, year = year.y)

pupils_orig <- pupils %>% droplevels()

# select variables
pupils <- pupils_orig[,c(4,6,7,8,10,11,12,13,14,16,18,19,28,29,30,31,33)]

pupils$denino <- as.factor(pupils$denino)
pupils$relig <- as.factor(pupils$relig)
```

```
# create training and test sets
train_pupils <- pupils[pupils$year == 2017,]
test_pupils <- pupils[pupils$year == 2019,]

# separate response variable
pupils_x <- train_pupils %>%
  select(-FPA)

test_x <- test_pupils %>%
  select(-FPA)

pupils_y <- train_pupils %>%
  select(FPA) %>%
  pull()

test_y <- test_pupils %>%
  select(FPA) %>%
  pull()

pupils_y <- as.factor(pupils_y)

#### Modelling ####

set.seed(1)

myFolds <- createFolds(pupils_y, k=5)

# Create trainControl

ctrl <- trainControl(
  method = "cv",
  number = 5,
  index = myFolds,
  summaryFunction = multiClassSummary,
  classProbs = TRUE,
  verboseIter = FALSE,
  savePredictions = TRUE
)
set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 100
```

```
)

saveRDS(model_nnet, "models//nnet-cv100Iter.RDS")

# Adjust train control

ctrl <- trainControl(
  method = "none",
  summaryFunction = multiClassSummary,
  classProbs = TRUE,
  verboseIter = FALSE,
  savePredictions = TRUE
)

# run the other models
set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 200
)
saveRDS(model_nnet, "models//nnet-noCv200Iter.RDS")

set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 1000
)
saveRDS(model_nnet, "models//nnet-noCv1000Iter.RDS")

set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 5000
```

```
)
saveRDS(model_nnet, "models//nnet-noCv5000Iter.RDS")

#### end ####

#### Remove potentially redundant variables ####
# select variables
pupils <- pupils_orig[,c(4,6,7,8,10,11,12,13,14,16,18,19,28,31,33)]

pupils$denino <- as.factor(pupils$denino)
pupils$relig <- as.factor(pupils$relig)

# create training and test sets
train_pupils <- pupils[pupils$year == 2017,]
test_pupils <- pupils[pupils$year == 2019,]

# separate response variable
pupils_x <- train_pupils %>%
  select(-FPA)

test_x <- test_pupils %>%
  select(-FPA)

pupils_y <- train_pupils %>%
  select(FPA) %>%
  pull()

test_y <- test_pupils %>%
  select(FPA) %>%
  pull()

pupils_y <- as.factor(pupils_y)

#### Modelling ####

# create train control

ctrl <- trainControl(
  method = "none",
  summaryFunction = multiClassSummary,
  classProbs = TRUE,
  verboseIter = FALSE,
  savePredictions = TRUE
)

# run the models
set.seed(1)
model_nnet <- train(
```

```
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 200
)
saveRDS(model_nnet, "models//nnet2-noCv200Iter.RDS")

set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 1000
)
saveRDS(model_nnet, "models//nnet2-noCv1000Iter.RDS")

set.seed(1)
model_nnet <- train(
  x = pupils_x,
  y = pupils_y,
  method = "nnet",
  trControl = ctrl,
  preProcess = c("zv", "center", "scale"),
  MaxNWts = 10000,
  maxit = 5000
)
saveRDS(model_nnet, "models//nnet2-noCv5000Iter.RDS")

#### end ####

# Review models on training and test data
source("functions//assessModel.R")

model_nnet <- readRDS("models//nnet-noCv1000Iter.RDS")

trainOutcome <- assessModel(model_nnet, data="train")
testOutcome <- assessModel(model_nnet, test_x, test_y, data="test")

cor(trainOutcome$Freq,trainOutcome$applications)^2
cor(testOutcome$Freq,testOutcome$applications)^2

t <- ggplot(testOutcome,aes(x=Freq,y=applications, text=school, col=correct)) +
```

```
  geom_point(size=3) +
  labs(title = "2017 model tested on 2018 data: nnet with 1000 iterations", x= "Actual FPAs", y =
"Predicted FPAs") +
  theme_minimal()
ggplotly(t)


train.predictions <- predict(model_nnet, pupils_x, type="raw")

confusionMatrix(train.predictions, pupils_y)
```

2.4 transitionMatrixVariants

2.4.1 Description

This script looks at the performance of the transition matrix for 2017 when applied to 2018 and 2019. It produces charts and R^2 values and can be adapted to use primary school or any of the location variables available in function createPredictionChartDf.

2.4.1 Code

```
library(tidyverse)
library(plotly)

#### Read in Data (2017-19) ####
sixteen <- read.csv("..//2016-17 pupils.csv")
firstPrefs <- read.csv("..//p7Preferences 2017-19.csv")
post_primary <- read.csv("..//post_primary_school_attributes.csv")
postcodes <- read.csv("..//postcodes.csv")

#### access functions ####

source("functions//createPredictionChartDf.R")

predictionChartDf <- createPredictionChartDf(sixteen, firstPrefs, post_primary, postcodes, "lsoa11")

#### 2017-19 based on 2017 matrix for SOA ####
# Get valid first preferences, replace DENI.no. with SOA (name stays the same to avoid changing any
other code)

# Removing unrecognised schools and those whose intake was 0 in 2018
predictions18 <- predictionChartDf %>%
  filter(!is.na(school), !is.na(a2018), a2018 != 0)

# Removing unrecognised schools and those whose intake was 0 in 2018
predictions19 <- predictionChartDf %>%
  filter(!is.na(school), !is.na(a2019), a2019 != 0)

g <- ggplot(predictions18) +
  geom_point(aes(x=a2018, y = p2018, text = schoolname), color="purple", size = 3) +
  geom_point(aes(x=a2019, y = p2019, text = schoolname), color="darkgreen", size = 3) +
```

```
 geom_abline(slope = 1, color = "blue") +
 labs(title="2017 matrix applied to 2018 and 2019", x = "Actual FPA", y = "Predicted FPAs") +
 annotate("text", x=360, y=435, label = "2018", col = "purple") +
 annotate("text", x=300, y=290, label = "2019", col = "darkgreen") +
 theme_minimal()

#ggplotly(g)

# Check r^2 values, equation of line and MAE
cor(predictions18$p2018, predictions18$a2018 )^2
cor(predictions19$p2019, predictions19$a2019 )^2

fit <- lm(p2018 ~ a2018, data=predictions18)

summary(fit)

# MAE
mean(abs(predictions18$p2018 - predictions18$a2018))
mean(abs(predictions19$p2019 - predictions19$a2019))

# For comparison, see how just guessing 2017's FPAs would repeat
cor(predictions18$a2017, predictions18$a2018 )^2
cor(predictions19$a2017, predictions19$a2019 )^2
```

2.5 tableIntersections

2.5.1 Description

This script runs through all three scenario planning functions for all of the post-primary schools in the dataset. It then compares the intersection between the lists created by each method, taking the top six other post-primary schools identified, where at least 4 schools have been identified by each method.

2.5.2 Code

```
library(tidyverse)
library(plotly)

post_primary <- read.csv("..//post_primary_school_attributes.csv", stringsAsFactors = F)
post_primary$schoolname <- as.factor(post_primary$schoolname)
pupils16 <- read.csv("..//2016-17 Pupils.csv", stringsAsFactors = F)

# deal with the Strandtown curiosity
pupils16[pupils16$denino %in% c(1016242,1010304,1010012),]$denino <- 1010252

# Import prepared data
## 2017
matrixProb17 <- read.csv("..//schoolClosures//matrixProb2017.csv")
p7Pred17 <- read.csv("..//schoolClosures//p7Pred2017.csv")

soaMatrixProb17 <- read.csv("..//schoolClosures//soaMatrixProb2017.csv")
```

```
soaP7Pred17 <- read.csv("..//schoolClosures//soaP7Pred2017.csv")

##2018
matrixProb18 <- read.csv("..//schoolClosures//matrixProb2018.csv")
p7Pred18 <- read.csv("..//schoolClosures//p7Pred2018.csv")

soaMatrixProb18 <- read.csv("..//schoolClosures//soaMatrixProb2018.csv")
soaP7Pred18 <- read.csv("..//schoolClosures//soaP7Pred2018.csv")

## 2019
matrixProb19 <- read.csv("..//schoolClosures//matrixProb2019.csv")
p7Pred19 <- read.csv("..//schoolClosures//p7Pred2019.csv")

soaMatrixProb19 <- read.csv("..//schoolClosures//soaMatrixProb2019.csv")
soaP7Pred19 <- read.csv("..//schoolClosures//soaP7Pred2019.csv")

#prefs <- read.csv("..\\AllP7Preferences.csv", header = FALSE)
#prefs <- rename(prefs,school = V1, pref = V2, UPN = V3, year = V4)
prefs <- read.csv("..\\p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]

stated2Prefs <- prefs %>%
 filter(pref %in% c(1,2)) %>%
 select(-year) %>%
 group_by(UPN) %>%
 summarise(fpa = school[which.min(pref)],secondPref = n() - 1) %>%
 ungroup() %>%
 select(fpa, secondPref) %>%
 mutate(fpa = as.numeric(as.character(fpa))) %>%
 group_by(fpa) %>%
 summarise(stated1 = n(), stated2 = sum(secondPref)) %>%
 as.data.frame()

schoolList <- matrixProb19 %>%
 inner_join(post_primary[,1:2],by = c("fpa2" = "denino")) %>%
 select(fpa2,schoolname) %>%
 unique()

tableIntersections <- function(simClosed, n){

 # transition matrix
 size <- p7Pred19 %>%
    filter(fpa2 %in% simClosed) %>%
    select(yXPred) %>%
    sum(na.rm=T)
  #print(paste("size:",size))
```

```
   #### Subjunctive: remove pupils who went to removed school from class size, to uprate everyone
else ####

   # Make sure no pupils are allocated to removed schools
   adjustedProbMatrix <- matrixProb19 %>%
     mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed),0,percentOfSchool))

   reallocationList <- list()

   # For each of the removed schools, get the list of feeders and the number coming from each
feeder,
   # and the school type (Grammar/Secondary)
   for(i in 1:length(simClosed)){
     feeders <- matrixProb19 %>%
       filter(fpa2 == simClosed[i]) %>%
       ungroup() %>%
       select(DENI.no.) %>%
       unique() %>%
       pull()

     numberPerFeeder <- p7Pred19 %>%
       filter(fpa2 == simClosed[i]) %>%
       select(DENI.no., Gender,  yXPred)

     gramOrSec <- post_primary %>%
       filter(denino == simClosed[i]) %>%
       select(Gram_Sec_label) %>%
       pull()

     # Work out how much of the school's allocation went to a particular type of school
     percentAllocatedPerPrimary <- adjustedProbMatrix %>%
       filter(DENI.no. %in% feeders, Gram_Sec_label == gramOrSec) %>%
       group_by(DENI.no., Gender) %>%
       summarise(percentAllocated = sum(percentOfSchool)) %>%
       filter(percentAllocated != 0) # remove cases where only one school of a particular type chosen
from the PS

     # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
     # for a particular type of school add up to 100%, and use these %s to make a prediction based on
how many
     # pupils are "sent back" to each primary school to be reallocated
     reallocation <- adjustedProbMatrix %>%
       filter(Gram_Sec_label == gramOrSec) %>%
       inner_join(percentAllocatedPerPrimary, by = c("DENI.no.", "Gender")) %>%
       inner_join(numberPerFeeder, by = c("DENI.no.", "Gender")) %>%
       mutate(revisedPercent = percentOfSchool / percentAllocated) %>%
       mutate(reallocation = yXPred * revisedPercent) %>%
```

```
      group_by(fpa2) %>%
      summarise(reallocationSum = sum(reallocation)) %>%
      arrange(desc(reallocationSum))

    reallocationList[[i]] <- reallocation
   }

   # combine reallocation lists for each school in the list and display results
   reallocations <- bind_rows(reallocationList)
   reallocationSummary <- reallocations %>%
    group_by(fpa2) %>%
    summarise(displacedFpas = round(sum(reallocationSum),1)) %>%
    left_join(post_primary[,c(1,2)], by = c("fpa2" = "denino")) %>%
    arrange(desc(displacedFpas)) %>%
    as.data.frame()

   reallocationSummary$displacedFpas <- round(reallocationSummary$displacedFpas *
(size/sum(reallocationSummary$displacedFpas, na.rm=T)),1)

topNTransitionMatrix <- reallocationSummary %>%
 head(n)

 count <- stated2Prefs %>%
  filter(fpa %in% simClosed) %>%
  select(stated2)

# 2nd pref list
 pref2List <- list()

 for(i in 1:length(simClosed)){
  upns <- prefs[prefs$school == simClosed[i] & prefs$pref == 1 & nchar(as.character(prefs$UPN)) >
5,3]
   #upns <- prefs[prefs$school %in% c(1420028) & prefs$pref == 1,3]

   secondPrefs <- prefs %>%
    filter(UPN %in% upns, pref == 2) %>%
    select(school) %>%
    table() %>%
    as.data.frame() %>%
    filter(!. %in% simClosed) %>%
    mutate(fpa = as.numeric(as.character(.))) %>%
    select(fpa, Freq) %>%
    filter(Freq != 0) %>%
    left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
    arrange(desc(Freq))

   #print(secondPrefs$Freq)
```

```
  pref2List[[i]] <- secondPrefs

 }
 all2Prefs <- bind_rows(pref2List)

 topNSecondPrefs <- all2Prefs %>%
  arrange(desc(Freq)) %>%
  head(n)

 # 1st PrefList
 pref1List <- list()

 for(i in 1:length(simClosed)){

  upns <- prefs[prefs$school %in% simClosed[i] & prefs$pref == 2 & nchar(as.character(prefs$UPN))
> 5,3]
  #upns <- prefs[prefs$school %in% c(2420054) & prefs$pref == 2,3]

  firstPrefs <- prefs %>%
   filter(UPN %in% upns, pref == 1) %>%
   inner_join(pupils16[,c(2,11)], by = "UPN") %>%
   select(school) %>%
   table() %>%
   as.data.frame() %>%
   filter(!. %in% simClosed) %>%
   mutate(fpa = as.numeric(as.character(.))) %>%
   select(fpa, Freq) %>%
   filter(Freq != 0) %>%
   left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
   arrange(desc(Freq)) %>%
   left_join(stated2Prefs, by = c("fpa" = "fpa")) %>%
   filter(!is.na(fpa)) %>%
   arrange(desc(Freq))

  pref1List[[i]] <- firstPrefs

 }

 topNFirstPrefs <- bind_rows(pref1List) %>%
  group_by(fpa, schoolname, stated2) %>%
  summarise(Freq = sum(Freq)) %>%
  as.data.frame() %>%
  select(fpa, Freq, stated2, schoolname) %>%
  rename(base = stated2) %>%
  arrange(desc(Freq)) %>%
  filter(!is.na(fpa)) %>%
  head(n)
```

```
  c2 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNSecondPrefs) > 4,
           length(intersect(topNFirstPrefs$fpa,topNSecondPrefs$fpa)),NA)
  c3 <- ifelse(nrow(topNTransitionMatrix) > 4 & nrow(topNSecondPrefs) > 4,
           length(intersect(topNTransitionMatrix$fpa2,topNSecondPrefs$fpa)),NA)
  c4 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNTransitionMatrix) > 4,
           length(intersect(topNTransitionMatrix$fpa2,topNFirstPrefs$fpa)),NA)
  c5 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNTransitionMatrix) > 4 & nrow(topNSecondPrefs)
>4,

length(intersect(topNFirstPrefs$fpa,intersect(topNSecondPrefs$fpa,topNTransitionMatrix$fpa2))),
NA)


  row <- c(simClosed, c2, c3, c4, c5)
  cat(".")
  row

#  print(length(intersect(topNFirstPref$fpa,topNSecondPrefs$fpa)))
#  print(paste0("For top ",n," results, first and second preferences have
",length(intersect(topNFirstPrefs$fpa,topNSecondPrefs$fpa))," in common"))
#  print(paste0("For top ",n," results, transition matrix and second preferences have
",length(intersect(topNTransitionMatrix$fpa2,topNSecondPrefs$fpa))," in common"))
#  print(paste0("For top ",n," results, transition matrix and first preferences have
",length(intersect(topNTransitionMatrix$fpa2,topNFirstPrefs$fpa))," in common"))

}

in_common <- list()
for(j in 1:length(schoolList$fpa2)){
  row <- tableIntersections(schoolList[j,]$fpa2,6)
  names(row) <- c("fpa","first.and.second.prefs", "transition.and.second", "transition.and.first",
"all.three")
  in_common[[j]] <- row %>% as.data.frame() %>% t()
}

commonSummary <- do.call(rbind.data.frame, in_common)

hist(commonSummary$transition.and.second)
hist(commonSummary$transition.and.first)
hist(commonSummary$first.and.second.prefs)
hist(commonSummary$all.three)

mean(commonSummary$transition.and.second, na.rm=T)
mean(commonSummary$transition.and.first, na.rm=T)
mean(commonSummary$first.and.second.prefs, na.rm=T)
mean(commonSummary$all.three, na.rm=T)

median(commonSummary$transition.and.second, na.rm=T)
```

```
median(commonSummary$transition.and.first, na.rm=T)
median(commonSummary$first.and.second.prefs, na.rm=T)
median(commonSummary$all.three, na.rm=T)

#commonSummary[commonSummary$first.and.second.prefs == 6,]

commonSummary <- arrange(commonSummary,desc(transition.and.second))

commonSummary$t2rank <- NA
commonSummary$t2rank <- 1:nrow(commonSummary)

max(commonSummary[commonSummary$transition.and.second > 3,]$t2rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.second),]$transition.and.second)
max(commonSummary[commonSummary$transition.and.second > 2,]$t2rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.second),]$transition.and.second)
max(commonSummary[commonSummary$transition.and.second > 1,]$t2rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.second),]$transition.and.second)
#1 - max(commonSummary[commonSummary$transition.and.second > 1,]$t2rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.second),]$transition.and.second)


### transition and first

commonSummary <- arrange(commonSummary,desc(transition.and.first))

commonSummary$t1rank <- NA
commonSummary$t1rank <- 1:nrow(commonSummary)

max(commonSummary[commonSummary$transition.and.first > 3,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.first),]$transition.and.first)
max(commonSummary[commonSummary$transition.and.first > 2,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.first),]$transition.and.first)
max(commonSummary[commonSummary$transition.and.first > 1,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$transition.and.first),]$transition.and.first)

### first and second

commonSummary <- arrange(commonSummary,desc(first.and.second.prefs))

commonSummary$t1rank <- NA
commonSummary$t1rank <- 1:nrow(commonSummary)

max(commonSummary[commonSummary$first.and.second.prefs > 3,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$first.and.second.prefs),]$first.and.second.prefs)
max(commonSummary[commonSummary$first.and.second.prefs > 2,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$first.and.second.prefs),]$first.and.second.prefs)
max(commonSummary[commonSummary$first.and.second.prefs > 1,]$t1rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$first.and.second.prefs),]$first.and.second.prefs)
```

```
### all three

commonSummary <- arrange(commonSummary,desc(all.three))

commonSummary$a3rank <- NA
commonSummary$a3rank <- 1:nrow(commonSummary)

max(commonSummary[commonSummary$all.three > 3,]$a3rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$all.three),]$all.three)
max(commonSummary[commonSummary$all.three > 2,]$a3rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$all.three),]$all.three)
max(commonSummary[commonSummary$all.three > 1,]$a3rank, na.rm = T)/
length(commonSummary[!is.na(commonSummary$all.three),]$all.three)


#### SOA ####
tableIntersections_soa <- function(simClosed, n){

  # transition matrix
  size <- soaP7Pred19 %>%
    filter(fpa2 %in% simClosed) %>%
    select(yXPred) %>%
    sum(na.rm=T)
  #print(paste("size:",size))

  #### Subjunctive: remove pupils who went to removed school from class size, to uprate everyone
else ####

  # Make sure no pupils are allocated to removed schools
  adjustedProbMatrix <- soaMatrixProb19 %>%
    mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed),0,percentOfSchool))

  reallocationList <- list()

  # For each of the removed schools, get the list of feeders and the number coming from each
feeder,
  # and the school type (Grammar/Secondary)
  for(i in 1:length(simClosed)){
    feeders <- soaMatrixProb19 %>%
      filter(fpa2 == simClosed[i]) %>%
      ungroup() %>%
      select(DENI.no.) %>%
      unique() %>%
      pull()

    numberPerFeeder <- soaP7Pred19 %>%
      filter(fpa2 == simClosed[i]) %>%
```

```
    select(DENI.no., Gender,  yXPred)

  gramOrSec <- post_primary %>%
    filter(denino == simClosed[i]) %>%
    select(Gram_Sec_label) %>%
    pull()

  # Work out how much of the school's allocation went to a particular type of school
  percentAllocatedPerPrimary <- adjustedProbMatrix %>%
    filter(DENI.no. %in% feeders, Gram_Sec_label == gramOrSec) %>%
    group_by(DENI.no., Gender) %>%
    summarise(percentAllocated = sum(percentOfSchool)) %>%
    filter(percentAllocated != 0) # remove cases where only one school of a particular type chosen
from the PS

  # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
  # for a particular type of school add up to 100%, and use these %s to make a prediction based on
how many
  # pupils are "sent back" to each primary school to be reallocated
  reallocation <- adjustedProbMatrix %>%
    filter(Gram_Sec_label == gramOrSec) %>%
    inner_join(percentAllocatedPerPrimary, by = c("DENI.no.", "Gender")) %>%
    inner_join(numberPerFeeder, by = c("DENI.no.", "Gender")) %>%
    mutate(revisedPercent = percentOfSchool / percentAllocated) %>%
    mutate(reallocation = yXPred * revisedPercent) %>%
    group_by(fpa2) %>%
    summarise(reallocationSum = sum(reallocation)) %>%
    arrange(desc(reallocationSum))

  reallocationList[[i]] <- reallocation
 }

 # combine reallocation lists for each school in the list and display results
 reallocations <- bind_rows(reallocationList)
 reallocationSummary <- reallocations %>%
  group_by(fpa2) %>%
  summarise(displacedFpas = round(sum(reallocationSum),1)) %>%
  left_join(post_primary[,c(1,2)], by = c("fpa2" = "denino")) %>%
  arrange(desc(displacedFpas)) %>%
  as.data.frame()

 reallocationSummary$displacedFpas <- round(reallocationSummary$displacedFpas *
(size/sum(reallocationSummary$displacedFpas, na.rm=T)),1)

 topNTransitionMatrix <- reallocationSummary %>%
  head(n)
```

```
  count <- stated2Prefs %>%
    filter(fpa %in% simClosed) %>%
    select(stated2)

  # 2nd pref list
  pref2List <- list()

  for(i in 1:length(simClosed)){
    upns <- prefs[prefs$school == simClosed[i] & prefs$pref == 1 & nchar(as.character(prefs$UPN)) >
5,3]
    #upns <- prefs[prefs$school %in% c(1420028) & prefs$pref == 1,3]

    secondPrefs <- prefs %>%
      filter(UPN %in% upns, pref == 2) %>%
      select(school) %>%
      table() %>%
      as.data.frame() %>%
      filter(!. %in% simClosed) %>%
      mutate(fpa = as.numeric(as.character(.))) %>%
      select(fpa, Freq) %>%
      filter(Freq != 0) %>%
      left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
      arrange(desc(Freq))

    #print(secondPrefs$Freq)

    pref2List[[i]] <- secondPrefs

  }
  all2Prefs <- bind_rows(pref2List)

  topNSecondPrefs <- all2Prefs %>%
    arrange(desc(Freq)) %>%
    head(n)

  # 1st PrefList
  pref1List <- list()

  for(i in 1:length(simClosed)){

    upns <- prefs[prefs$school %in% simClosed[i] & prefs$pref == 2 & nchar(as.character(prefs$UPN))
> 5,3]
    #upns <- prefs[prefs$school %in% c(2420054) & prefs$pref == 2,3]

    firstPrefs <- prefs %>%
      filter(UPN %in% upns, pref == 1) %>%
      inner_join(pupils16[,c(2,11)], by = "UPN") %>%
      select(school) %>%
```

```
    table() %>%
    as.data.frame() %>%
    filter(!. %in% simClosed) %>%
    mutate(fpa = as.numeric(as.character(.))) %>%
    select(fpa, Freq) %>%
    filter(Freq != 0) %>%
    left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
    arrange(desc(Freq)) %>%
    left_join(stated2Prefs, by = c("fpa" = "fpa")) %>%
    filter(!is.na(fpa)) %>%
    arrange(desc(Freq))

  pref1List[[i]] <- firstPrefs

 }

 topNFirstPrefs <- bind_rows(pref1List) %>%
  group_by(fpa, schoolname, stated2) %>%
  summarise(Freq = sum(Freq)) %>%
  as.data.frame() %>%
  select(fpa, Freq, stated2, schoolname) %>%
  rename(base = stated2) %>%
  arrange(desc(Freq)) %>%
  filter(!is.na(fpa)) %>%
  head(n)

 c2 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNSecondPrefs) > 4,
         length(intersect(topNFirstPrefs$fpa,topNSecondPrefs$fpa)),NA)
 c3 <- ifelse(nrow(topNTransitionMatrix) > 4 & nrow(topNSecondPrefs) > 4,
         length(intersect(topNTransitionMatrix$fpa2,topNSecondPrefs$fpa)),NA)
 c4 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNTransitionMatrix) > 4,
         length(intersect(topNTransitionMatrix$fpa2,topNFirstPrefs$fpa)),NA)
 c5 <- ifelse(nrow(topNFirstPrefs) > 4 & nrow(topNTransitionMatrix) > 4 & nrow(topNSecondPrefs)
>4,

length(intersect(topNFirstPrefs$fpa,intersect(topNSecondPrefs$fpa,topNTransitionMatrix$fpa2))),
NA)


 row <- c(simClosed, c2, c3, c4, c5)
 cat(".")
 row

 #  print(length(intersect(topNFirstPref$fpa,topNSecondPrefs$fpa)))
 #  print(paste0("For top ",n," results, first and second preferences have
",length(intersect(topNFirstPrefs$fpa,topNSecondPrefs$fpa))," in common"))
 #  print(paste0("For top ",n," results, transition matrix and second preferences have
",length(intersect(topNTransitionMatrix$fpa2,topNSecondPrefs$fpa))," in common"))
```

```
  #  print(paste0("For top ",n," results, transition matrix and first preferences have
",length(intersect(topNTransitionMatrix$fpa2,topNFirstPrefs$fpa))," in common"))

}

in_common_soa <- list()
for(j in 1:length(schoolList$fpa2)){
  row <- tableIntersections_soa(schoolList[j,]$fpa2,6)
  names(row) <- c("fpa","first.and.second.prefs", "transition.and.second", "transition.and.first",
"all.three")
  in_common_soa[[j]] <- row %>% as.data.frame() %>% t()
}

commonSummary_soa <- do.call(rbind.data.frame, in_common_soa)

hist(commonSummary_soa$transition.and.second)
hist(commonSummary_soa$transition.and.first)
hist(commonSummary_soa$first.and.second.prefs)

mean(commonSummary_soa$transition.and.second, na.rm=T)
mean(commonSummary_soa$transition.and.first, na.rm=T)
mean(commonSummary_soa$first.and.second.prefs, na.rm=T)
mean(commonSummary_soa$all.three, na.rm=T)

median(commonSummary_soa$transition.and.second, na.rm=T)
median(commonSummary_soa$transition.and.first, na.rm=T)
median(commonSummary_soa$first.and.second.prefs, na.rm=T)
median(commonSummary_soa$all.three, na.rm=T)

#commonSummary_soa[commonSummary_soa$first.and.second.prefs == 6,]

commonSummary_soa <- arrange(commonSummary_soa,desc(transition.and.second))

commonSummary_soa$t2rank <- NA
commonSummary_soa$t2rank <- 1:nrow(commonSummary_soa)

max(commonSummary_soa[commonSummary_soa$transition.and.second > 3,]$t2rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.second),]$transition.and
.second)
max(commonSummary_soa[commonSummary_soa$transition.and.second > 2,]$t2rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.second),]$transition.and
.second)
max(commonSummary_soa[commonSummary_soa$transition.and.second > 1,]$t2rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.second),]$transition.and
.second)
#1 - max(commonSummary_soa[commonSummary_soa$transition.and.second > 1,]$t2rank, na.rm =
T)/
```

```
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.second),]$transition.and
.second)


### transition and first

commonSummary_soa <- arrange(commonSummary_soa,desc(transition.and.first))

commonSummary_soa$t1rank <- NA
commonSummary_soa$t1rank <- 1:nrow(commonSummary_soa)

max(commonSummary_soa[commonSummary_soa$transition.and.first > 3,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.first),]$transition.and.fir
st)
max(commonSummary_soa[commonSummary_soa$transition.and.first > 2,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.first),]$transition.and.fir
st)
max(commonSummary_soa[commonSummary_soa$transition.and.first > 1,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$transition.and.first),]$transition.and.fir
st)

### first and second

commonSummary_soa <- arrange(commonSummary_soa,desc(first.and.second.prefs))

commonSummary_soa$t1rank <- NA
commonSummary_soa$t1rank <- 1:nrow(commonSummary_soa)

max(commonSummary_soa[commonSummary_soa$first.and.second.prefs > 3,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$first.and.second.prefs),]$first.and.seco
nd.prefs)
max(commonSummary_soa[commonSummary_soa$first.and.second.prefs > 2,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$first.and.second.prefs),]$first.and.seco
nd.prefs)
max(commonSummary_soa[commonSummary_soa$first.and.second.prefs > 1,]$t1rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$first.and.second.prefs),]$first.and.seco
nd.prefs)

### all three

commonSummary_soa <- arrange(commonSummary_soa,desc(all.three))

commonSummary_soa$a3rank <- NA
commonSummary_soa$a3rank <- 1:nrow(commonSummary_soa)

max(commonSummary_soa[commonSummary_soa$all.three > 3,]$a3rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$all.three),]$all.three)
```

```
max(commonSummary_soa[commonSummary_soa$all.three > 2,]$a3rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$all.three),]$all.three)
max(commonSummary_soa[commonSummary_soa$all.three > 1,]$a3rank, na.rm = T)/
length(commonSummary_soa[!is.na(commonSummary_soa$all.three),]$all.three)
```

3. Shiny Applications

3.1 Education Authority Candidate

3.1.1 Description

This first application is the application handed over to SIB for further deployment to the Education Authority. Some of the functionality described in the report has been removed, but other functionality has been added.

3.1.3 Code

3.1.3.1 Global

```
library(shiny)
library(DT)
library(tidyverse)
library(plotly)
library(igraph)
library(shinydashboard)
library(leaflet)
library(geojsonio)

post_primary <- read.csv("data//post_primary_school_attributes.csv", stringsAsFactors = F)
post_primary$schoolname <- as.factor(stringr::str_to_title(post_primary$schoolname))
#post_primary$schoolname <- as.factor(post_primary$schoolname)

primary_schools <- read.csv("data//primary_schools.csv")
primary_schools$postcode <- gsub('\\s+', '', primary_schools$postcode)
primaryEnrolments <- read.csv("data//primaryEnrolments.csv")

postcodes <- read.csv("data//postcodes.csv")

enrollments <- read.csv("data//enrollments201819.csv")
enrollments$denino <- suppressWarnings(as.numeric(as.character(enrollments$denino)))

matrixProb <- read.csv("data//matrixProb2019.csv")
p7Pred <- read.csv("data//p7Pred2019.csv")

primaryMatrixProb <- read.csv("data//primaryMatrixProb2016.csv")
primaryPred <- read.csv("data//primaryPred2016.csv")

prefs <- read.csv("data//p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]

sa <- geojsonio::geojson_read("data\\Small Areas.json",
                              what = "sp")
```

```
schoolList <- matrixProb %>%
  left_join(post_primary[,1:2],by = c("fpa2" = "denino")) %>%
  select(fpa2,schoolname) %>%
  mutate(schoolname = stringr::str_to_title(schoolname)) %>%
  filter(!is.na(fpa2),!is.na(schoolname)) %>%
  unique()

yearList <- prefs %>%
  select(year) %>%
  unique() %>%
  arrange(year) %>%
  pull()

noSixthForm <- enrollments[enrollments$year13 == 0,]$denino
```

3.1.3.2 Server

```
# Function to list 'acceptable schools'; that is, if a primary school of a certain type closes, which other
primary
# schools are likely to be considered by the pupils who attended the clsoed school. There is no
evidence behind the
# acceptable types of schools in the statements below
getAcceptableList <- function(manType){
  if(manType == "GMI"){
    acceptableTypes <- c("Controlled Integrated", "Catholic Maintained", "Controlled", "Voluntary",
"GMI")
  }
  if(manType == "Controlled Integrated"){
    acceptableTypes <- c("Controlled Integrated", "Catholic Maintained", "Controlled", "Voluntary",
"GMI")
  }
  if(manType == "Catholic Maintained"){
    acceptableTypes <- c("Controlled Integrated", "Catholic Maintained", "GMI")
  }
  if(manType == "Other Maintained"){
    acceptableTypes <- c("Catholic Maintained", "Other Maintained")
  }
  if(manType == "Controlled"){
    acceptableTypes <- c("Controlled Integrated", "Controlled", "Voluntary", "GMI")
  }
  if(manType == "Voluntary"){
    acceptableTypes <- c("Controlled Integrated", "Controlled", "Voluntary", "GMI")
  }
  acceptableTypes
}
```

```
shinyServer(function(input, output) {
  # Select school and exclude school should update based on Primary or Post-primary radio button
  output$selectSchool <- renderUI({
    if(input$level == "Post-primary"){
      choiceList = paste(schoolList$schoolname,schoolList$fpa2,sep=":")
    }
    else{
      choiceList = paste(primary_schools[1:813,]$schoolname, primary_schools[1:813,]$denino, sep =
":")
    }
    selectInput("schools", "Schools", multiple = TRUE,
                choices = choiceList)
  })

  output$excludeSchool <- renderUI({
    if(input$level == "Post-primary"){
      choiceList = paste(schoolList$schoolname,schoolList$fpa2,sep=":")
    }
    else{
      choiceList = paste(primary_schools[1:813,]$schoolname, primary_schools[1:813,]$denino, sep =
":")
    }
    selectInput("exSchools", "Exclude Schools", multiple = TRUE,
                choices = choiceList)
  })

  # 6th form radio buttons and text describing the map should also depend on Primary or Post-
primary radio button

  output$RadioButtons6thForm <- renderUI({
    if(input$level == "Post-primary"){
      radioButtons("sixthForm", "Include Years", c("Years 8 to 12", "Years 13 and 14", "Years 8 to 14"),
"Years 8 to 12")
    }
  })

  output$mapDescription <- renderUI({
    if(input$level == "Primary"){
      p("Small areas containing pupils enrolled in P1 at the selected schools are shown, with small
areas
                containing no children enrolled in P1 at any other primary school in red.")
    }
  })

  # Take the inputs for schools and excluded schools and create a list of deninos
  simClosed <- eventReactive(input$run,{
    simClosed <- c()
    if(length(input$schools > 0)){
```

```
    for(i in 1:length(input$schools)){
      simClosed <- c(simClosed,strsplit(input$schools,":")[[i]][2])
    }
  }
  simClosed
  }
)

 excluded <- eventReactive(input$run,{
  simClosed <- c()
  if(length(input$exSchools) == 0){
   simClosed <- "0"
  }
  else {
   for(i in 1:length(input$exSchools)){
    simClosed <- c(simClosed,strsplit(input$exSchools,":")[[i]][2])
   }
  }
  simClosed
 })

 #### realloction summary: this is the main piece of work, creating a data.frame that is used to
populate tables ####
 # and charts by adapting the transition matrix based on the input schools. There are Primary and
post-primary versions
 reallocationSummary <- reactive({
  req(input$run) # waits for the run scenario button
  if(input$level == "Post-primary")
  {
   ## Update the probability matrix: set probability to 0 where the school is removed from the
choice set
   adjustedProbMatrix <- matrixProb %>%
    mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed(), excluded()),0,percentOfSchool))

   reallocationList <- list()

   # For each of the removed schools, get the list of feeders and the number coming from each
feeder,
   # and the school type (Grammar/Secondary)
   for(i in 1:length(simClosed())){
    feeders <- matrixProb %>%
     filter(fpa2 == simClosed()[i]) %>%
     ungroup() %>%
     select(DENI.no.) %>%
     unique() %>%
     pull()

    numberPerFeeder <- p7Pred %>%
```

```
    filter(fpa2 == simClosed()[i]) %>%
    select(DENI.no., Gender,  yXPred)

  gramOrSec <- post_primary %>%
    filter(denino == simClosed()[i]) %>%
    select(Gram_Sec_label) %>%
    pull()

  # Work out how much of the school's allocation is accounted for by the remaining schools
  percentAllocatedPerPrimary <- adjustedProbMatrix %>%
    filter(DENI.no. %in% feeders, Gram_Sec_label == gramOrSec) %>%
    group_by(DENI.no., Gender) %>%
    summarise(percentAllocated = sum(percentOfSchool))

  # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
  # for a particular type of school add up to 100%, and use these %s to make a prediction based
on how many
  # pupils are "sent back" to each primary school to be reallocated
  reallocation <- adjustedProbMatrix %>%
    filter(Gram_Sec_label == gramOrSec) %>%
    inner_join(percentAllocatedPerPrimary, by = c("DENI.no.", "Gender")) %>%
    inner_join(numberPerFeeder, by = c("DENI.no.", "Gender")) %>%
    mutate(revisedPercent = percentOfSchool / percentAllocated) %>%
    mutate(reallocation = yXPred * revisedPercent) %>%
    group_by(fpa2) %>%
    summarise(reallocationSum = sum(reallocation)) %>%
    arrange(desc(reallocationSum))

  reallocationList[[i]] <- reallocation
}

## Now exclude schools with no 6th form
# Make sure no pupils are allocated to removed schools
adjustedProbMatrix <- matrixProb %>%
  mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed(), noSixthForm),0,percentOfSchool))

reallocationList6th <- list()
for(i in 1:length(simClosed())){
  feeders <- matrixProb %>%
    filter(fpa2 == simClosed()[i]) %>%
    ungroup() %>%
    select(DENI.no.) %>%
    unique() %>%
    pull()

  numberPerFeeder <- p7Pred %>%
    filter(fpa2 == simClosed()[i]) %>%
```

```
    select(DENI.no., Gender,  yXPred)

  gramOrSec <- post_primary %>%
    filter(denino == simClosed()[i]) %>%
    select(Gram_Sec_label) %>%
    pull()

  # Work out how much of the school's allocation is accounted for by the remaining schools
  percentAllocatedPerPrimary <- adjustedProbMatrix %>%
    filter(DENI.no. %in% feeders, Gram_Sec_label == gramOrSec) %>%
    group_by(DENI.no., Gender) %>%
    filter(!fpa2 %in% c(simClosed(),excluded())) %>%
    summarise(percentAllocated = sum(percentOfSchool)) %>%
    filter(percentAllocated != 0) # remove cases where only one school of a particular type chosen
from the PS

  # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
  # for a particular type of school add up to 100%, and use these %s to make a prediction based
on how many
  # pupils are "sent back" to each primary school to be reallocated
  reallocation <- adjustedProbMatrix %>%
    filter(Gram_Sec_label == gramOrSec) %>%
    inner_join(percentAllocatedPerPrimary, by = c("DENI.no.", "Gender")) %>%
    inner_join(numberPerFeeder, by = c("DENI.no.", "Gender")) %>%
    mutate(revisedPercent = percentOfSchool / percentAllocated) %>%
    mutate(reallocation = yXPred * revisedPercent) %>%
    group_by(fpa2) %>%
    summarise(reallocationSum = sum(reallocation)) %>%
    arrange(desc(reallocationSum))

  reallocationList6th[[i]] <- reallocation
 }

 # combine reallocation lists for 6th formers, ready to add to main list
 reallocations6th <- bind_rows(reallocationList6th) %>%
  group_by(fpa2) %>%
  summarise(displacedFpas6th = sum(reallocationSum)) %>%
  as.data.frame()

 # combine reallocation lists for each schoo, and add 6th form reallocations
 reallocations <- bind_rows(reallocationList)
 reallocationSummary <- reallocations %>%
  group_by(fpa2) %>%
  summarise(displacedFpas = sum(reallocationSum)) %>%
  left_join(post_primary[,c(1,2)], by = c("fpa2" = "denino")) %>%
  left_join(reallocations6th, by = "fpa2") %>%
  arrange(desc(displacedFpas)) %>%
```

```
      as.data.frame()

    # calculate percentages of reallocations to each school, with and without 6th form
    reallocationSummary$percentage <- round(100 * reallocationSummary$displacedFpas /
sum(reallocationSummary$displacedFpas, na.rm=T),1)
    reallocationSummary$percentage6th <- round(100 * reallocationSummary$displacedFpas6th /
sum(reallocationSummary$displacedFpas6th, na.rm = T),1)

    # Work out the nubmer of people enrolled in each year at the school. This will be multiplies by
percentage reallocations
    # based on a single year's transfer
    yearSize <- enrollments %>%
      filter(denino %in% simClosed()) %>%
      summarise(year8 = sum(year8), year9 = sum(year9), year10 = sum(year10), year11 =
sum(year11),
                year12 = sum(year12), year13 = sum(year13), year14 = sum(year14), total.pupils =
sum(total.pupils))

    reallocationByYear <- cbind(reallocationSummary, yearSize) %>%
      mutate(year8 = round(percentage/100 * year8,1), year9 = round(percentage/100 * year9,1),
year10 = round(percentage/100 * year10,1),
             year11 = round(percentage/100 * year11, 1), year12 = round(percentage/100 * year12, 1),
             year13 = round(percentage6th/100 * year13, 1), year14 = round(percentage6th/100 *
year14, 1)) %>%
      mutate(percentageY8.12 = percentage,
             percentageY8.14 = round(100 * (year8 + year9 + year10 + year11 + year12 + year13 +
year14)/total.pupils,1), denino = fpa2,
             percentageY13.14 = round(100 * (year13 + year14) / sum(year13,year14, na.rm=T)),1)
   }
   else{
    # set probability to 0 for schools removed from choice set
    adjustedProbMatrix <- primaryMatrixProb %>%
      mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed(), excluded()),0,percentOfSchool))

    reallocationList <- list()

    # For each of the removed schools, get the list of feeders and the number coming from each
feeder,
    # and the school type
    for(i in 1:length(simClosed())){
      feeders <- primaryMatrixProb %>%
        filter(fpa2 == simClosed()[i]) %>%
        ungroup() %>%
        select(DENI.no.) %>%
        unique() %>%
        pull()

      numberPerFeeder <- adjustedProbMatrix %>%
```

```
      filter(fpa2 == simClosed()[i]) %>%
      select(DENI.no., primarySizeY1) %>%
      unique() %>%
      mutate(DENI.no. = as.factor(DENI.no.))

    # Get the type of school and have acceptable types for each type:
    manType <- unique(primaryMatrixProb[primaryMatrixProb$fpa2 ==
simClosed()[i],]$management.type)
    acceptableTypes <- getAcceptableList(manType)

    # Work out how much of the school's allocation went to a particular type of school
    percentAllocatedPerPrimary <- adjustedProbMatrix %>%
      filter(DENI.no. %in% feeders, management.type %in% acceptableTypes) %>%
      filter(!fpa2 %in% c(simClosed(),excluded())) %>%
      group_by(DENI.no.) %>%
      summarise(percentAllocated = sum(percentOfSchool))

    # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
    # for a particular type of school add up to 100%, and use these %s to make a prediction based
on how many
    # pupils are "sent back" to each primary school to be reallocated
    reallocation <- adjustedProbMatrix %>%
      filter(management.type %in% acceptableTypes) %>%
      inner_join(numberPerFeeder, by = c("DENI.no.")) %>%
      left_join(percentAllocatedPerPrimary, by = c("DENI.no.")) %>%
      mutate(revisedPercent = ifelse(!is.na(percentAllocated),percentOfSchool /
percentAllocated,ifelse(fpa2 %in% excluded(),0,1))) %>% # Trying to capture areas that don't send
children anywhere else
      mutate(reallocation = primarySizeY1.x * revisedPercent) %>%
      group_by(fpa2) %>%
      summarise(reallocationSum = sum(reallocation)) %>%
      filter(!fpa2 %in% excluded()) %>%
      arrange(desc(reallocationSum))

    reallocationList[[i]] <- reallocation
  }

  # combine reallocation lists for each school in the list and display results
  reallocations <- bind_rows(reallocationList)
  reallocationSummary <- reallocations %>%
    mutate(fpa2 = as.factor(fpa2)) %>%
    group_by(fpa2) %>%
    summarise(displacedFpas = sum(reallocationSum)) %>%
    left_join(primary_schools[,c(1,2)], by = c("fpa2" = "denino")) %>%
    arrange(desc(displacedFpas)) %>%
    as.data.frame()
```

```
    reallocationSummary$percentage <- round(100 * reallocationSummary$displacedFpas /
sum(reallocationSummary$displacedFpas),1)

    yearSize <- primaryEnrolments %>%
     filter(denino %in% simClosed()) %>%
     summarise(year1 = sum(year.1), year2 = sum(year.2), year3 = sum(year.3), year4 = sum(year.4),
           year5 = sum(year.5), year6 = sum(year.6), year7 = sum(year.7), total.pupils =
sum(total.enrolment))

    reallocationByYear <- cbind(reallocationSummary, yearSize) %>%
     mutate(year1 = round(percentage/100 * year1,1), year2 = round(percentage/100 * year2,1),
year3 = round(percentage/100 * year3,1),
          year4 = round(percentage/100 * year4, 1), year5 = round(percentage/100 * year5, 1),
          year6 = round(percentage/100 * year6, 1), year7 = round(percentage/100 * year7, 1),
          denino = fpa2)
  }
  #print(reallocationByYear)
  reallocationByYear
 })

 output$impactSummary <- DT::renderDataTable({
  if(input$level == "Post-primary"){
   if(input$sixthForm == "Years 8 to 12"){
    reallocationByYear <- reallocationSummary() %>%
     mutate(percentage = percentageY8.12) %>%
     select(schoolname, denino, percentage, year8, year9, year10, year11, year12)
   }
   if(input$sixthForm == "Years 13 and 14"){
    reallocationByYear <- reallocationSummary() %>%
     mutate(percentage = percentageY13.14) %>%
     select(schoolname, denino, percentage, year13, year14)
   }
   if(input$sixthForm == "Years 8 to 14"){
    reallocationByYear <- reallocationSummary() %>%
     mutate(percentage = percentageY8.14) %>%
     select(schoolname, denino, percentage, year8, year9, year10, year11, year12, year13, year14)
   }
  }
  else{
   reallocationByYear <- reallocationSummary() %>%
    select(schoolname, denino, percentage, year1, year2, year3, year4, year5, year6, year7)
  }

  impactSummary <- reallocationByYear
  #impactSummary$denino %in% simClosed()
  levels(impactSummary$schoolname)[levels(impactSummary$schoolname) ==
impactSummary[impactSummary$denino %in% simClosed(),]$schoolname] <- "Not Reallocated"
```

```
  DT::datatable(impactSummary)
 })

 output$network <- renderPlot({
  if(input$level == "Post-primary"){
   if(input$sixthForm == "Years 8 to 12"){
    reallocations <- reallocationSummary() %>%
     mutate(percentage = percentageY8.12)
   }
   if(input$sixthForm == "Years 13 and 14"){
    reallocations <- reallocationSummary() %>%
     mutate(percentage = percentageY13.14)
   }
   if(input$sixthForm == "Years 8 to 14"){
    reallocations <- reallocationSummary() %>%
     mutate(percentage = percentageY8.14)
   }
   reallocatedTop10 <- reallocations %>%
    arrange(desc(percentage)) %>%
    head(12)

   schoolListNames <- schoolList[!is.na(schoolList$schoolname),]
   schoolListNames  <- schoolListNames %>%
    filter(fpa2 == simClosed()[1] | fpa2 %in% reallocatedTop10$fpa2) %>%
    mutate(schoolname = as.factor(schoolname))

   if(length(simClosed()) > 1 ){
    levels(schoolListNames$schoolname)[levels(schoolListNames$schoolname) ==
schoolListNames[schoolListNames$fpa2 == simClosed()[1],]$schoolname] <- "Listed Schools"
    print(str(schoolListNames$schoolname))
   }

   totalPupils <- enrollments %>%
    filter(denino %in% simClosed()) %>%
    summarise(pupils = sum(total.pupils))

   reallocationBySchool <- cbind(simClosed()[1],reallocatedTop10, totalPupils) %>%
    mutate(total.pupils = percentage/100 * pupils) %>%
    select(-displacedFpas)
  }
  else{
   reallocations <- reallocationSummary()
   reallocatedTop10 <- reallocations %>%
    arrange(desc(percentage)) %>%
    filter(!is.na(fpa2)) %>%
    mutate(fpa2 = ifelse(fpa2 %in% simClosed(), -1, fpa2)) %>%
    head(12)
```

```
    schoolListNames <- primary_schools[!is.na(primary_schools$schoolname),1:2] %>%
      mutate(fpa2 = denino) %>%
      select(fpa2, schoolname)
    schoolListNames  <- schoolListNames %>%
      filter(fpa2 == simClosed()[1] | fpa2 %in% reallocatedTop10$fpa2)
    levels(schoolListNames$schoolname) <- c(levels(schoolListNames$schoolname),"Not
Reallocated")
    levels(schoolListNames$fpa2) <- c(levels(schoolListNames$fpa2),-1)

    schoolListNames <- rbind(schoolListNames,c(-1, "Not Reallocated"))

    if(length(simClosed()) > 1 ){
      levels(schoolListNames$schoolname)[levels(schoolListNames$schoolname) ==
schoolListNames[schoolListNames$fpa2 == simClosed()[1],]$schoolname] <- "Listed Schools"
    }

    totalPupils <- primaryEnrolments %>%
      filter(denino %in% simClosed()) %>%
      summarise(pupils = sum(total.enrolment))

    reallocationBySchool <- cbind(simClosed()[1],reallocatedTop10, totalPupils) %>%
      mutate(total.pupils = percentage/100 * pupils) %>%
      select(-displacedFpas)
  }

  net <- graph_from_data_frame(d=reallocationBySchool, vertices=schoolListNames, directed=T)

  net <- simplify(net, remove.multiple = F, remove.loops = T)

  vCols <- ifelse(schoolListNames$fpa2 == -1, "red", "orange")

  layout <- layout_in_circle(net, order = c(V(net)[(degree(net)) ==
max(degree(net))],V(net)[(degree(net)) != max(degree(net))]))

  plot(net, edge.label = edge_attr(net, "percentage"), vertex.label = V(net)$schoolname,
edge.width=abs((E(net)$percentage)/2),
       layout = layout, vertex.color = vCols)

 }, width = 750, height = 750)

 output$displacementSummary <- DT::renderDataTable({
  if(input$level == "Post-primary"){
    displaced <- enrollments %>%
      filter(denino %in% simClosed()) %>%
      select(Schoolname, denino, total.pupils, SEN5, SEN1to4, fsme, Newcomers)
  }
  else{
    displaced <- primaryEnrolments %>%
```

```
    filter(denino %in% simClosed()) %>%
    select(schoolname, denino, total.enrolment, SEN5, SEN1to4, fsme, Newcomer)
  }

  DT::datatable(displaced)
 })

 output$map <- renderLeaflet({
  if(input$level == "Post-primary"){

   schools <- schoolList %>%
    left_join(post_primary, by = c("fpa2" = "denino")) %>%
    left_join(postcodes[,c(5,46,47)], by = c("school_postcode" = "pcd"))

   secondarySchools <- schools %>%
    filter(Gram_Sec_label == "Secondary")

   grammarSchools <- schools %>%
    filter(Gram_Sec_label == "Grammar")

   selectedSchools <- post_primary %>%
    filter(denino %in% simClosed()) %>%
    left_join(postcodes[,c(5,46,47)], by = c("school_postcode" = "pcd"))

   m <- leaflet() %>%
    addProviderTiles(providers$Stamen.TonerLite)

   for(i in 1:nrow(selectedSchools)){
    m <- addAwesomeMarkers(m, selectedSchools[i,]$long, selectedSchools[i,]$lat)
   }

   for(i in 1:nrow(grammarSchools)){
    m <- addCircles(m,grammarSchools[i,]$long, grammarSchools[i,]$lat,
                    label = grammarSchools[i,]$schoolname.x,
                    stroke = T, fillOpacity = 0.5,
                    color = "purple", weight = 3, radius = 100)
   }
   for(i in 1:nrow(secondarySchools)){
    m <- addRectangles(m,lng1 = secondarySchools[i,]$long - 0.0012, lat1 = secondarySchools[i,]$lat
- 0.0006,
                       lng2 = secondarySchools[i,]$long + 0.0012, lat2 = secondarySchools[i,]$lat + 0.0006,
                    label = secondarySchools[i,]$schoolname.x,
                    stroke = T, fillOpacity = 0.5,
                    color = "purple", weight = 3)
   }

  }
  else{
```

```
    saList <- primaryMatrixProb %>%
      filter(fpa2 %in% simClosed()) %>%
      select(DENI.no.) %>%
      unique() %>%
      pull()

    singlesList <- list()

    for(i in 1:length(simClosed())){
      manType <- unique(primaryMatrixProb[primaryMatrixProb$fpa2 ==
simClosed()[i],]$management.type)
      acceptableTypes <- getAcceptableList(manType)
      shortSaList <- primaryMatrixProb %>%
        filter(fpa2 %in% simClosed()[i]) %>%
        select(DENI.no.) %>%
        unique() %>%
        pull()

      singleSas <- primaryMatrixProb %>%
        filter(management.type %in% acceptableTypes, DENI.no. %in% shortSaList, !fpa2 %in%
excluded()) %>%
        group_by(DENI.no.) %>%
        summarise(destinations = n())
      print(singleSas)
      singlesList[[i]] <- singleSas
    }

    singleSas <- bind_rows(singlesList)

    otherSchools <- reallocationSummary() %>%
      left_join(primary_schools, by = c("fpa2" = "denino")) %>%
      left_join(postcodes[,c(5,46,47)], by = c("postcode" = "pcd"))

    selectedSchools <- primary_schools %>%
      filter(denino %in% simClosed()) %>%
      left_join(postcodes[,c(5,46,47)], by = c("postcode" = "pcd"))

    reach <- subset(sa,id %in% saList)
    reach$single <- ifelse(reach$id %in% singleSas[singleSas$destinations == 1,]$DENI.no., 1,0)
    singles <- sum(reach$single)
    reach$single <- as.factor(reach$single)

    factpal <- colorFactor(c("black"), reach$single)
    if(singles > 0){
      factpal <- colorFactor(c("black", "red"), reach$single)
    }
```

```
    m <- leaflet(reach) %>%
     addProviderTiles(providers$Stamen.TonerLite) %>%
     addPolygons(stroke = TRUE, smoothFactor = 0.3, fillOpacity = 0.5, weight = 1,
                 color = ~factpal(single))

    for(i in 1:nrow(selectedSchools)){
    m <- addAwesomeMarkers(m, selectedSchools[i,]$long, selectedSchools[i,]$lat)
    }

    for(i in 1:nrow(otherSchools)){
     m <- addCircles(m,otherSchools[i,]$long, otherSchools[i,]$lat,
                     label = otherSchools[i,]$schoolname.x,
                     stroke = T, fillOpacity = 0.5,
                     color = "purple", weight = 3)
    }
   }

   m
})
})
```

3.1.3.3 UI

```
dashboardPage(
 dashboardHeader(title = tags$a(tags$img(src="EAbrand.jpg"))),
 dashboardSidebar(width = 350,
                  radioButtons("level", "School Level", c("Primary", "Post-primary"), "Post-primary"),
                  htmlOutput("selectSchool"),
                  #radioButtons("geo","Source (for Transition Matrix)",c("Primary School","SOA"),"Primary
School"),
                  #radioButtons("matrixYear","Year (for Transition Matrix)",yearList,2019),
                  htmlOutput("RadioButtons6thForm"),
                  htmlOutput("excludeSchool"),
                  actionButton("run", "Run scenario"),
                  tags$img(src="SIBbrand.jpg")),
 dashboardBody(fluidRow(
  tabBox(title = "Transition Matrix",
         id = "transitionMatrix", width = "100%", height = "875px",
         tabPanel("Tables",
                  p("Showing the reallocation of pupils from the selected school(s), based on the
proportion
                    of pupils from the primary school of origin who went to each school, (accounting
                    for gender and grammar/secondary choice. Figures are scaled up as pupils cannot
                    be reallocated if their school did not send pupils of the same gender to another school
                    of the same type"),
                  dataTableOutput("impactSummary")
                  ),
         tabPanel("Charts",
                  p("Graph showing the percentage of pupils (excluding 6th form) from selected schools
who would be expected to
```

```
                go to each of the alternative schools if their first choice school was not available, based
on primary school
                 attended. A maximum of the top 12 other schools are shown."),
               plotOutput("network")
               ),
        tabPanel("Displacement Summary",
               dataTableOutput("displacementSummary")
        ),
        tabPanel("Map",
               htmlOutput("mapDescription"),
               leafletOutput("map")
        )
   )
 )
 )
)
```

3.2 Alpha Version

3.2.1 Description

This application was developed before the EA Candidate application and is more aligned to the project than the actual needs of the Education Authority.

3.2.3 Code

```
library(shiny)
library(DT)
library(tidyverse)
library(plotly)
library('igraph')

post_primary <- read.csv("..//post_primary_school_attributes.csv", stringsAsFactors = F)
post_primary$schoolname <- as.factor(post_primary$schoolname)
pupils16 <- read.csv("pupils16.csv", stringsAsFactors = F)

edges <- read.csv("..//edges.csv")
dirEdges <- read.csv("..//dirEdges.csv")

# deal with the Strandtown curiosity
pupils16[pupils16$denino %in% c(1016242,1010304,1010012),]$denino <- 1010252

# Import prepared data
## 2017
matrixProb17 <- read.csv("matrixProb2017.csv")
p7Pred17 <- read.csv("p7Pred2017.csv")

soaMatrixProb17 <- read.csv("soaMatrixProb2017.csv")
soaP7Pred17 <- read.csv("soaP7Pred2017.csv")

##2018
```

```
matrixProb18 <- read.csv("matrixProb2018.csv")
p7Pred18 <- read.csv("p7Pred2018.csv")

soaMatrixProb18 <- read.csv("soaMatrixProb2018.csv")
soaP7Pred18 <- read.csv("soaP7Pred2018.csv")

## 2019
matrixProb19 <- read.csv("matrixProb2019.csv")
p7Pred19 <- read.csv("p7Pred2019.csv")

soaMatrixProb19 <- read.csv("soaMatrixProb2019.csv")
soaP7Pred19 <- read.csv("soaP7Pred2019.csv")

#prefs <- read.csv("..\\AllP7Preferences.csv", header = FALSE)
#prefs <- rename(prefs,school = V1, pref = V2, UPN = V3, year = V4)
prefs <- read.csv("..\\p7Preferences 2017-19.csv")
prefs <- prefs[,1:4]

stated2Prefs <- prefs %>%
 filter(pref %in% c(1,2)) %>%
 select(-year) %>%
 group_by(UPN) %>%
 summarise(fpa = school[which.min(pref)],secondPref = n() - 1) %>%
 ungroup() %>%
 select(fpa, secondPref) %>%
 mutate(fpa = as.numeric(as.character(fpa))) %>%
 group_by(fpa) %>%
 summarise(stated1 = n(), stated2 = sum(secondPref)) %>%
 as.data.frame()

schoolList <- matrixProb19 %>%
 left_join(post_primary[,1:2],by = c("fpa2" = "denino")) %>%
 select(fpa2,schoolname) %>%
 unique()

yearList <- prefs %>%
 select(year) %>%
 unique() %>%
 arrange(year) %>%
 pull()

# Define UI for application that draws a histogram
ui <- fluidPage(

 # Application title
 titlePanel("School Preferences"),

 # Sidebar with a slider input for number of bins
```

```
sidebarLayout(
 sidebarPanel(
  selectInput("schools", "Schools", multiple = TRUE,
          choices = paste(schoolList$schoolname,schoolList$fpa2,sep=":"),
          selected="LAGAN COLLEGE:4260255"),
  radioButtons("geo","Source (for Transition Matrix)",c("Primary School","SOA"),"Primary School"),
  radioButtons("matrixYear","Year (for Transition Matrix)",yearList,2019)
 ),

 # Show a plot of the generated distribution
 mainPanel(
  tabsetPanel(
   tabPanel("Tables",
    h1("Transition Matrix"),
    p("Showing the reallocation of pupils from the selected school(s), based on the proportion
      of pupils from the primary school of origin who went to each school, (accounting
      for gender and grammar/secondary choice. Figures are scaled up as pupils cannot
      be reallocated if their school/SOA did not send pupils of the same gender to another school
      of the same type"),
    dataTableOutput("impactSummary"),
    h1("Second Preferences"),
    p("Recorded second preference data scaled to account for the number of pupils being
      reallocated"),
    htmlOutput("secondPreferenceBase"),
    dataTableOutput("secondPreferenceSummary"),
    h1("Source of second preferences"),
    p("Showing the first preferences, where a second preference is for the selected school(s).
      Numbers have been scaled to estimate second preferences from each school. The base
      column refers to the number of second preferences recorded for each school over 3 years,
      and where the base is low numbers should be treated with caution."),
    dataTableOutput("firstPreferenceSummary")
   ),
   tabPanel("Charts",
   # p("This chart indicates how stable the transition matrix has been over the
   #   time-period available"),
   # plotlyOutput("stability"),
    p("The chart below shows estimated first preference applications for 2017-23, with dots
      representing the actual number of first preference applications"),
   plotlyOutput("forecast"),
   p("Note that the Transition Matrix (used for estimates/predictions) is created from the
     intersection of pupils in the 16-17 pupil file and first preferences recorded in 20XX.
     If pupils are missing from the 16-17 file, FPAs will be understimated. If preferences are
     missing, FPAs will be overestimated, as they were worked out as a fraction of the pupils
     who expressed a first preference multiplied by the number of pupils (whether or not they
     expressed a preference."),
   plotOutput("network")
   )
  )
```

```
   )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

  #### Remove schools scenarios ####

  simClosed <- reactive({
    simClosed <- c()
    for(i in 1:length(input$schools)){
      simClosed <- c(simClosed,strsplit(input$schools,":")[[i]][2])
    }
    simClosed
  })

  p7Pred <- reactive({
    if(input$geo == "Primary School"){
      chosenP7Pred <- eval(parse(text=paste0("p7Pred",substr(input$matrixYear,3,4))))
    }
    if(input$geo == "SOA"){
      chosenP7Pred <- eval(parse(text=paste0("soaP7Pred",substr(input$matrixYear,3,4))))
    }
    chosenP7Pred
  })

  matrixProb <- reactive({
    if(input$geo == "Primary School"){
      chosenMatrixProb <- eval(parse(text=paste0("matrixProb",substr(input$matrixYear,3,4))))
    }
    if(input$geo == "SOA"){
      chosenMatrixProb <- eval(parse(text=paste0("soaMatrixProb",substr(input$matrixYear,3,4))))
    }
    chosenMatrixProb
  })

    output$impactSummary <- DT::renderDataTable({

      size <- p7Pred() %>%
        filter(fpa2 %in% simClosed()) %>%
        select(yXPred) %>%
        sum(na.rm=T)
      #print(paste("size:",size))

    #### Subjunctive: remove pupils who went to removed school from class size, to uprate everyone
else ####
```

```
  # Make sure no pupils are allocated to removed schools
  adjustedProbMatrix <- matrixProb() %>%
    mutate(percentOfSchool = ifelse(fpa2 %in% c(0,simClosed()),0,percentOfSchool))

  reallocationList <- list()

  # For each of the removed schools, get the list of feeders and the number coming from each
feeder,
  # and the school type (Grammar/Secondary)
  for(i in 1:length(simClosed())){
    feeders <- matrixProb() %>%
      filter(fpa2 == simClosed()[i]) %>%
      ungroup() %>%
      select(DENI.no.) %>%
      unique() %>%
      pull()

    numberPerFeeder <- p7Pred() %>%
      filter(fpa2 == simClosed()[i]) %>%
      select(DENI.no., Gender,  yXPred)

    gramOrSec <- post_primary %>%
      filter(denino == simClosed()[i]) %>%
      select(Gram_Sec_label) %>%
      pull()

    # Work out how much of the school's allocation went to a particular type of school
    percentAllocatedPerPrimary <- adjustedProbMatrix %>%
      filter(DENI.no. %in% feeders, Gram_Sec_label == gramOrSec) %>%
      group_by(DENI.no., Gender) %>%
      summarise(percentAllocated = sum(percentOfSchool)) %>%
      filter(percentAllocated != 0) # remove cases where only one school of a particular type chosen
from the PS

    # Divide allocations to each school of appropriate type by the above %, so that the percentage
allocations
    # for a particular type of school add up to 100%, and use these %s to make a prediction based on
how many
    # pupils are "sent back" to each primary school to be reallocated
    reallocation <- adjustedProbMatrix %>%
      filter(Gram_Sec_label == gramOrSec) %>%
      inner_join(percentAllocatedPerPrimary, by = c("DENI.no.", "Gender")) %>%
      inner_join(numberPerFeeder, by = c("DENI.no.", "Gender")) %>%
      mutate(revisedPercent = percentOfSchool / percentAllocated) %>%
      mutate(reallocation = yXPred * revisedPercent) %>%
      group_by(fpa2) %>%
      summarise(reallocationSum = sum(reallocation)) %>%
      arrange(desc(reallocationSum))
```

```
    reallocationList[[i]] <- reallocation
   }

   # combine reallocation lists for each school in the list and display results
   reallocations <- bind_rows(reallocationList)
   reallocationSummary <- reallocations %>%
    group_by(fpa2) %>%
    summarise(displacedFpas = round(sum(reallocationSum),1)) %>%
    left_join(post_primary[,c(1,2)], by = c("fpa2" = "denino")) %>%
    arrange(desc(displacedFpas)) %>%
    as.data.frame()

   reallocationSummary$displacedFpas <- round(reallocationSummary$displacedFpas *
(size/sum(reallocationSummary$displacedFpas, na.rm=T)),1)

   DT::datatable(reallocationSummary)
   })

   output$secondPreferenceBase <- renderUI({
    count <- stated2Prefs %>%
     filter(fpa %in% simClosed()) %>%
     select(stated2)

    #p(strong(paste0("<font color=\"red\">These estimates are based on ",count," stated second
preferences (over 3 years)</font>" )))
    col <- "black"
    if(min(count) < 20){
     col <- "red"
    }
    HTML(paste0("<p style=\"color: ",col,"\">These estimates are based on ",count," stated second
preferences (over 3 years)</p>"))
   })

   output$secondPreferenceSummary <- DT::renderDataTable({

    # Get the number of reallocations
    size <- p7Pred() %>%
     filter(fpa2 %in% simClosed()) %>%
     select(fpa2, yXPred) %>%
     group_by(fpa2) %>%
     summarise(total = sum(yXPred))
    #print(paste("size:",size))

    pref2List <- list()

    for(i in 1:length(simClosed())){
```

```
      upns <- prefs[prefs$school == simClosed()[i] & prefs$pref == 1 & nchar(as.character(prefs$UPN))
> 5,3]
      #upns <- prefs[prefs$school %in% c(1420028) & prefs$pref == 1,3]

      secondPrefs <- prefs %>%
        filter(UPN %in% upns, pref == 2) %>%
        select(school) %>%
        table() %>%
        as.data.frame() %>%
        filter(!. %in% simClosed()) %>%
        mutate(fpa = as.numeric(as.character(.))) %>%
        select(fpa, Freq) %>%
        filter(Freq != 0) %>%
        left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
        arrange(desc(Freq))

      #print(secondPrefs$Freq)

        secondPrefs$Freq <- round(secondPrefs$Freq * max(size[size$fpa2 ==
simClosed()[i],2])/sum(secondPrefs$Freq, na.rm=T),1)
        pref2List <- bind_rows(pref2List, secondPrefs)

    }

#
    all2Prefs <- pref2List %>%
      group_by(fpa, schoolname) %>%
      summarise(Freq = sum(Freq)) %>%
      as.data.frame() %>%
      select(fpa, Freq, schoolname) %>%
      arrange(desc(Freq)) %>%
      filter(!is.na(fpa))

    #print(sum(secondPrefs$Freq, na.rm=T))

    DT::datatable(all2Prefs)
  })

  output$firstPreferenceSummary <- DT::renderDataTable({

    # Get the number of reallocations
    size <- p7Pred() %>%
      select(fpa2, yXPred) %>%
      group_by(fpa2) %>%
      summarise(total = sum(yXPred))
    #print(paste("size:",size))

    pref1List <- list()
```

```
    for(i in 1:length(simClosed())){

      upns <- prefs[prefs$school %in% simClosed()[i] & prefs$pref == 2 &
nchar(as.character(prefs$UPN)) > 5,3]
      #upns <- prefs[prefs$school %in% c(2420054) & prefs$pref == 2,3]

      firstPrefs <- prefs %>%
        filter(UPN %in% upns, pref == 1) %>%
        inner_join(pupils16[,c(3,12)], by = "UPN") %>%
        select(school) %>%
        table() %>%
        as.data.frame() %>%
        filter(!. %in% simClosed()) %>%
        mutate(fpa = as.numeric(as.character(.))) %>%
        select(fpa, Freq) %>%
        filter(Freq != 0) %>%
        left_join(post_primary[,1:2], by = c("fpa" = "denino")) %>%
        arrange(desc(Freq)) %>%
        left_join(stated2Prefs, by = c("fpa" = "fpa")) %>%
        filter(!is.na(fpa)) %>%
        arrange(desc(Freq)) %>%
        left_join(size, by = c("fpa" = "fpa2"))

      ## total should be the total number of first preferences from [school] (estimated by year),
      # sum should be the number of recorded 2nd prefs where [school] is first pref, (across all years)
      # This is not intuitive (which may be another way of saying I'm not that samrt), so I'll spell
      # this out: for each school that is a source of 2nd preferences, the rescaled 2nd prefs should
      # add up to firstPrefs$total (number of FPAs for the year). Stated2 is the number of stated 2nd
      # prefs recorded in all the data we are looking at, so since we look at 3 year's of 2nd pref data
      # this is the denominator

      #print(head(firstPrefs))
      firstPrefs$Freq <- round(firstPrefs$Freq * firstPrefs$total/firstPrefs$stated2,1)
      pref1List <- bind_rows(pref1List, firstPrefs)
      #print(sum(firstPrefs$Freq, na.rm=T))
    }

    all1Prefs <- pref1List %>%
      group_by(fpa, schoolname, stated2) %>%
      summarise(Freq = sum(Freq)) %>%
      as.data.frame() %>%
      select(fpa, Freq, stated2, schoolname) %>%
      rename(base = stated2) %>%
      arrange(desc(Freq)) %>%
      filter(!is.na(fpa))

    DT::datatable(all1Prefs)
```

```
  })

  output$network <- renderPlot({

    # remove 0 values from edges:
    edges2 <- edges[edges$crossings > 0,]
    colnames(edges2) <- c("x","y","weight")

    schoolListNames <- schoolList[!is.na(schoolList$schoolname),]

    edges3 <- edges2[edges2$x %in% schoolListNames$fpa2 & edges2$y %in%
schoolListNames$fpa2,]

    vertices <- edges3 %>%
      filter_all(any_vars(. %in% simClosed())) %>%
      arrange(desc(weight)) %>%
      mutate(school = ifelse(x %in% simClosed(),y,x))
    remove <- vertices %>%
      anti_join(head(vertices,6))

    dirEdges2 <- dirEdges[dirEdges$directedCrossings > 0,]
    colnames(dirEdges2) <- c("x","y","weight")
    dirEdges3 <- dirEdges2[dirEdges2$x %in% schoolListNames$fpa2 & dirEdges2$y %in%
schoolListNames$fpa2,]

    # d describes the edges of the network. Its first two columns are the IDs of the source and the
target node for each edge. The following columns are edge attributes (weight, type, label, or
anything else).
    #vertices starts with a column of node IDs. Any following columns are interpreted as node
attributes.
    net <- graph_from_data_frame(d=dirEdges3, vertices=schoolListNames, directed=T)
    #net <- simplify(net, remove.multiple = F, remove.loops = T)

    node <- V(net)[name %in% simClosed()]

    dirV <- ego(net, order=1, nodes = node, mode = "all", mindist = 0)
    dirG <- induced_subgraph(net,unlist(dirV))

    removeVec <- as.character(intersect(get.data.frame(dirG, what= c("vertices")
)$name,remove$school))

    dirG <- dirG - removeVec

    plot(dirG, edge.label = edge_attr(dirG, "weight"), vertex.label = V(dirG)$schoolname,
edge.width=(E(dirG)$weight)/10,
        layout = layout_with_fr(dirG), edge.curved=seq(-0.5, 0.5, length = ecount(dirG)))

  }, width = 750, height = 750)
```

```
  output$forecast <- renderPlotly({

    # Actual FPAs in each year for selected schools:
    actual <- prefs %>%
      filter(pref == 1, school %in% simClosed()) %>%
      group_by(school, year) %>%
      summarise(Applications = n()) %>%
      as.data.frame() %>%
      mutate(school = as.numeric(as.character(school)))

    # pupils at each primary school, subsequent predictions and joined to actual
    if(input$geo == "Primary School"){
      forecastSummary <- pupils16 %>%
        mutate(year = ifelse(ccyear==7,17,
                      ifelse(ccyear==6,18,
                        ifelse(ccyear==5,19,
                          ifelse(ccyear==4,20,
                            ifelse(ccyear==3,21,
                              ifelse(ccyear==2,22,
                                ifelse(ccyear == 1,23,-1)))))))) %>%
        filter(year != -1) %>%
        count(denino,gender,year) %>%
        as.data.frame() %>%
        #inner_join(matrixProb19[,c(2,3,4,7)], by = c("denino" = "DENI.no.", "gender" = "Gender"))
  %>%
        inner_join(matrixProb()[,c(2,3,4,7)], by = c("denino" = "DENI.no.", "gender" = "Gender")) %>%
        filter(fpa2 %in% simClosed()) %>%
        mutate(applications = percentOfSchool * n) %>%
        ungroup() %>%
        select(fpa2, year, applications) %>%
        rename(school = fpa2) %>%
        group_by(school,year) %>%
        summarise(Applications = sum(applications)) %>%
        as.data.frame()
      }

    if(input$geo == "SOA"){
      forecastSummary <- pupils16 %>%
        mutate(year = ifelse(ccyear==7,17,
                        ifelse(ccyear==6,18,
                          ifelse(ccyear==5,19,
                            ifelse(ccyear==4,20,
                              ifelse(ccyear==3,21,
                                ifelse(ccyear==2,22,
                                  ifelse(ccyear == 1,23,-1)))))))) %>%
        filter(year != -1) %>%
```

```
        count(lsoa11,gender,year) %>%
        as.data.frame() %>%
        #inner_join(matrixProb19[,c(2,3,4,7)], by = c("denino" = "DENI.no.", "gender" = "Gender"))
%>%
        inner_join(matrixProb()[,c(2,3,4,7)], by = c("lsoa11" = "DENI.no.", "gender" = "Gender")) %>%
        filter(fpa2 %in% simClosed()) %>%
        mutate(applications = percentOfSchool * n) %>%
        ungroup() %>%
        select(fpa2, year, applications) %>%
        rename(school = fpa2) %>%
        group_by(school,year) %>%
        summarise(Applications = sum(applications)) %>%
        as.data.frame()
    }

    actual$year <- ifelse(actual$year == 2017,17,ifelse(actual$year == 2018,18,19))
    #forecastPlot <- bind_rows(forecastSummary,actual)

    forecastSummary <- forecastSummary %>%
      left_join(post_primary[,c(1:2,23)], by = c("school" = "denino")) %>%
      rename(enroll1718 = X201718)

    actual <- actual %>%
      left_join(forecastSummary[,c(1,4,5)], by = "school")

    g <- ggplot(forecastSummary, aes(x=year,y=Applications, group = school, col = fct_reorder(.f =
schoolname, .x = enroll1718, .fun = max, .desc = T))) +
      geom_line() +
      theme_minimal() +
      expand_limits(x = 17, y = 0) +
      geom_hline(data = forecastSummary, aes(yintercept = enroll1718, col=schoolname), linetype =
"dashed") +
      labs(col = "School") +
      geom_point(data = actual,aes(x=year, y= Applications))

    ggplotly(g)

  })

 ### check that reallocations add up to original allocations ###

 # restore this code later to add in helpful summary

 #### end ####

}

# Run the application
```

```
shinyApp(ui = ui, server = server)
```