# Pupilinform on Power BI

## Proof-of-concept (POC)

### Contents

# Introduction

This report documents the findings of the proof-of-concept (POC) work for Gloucestershire County Council (GCC) on whether a replacement to Pupilinform – the pupil wellbeing survey analysis tool – can be built using Power BI. It aims to provide an initial understanding of whether it is possible to replicate Pupilinform on Power BI, and if so, identify the elements that are straightforward, those that are difficult, and those that may not be possible. It is not exhaustive, as there may be several ways in which to accomplish some elements, and it does not aim to fully duplicate all of the functionality of Pupilinform exactly. However, it should provide a sound basis on which to make decisions regarding the use of Power BI to replace the current version of Pupilinform, and provide some indication of the effort required to complete a first beta version.

# Data parsing

Power BI uses filtering ('slicing') and joins between data tables to undertake a lot of its data parsing via the user interface (UI). Additional processing can also be undertaken using calculated columns (simple by-row calculations) and measures (multi-row summarising functions). In contrast, R is a programming language with very flexible options and packages for all manner of data processing, manipulation, and UI development tasks. As a result, the data parsing Pupilinform performs in R is fairly convoluted to achieve in Power BI.

Several considerations need to be kept in mind:

- Filtering topics using slicers – in order to filter data, values need to be kept in a column. Thus, to filter questions by topic, the topics need to be kept in a column, and the questions relating to those topics are in another column.
- Linking data via joins – in order to link data (for example, question topics and questions to the survey data for those questions) the data tables need to be joined. Thus, the column of questions for a topic needs to be joined to a column of question identifiers for the survey data. This means that the survey data also has to be in long-format, with a column for the question identifiers and a column for the answer values. This makes the dataset for the survey answers very long – with millions of rows.
- Comparison of data sets – Pupilinform compares two versions of the data. For school users, this is a comparison between the data for the user's school and data for the county/district/etc. This means that two datasets must be sent to the plot. Typically, Power BI does not allow for two independent datasets to be sent to plots. Therefore, the two datasets must be joined on the question identifiers, the answer values and any additional variables required for the plot must be included somehow.
- Cross-tabulation and miscellaneous filtering – the two datasets need to be able to be cross-tabulated by common variables (sex, year group, etc.) and also filtered (by sex, phase, etc.) Thus, an additional table is required and needs to be joined to the long-format survey

answers. The user needs to be able to select a cross-tabulating variable and the results for the two versions of the data should then be broken down by the values for this variable in a single results table. Any filters need to be applied to both versions of the survey data using a single control, rather than one control for each version (e.g. school/comparator) of the data.

Methods used to deal with these considerations in the POC work include:

- Using single variables for joins – where tables are joined on multiple columns, these are combined into a single column first before joining (e.g. the question identifier and value is combined in a calculated column and then used to join tables).
- Using many-to-many joins – these are used so that any filters applied to a particular table will flow through to joined tables, controlling what data is retained in the final analysis.
- Encoding the data to save memory – the survey answer values are generally text categories. In order to minimise the amount of memory required, these have been encoded into numeric values. Thus, additional tables are required to lookup text values from the encoded values for presentation.
- Counting unique IDs for respondents – there is a risk with complicated joins and long-format data that counts of answer values are duplicated or counted incorrectly. To ensure this does not happen, the counts of answers are based on distinct counts of the respondents' unique ID numbers. A single response to the survey (a row of data in the original wide-format dataset) has a single, unique ID value.
- Switching cross-tabulation variables – an experimental Power BI feature called 'field parameters' is used to switch between different cross-tabulating variables. A copy of all the variables used for cross-tabulation is kept in a wide-format table and added to a field parameter. An on-screen control is then used to choose the variable that is being used to cross-tabulate.

## Preparing data for the Summary plot page (with tartan rug plots)

The Summary plot tab in Pupilinform is based on binary versions of many different survey questions. These are grouped into topics and then used as the basis of tartan rug plots which include statistical significance testing of the difference between a main analysis (e.g. school) and a comparator analysis (e.g. district).

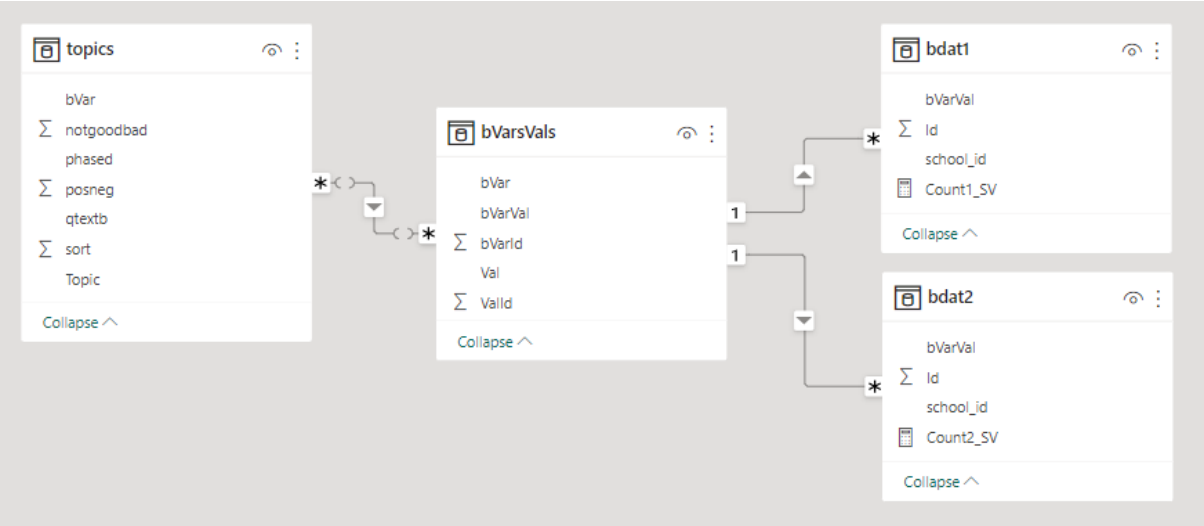
### *Linking topics to binary variables to survey answers*

Topics are contained within a table (topics) and joined to a table of binary variables and answer values (bVarsVals) using a column of binary variable names (bVar). Thus, by slicing on the topic, the table of binary variables and values is filtered. This join is many-to-many, as a binary variable can exist in more than one topic, and the binary variables table (bVarsVals) lists the values for each binary variable in long-format. Because the variables are all intended to be binary, they all have the same possible answer values (Yes, No, and Not answered). In the bVarsVals table, the questions and answer values have all been encoded into a numeric value. The binary variables are encoded into an

integer and the values are encoded into an integer. These are then combined into a decimal number, with the binary variable encoding before the decimal point and the value encoding after the decimal point. The purpose of this encoding is to minimise the size of the survey answer table.

The bVarVals table is joined to two copies of the binary survey answer table with a one-to-many join. These tables (bdat1 and bdat2) contain the encoded values of the questions/answers and an Id column which uniquely identifies the respondent. This table is in long-format, with each row representing a respondent-question-answer combination; in the original, wide-format survey data (a copy of which is included in the Appendices), each Id represents a row of data. As a result of these relationships, when a topic is selected via a slicer, the related variables and respondents' answers are filtered. These relationships are represented in Power BI as shown in the diagram in Figure 1.

Figure 1: Table relationships for filtering binary variables by Summary topic



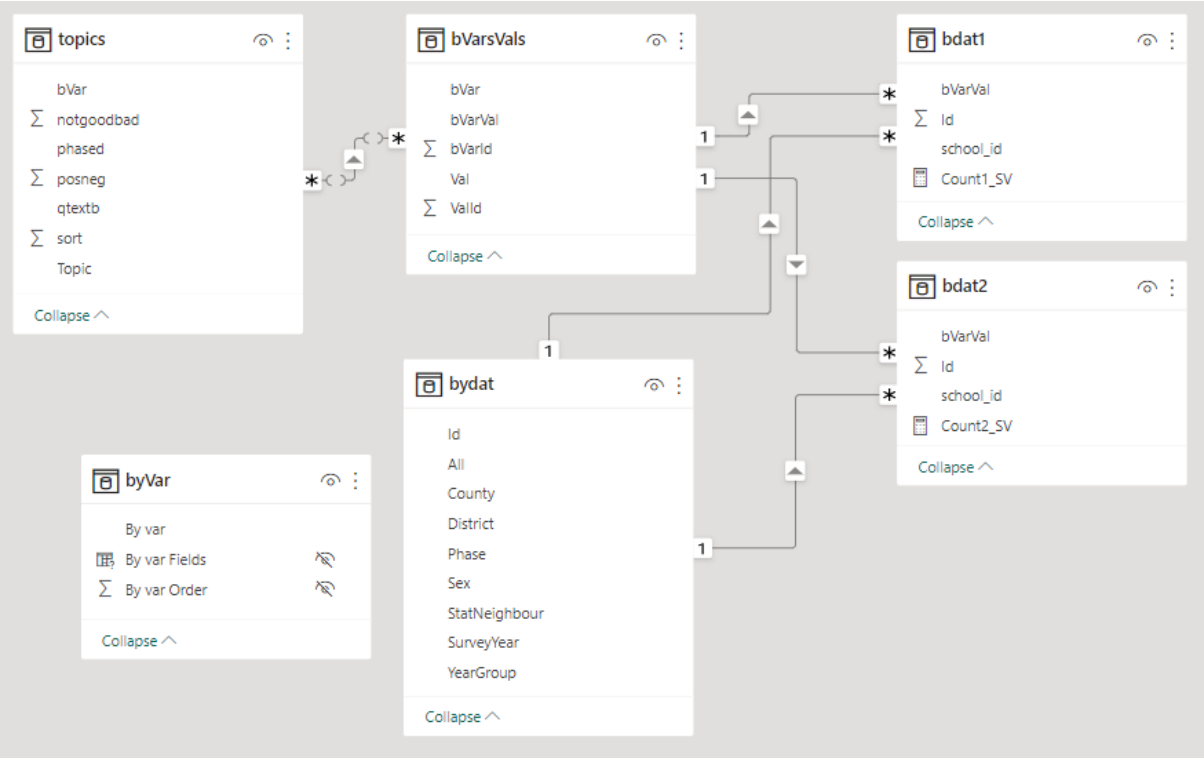

### *Incorporating the 'plot by' variable and additional filtering of the data*

In order to cross-tabulate the outputs by an additional variable (e.g. sex, phase, year group, etc.), this variable needs to be joined to the answer values somehow. It also needs to be easy to change using a slicer. One method of doing this would be to create another long-format table of variables that are to be used for cross-tabulation and join them to the answer value tables (bdat1 and bdat2) by the respondent Id. Instead, in the POC work, we have made use of a currently experimental functionality of Power BI that uses 'field parameters' to allow switching of a variable using a slicer. A wide-format table of variables that are to be used for cross-tabulation (bydat), also containing the respondent Id column, are joined to the long-format answer value tables via the respondent Id using a one-to-many join. A field parameter is added that contains the names of the variables that are to be used for cross-tabulation. A slicer containing this field parameter can then be used to pick a cross-tabulation variable, which then causes the answer values (from bdat1 and bdat2) to be broken down by any values of the field parameter included in an output.

The advantage of this approach is that any additional filtering (e.g. to select a specific sex, phase, survey year, etc.) can also be achieved using standard slicers based on variables in this table. Thus,

the two answer value tables can be filtered and broken down by any variable in this table. The diagram in Figure 2 shows how these relationships are represented in Power BI, along with the field parameter table (byVar).

Figure 2: Table relationships for additional filtering and cross-tabulating results using a 'plot by' variable




### *Automatic filtering for school users and comparator choice*

When a school user logs in to Pupilinform the data for the main analysis is automatically filtered to include only respondents for their school. Pupilinform also allows school users to choose a comparator analysis by name, rather than value. For example, if a school is based in Gloucester, they could choose District as the comparator, and Pupilinform would identify the school's district and filter the comparator data to that district. This functionality can be achieved in Power BI but is somewhat complicated. There are several steps to achieving the desired functionality:

1. the logged-in user needs to be identified,
2. their school and the school's details need to be retrieved,
3. the data for the main analysis needs to be filtered to that school,
4. the choice of comparator type needs to be selected, and
5. the data for the comparator analysis needs to be filtered according to the choice and matching value for the logged-in user's school.

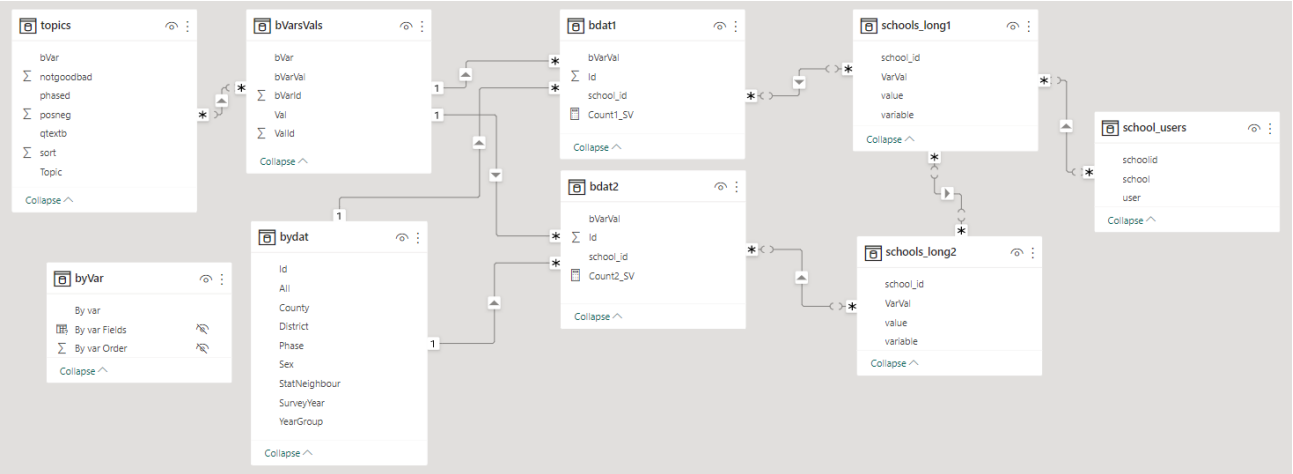
Filtering the data for the main analysis by the logged-in user is achieved in Power BI by using 'row-level security' (RLS). A table of school users (school_users) is stored which contains their email addresses and their 'school_id' numbers. When a user logs in to Power BI, their user group is identified and filtering rules are applied based on the group returned. In the POC, we have added an

example ‘school_users’ group which has users listed against a specific school. When viewing the report as a member of this group, Power BI filters the school_users table by the logged in user’s email address (which is retrieved using the USERPRINCIPALNAME() DAX function).

This table is joined to a table (schools_long1), via the school_id, which contains information on the schools, such as their default phase, statistical neighbour group, district, etc. This table is in long-format and contains a variable and a value column, which is then combined to create a column (VarVal) to be used for joining. When a school user logs in, the school_users table is filtered by row-level security, which causes the schools_long1 table to be filtered to the matching school_id. This table is then joined to the data for the main analysis (bdat1) by the school_id, causing the main analysis to only show data for the user’s school.

Achieving the comparator filtering requires an additional copy of the school table (schools_long2) to be added and joined to the schools_long1 table. In this case, the join is made on the VarVal column, and this table is then joined to the data for the comparator analysis (bdat2) using the school_id. The end result of this rather convoluted process is that along with the main analysis being filtered by the logged-in user’s school, a slicer based on the variable column of the schools_long1 table can be used to filter the schools_long2 table. Due to the join between the school tables, the schools_long2 table will be restricted to the same variable and value combination as the schools_long1 table, which is already filtered to the logged-in user’s school. Hence, if the user chooses ‘District’ as the comparator from the slicer, the schools_long2 table (and hence bdat2) will be filtered to the same district as the user’s school. The diagram in Figure 3 shows how these relationships are represented in Power BI.

Figure 3: Table relationships for automatically filtering by school user and selected comparator type



### *Phase-specific analyses*

There are a set of Ofsted-related topics which are phase-specific – i.e. the data should be filtered by a single phase when using them. The easiest way in which to deal with this additional complexity would be to simply highlight that a phase filter should be applied when using these topics. This is similar to the approach used in Pupilinform; however, Pupilinform also prevents analysis without the data being appropriately filtered. The simplest way in Power BI to highlight that a phase filter should be applied is to add a message column to the topics table. When a topic is picked using the topic slicer, a card visual is used to display any message relating to that topic. Enforcing the filtering of the data would likely be much more complex, and no simple way to do this in Power BI was identified for table visuals. It would likely involve adding additional rows to the topics table, one for each topic-bvar-phase combination, and then joining on both bvar (to bdat1 and bdat2) and an additional phase column. However, it would be fairly trivial to incorporate this restriction into the code for the R script visuals. By simply identifying the phases in any Ofsted topic and comparing them to the phases in the data passed to R, the plot could be drawn or not (with a message sent to the user via the plot title if phase filtering is needed, for example).

**Limitation:** Enforcement of single-phase filtering for specific topics in standard Power BI visuals is complex. However, this can be done fairly easily in R script visuals.

## Preparing data for the Custom plot page

The Custom plot tab in Pupilinform allows users to plot bar charts of any of the survey questions. The plots overlay filled and unfilled bar charts so that comparisons can be made between a main and comparator analysis. Unlike with the Summary plot tab, there is no statistical testing. The main challenge with this functionality is the variety of bar charts that are produced by Pupilinform. Some survey questions incorporate sub-questions, and so are grouped when presented. Some are categorical and some are binary, and these are presented differently by Pupilinform.

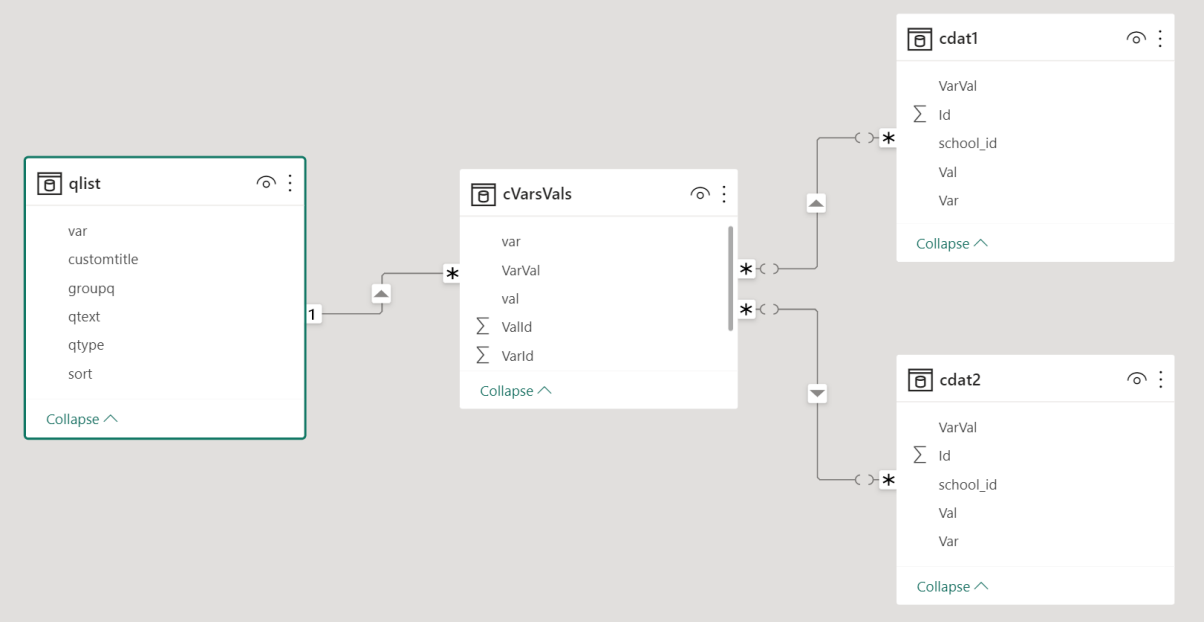
### *Linking survey questions to variables*

The survey questions are contained in a table (qlist) and joined to a table of variables and values (cVarsVals), in a similar fashion as to how the data for the Summary plot page is prepared (joining one-to-many on the variable name, var). The main differences are that rather than grouping variables into topics, the qlist table has repeating customtitle values, and rather than being binary, the values relate to the original answer options in the survey. The customtitle column represents the top-level questions in the survey, with some variables relating to sub-questions which are grouped by these top-level questions. In the Custom plot page, it is the customtitle column which is used to populate a slicer for choosing which question to analyse. As with data preparation for the Summary plot page, these tables are long-format and numerically encoded.

Similarly, the cVarsVals table is joined to two long-format copies of the survey answer table with one-to-many joins. These tables are named cdat1 and cdat2 to differentiate them from the binary data

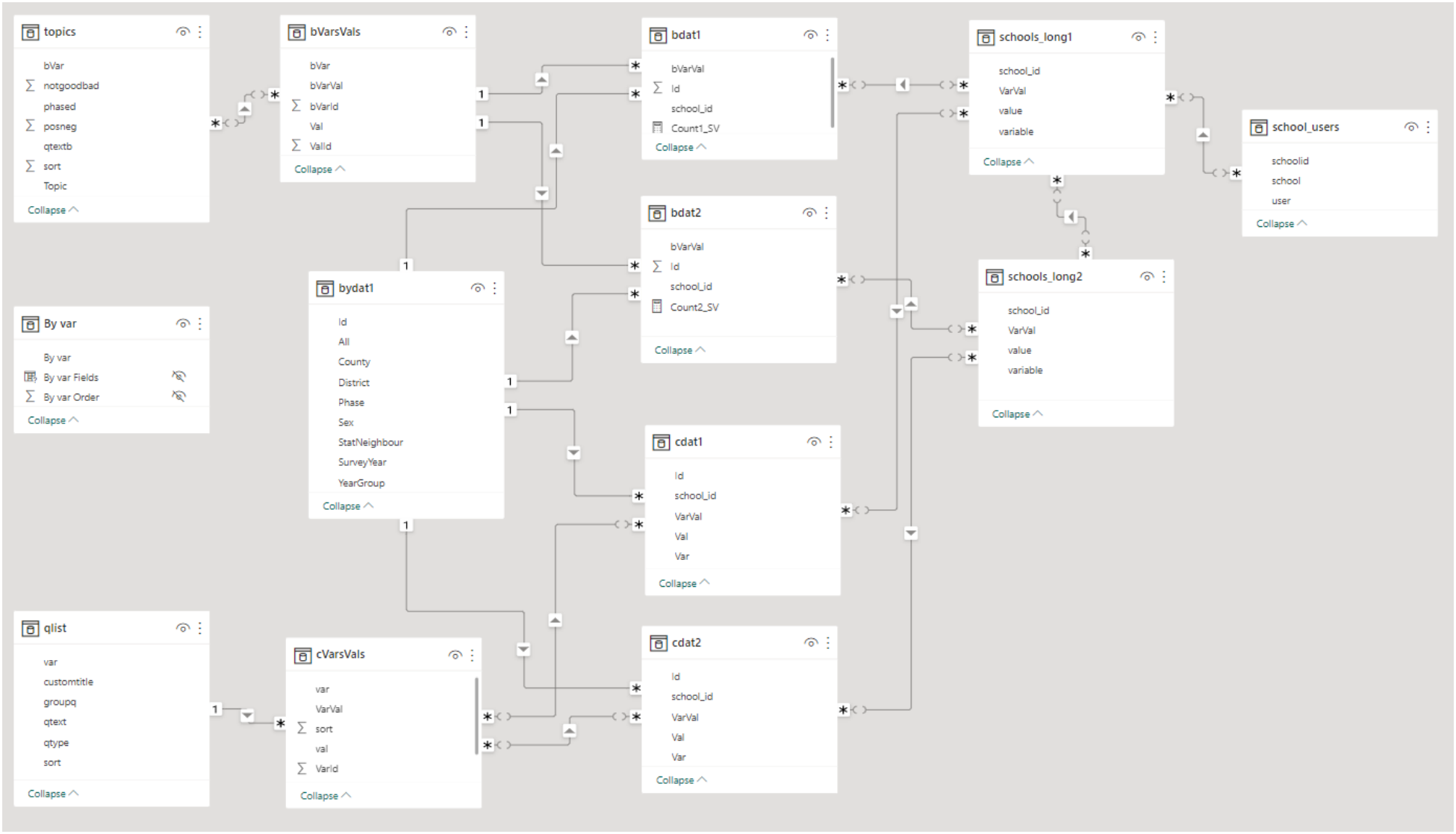
contained in bdat1 and bdat2. The similarity between the data preparation for the Custom plot page and the Summary plot page can be seen in Figure 4.

Figure 4: Table relationships for filtering variables by top-level question for Custom plots




In all other respects, the data preparation for the Custom plot page is identical to that for the Summary plot page. The cdat1 and cdat2 tables are joined to the bydat table, which is used for filtering and cross-tabulation of the data. The cdat1 and cdat2 tables are also joined to the schools_long1 and schools_long2 tables, the first of which is also joined to the school_users table, ensuring school users see only their school's data in the main analysis, and enabling choice of comparator type. The resulting relationships (for both the Summary plot and Custom plot pages) are shown in Figure 5, on the next page.

Figure 5: All table relationships required for school users



## User roles

In Pupilinform, the user logs in and their role is looked up in a data table. The software then adjusts the controls, main analysis filtering, and whether small number suppression is imposed, based on the role. The easiest way to replicate this kind of functionality in Power BI is to simply have different versions of the dashboard for the different roles:

- School user version – the main analysis is filtered to the user's school by row-level security and suppression is enforced. The analysis always compares the user's school to a choice of comparator areas based on the school's district, statistical neighbour, etc.
- GCC/Partner user version – the main analysis can be filtered to any district, statistical neighbour, etc., as can the comparator analysis. Suppression is enforced. GCC users cannot filter analyses to view a particular school.
- Super GCC user version – the main analysis can be filtered in any way desired, as can the comparator analysis. Results are not suppressed.

The different roles necessitate slightly different slicers added to the reports for each role, and slight variations in the relationships and tables. For example, the school_users table and row-level security are not required for GCC and Super GCC roles. Likewise, the relationship between the schools_long1 and schools_long2 table is not required, allowing more flexibility in comparisons between areas; one district compared to another, for example. For the GCC users, they are not allowed to select individual schools for analysis, whereas the Super GCC role can.

A key limitation with this role-based approach is that – for Super GCC users – it does not incorporate the ability to switch suppression on and off, which can be done in Pupilinform with an on-screen control. It would be possible to add this to the plots, by adding a flag value to the dataset passed to the R script visual. The R code can then use this flag to determine whether small numbers are suppressed or not. The flag would be stored in a two-row table with on/off in one column and a dummy joining variable in another column (e.g. a column of TRUE values). The suppression slicer would essentially have no effect except to change the control value (on/off) included in the dataset passed to the R script visual.

It's possible that this switching approach could be applied to Power BI data tables used for data downloads to suppress values smaller than five (by using a DAX IF statement to determine whether suppression is applied). However, we could not replicate all the suppression rules used in Pupilinform. Suppression is discussed in more detail in the section on Suppressing values and data downloads.

# User authentication

Depending on the licensing and setup of Power BI and related software in Gloucestershire County Council, the way in which users can access Power BI varies. Going into the detail of this is beyond the scope of this POC work but could have a significant impact on costs and the complexity of managing users. Currently, Pupilinform has over 700 registered users (although many of these have not accessed or do not frequently use it). If users can be added to a managed authorization service (such as Microsoft Azure Active Directory) using GCC's current provision, no extra cost may be incurred. However, if each user requires a Pro or Premium license to access Power BI, the cost may be significant. More information is available on how to share Power BI reports with users from outside organisations here: https://learn.microsoft.com/en-us/power-bi/collaborate-share/service-share-dashboards

**Limitation:** If external users require a Power BI Pro or Premium licence to access the reports, the total cost could be considerable (for continuous access for all registered users).

# Plotting data

There are two main plots in Pupilinform: the tartan rug plots in the Summary plot tab, and the bar charts in the Custom plot tab. The tartan rug plots are fairly complex due to the incorporation of statistical testing, whereas the bar charts are relatively simple, with most of the complexity related to the way in which sub-questions can be grouped and plotted together, and the reorientation of the plot when binary data is displayed. In Power BI, there are no standard visualisations that can duplicate either of these plots. As a result, it is necessary to use custom Power BI, R or Python script visuals. Seeing as Pupilinform is developed in R, R script visuals have been used in the POC work. In effect, data is passed to R, where the plot is constructed, and the resulting image is sent back to Power BI to be displayed.

**Limitation:** No standard Power BI visualisations that can display the plots used in Pupilinform.

## Tartan rug plotting

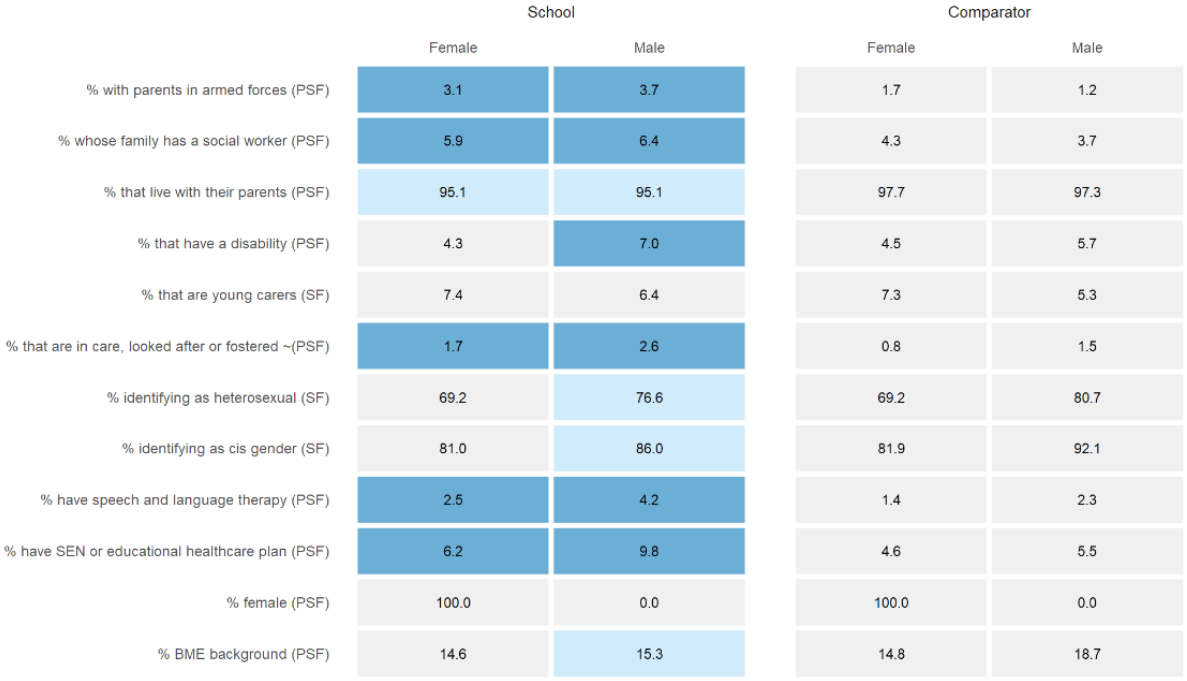
The data passed to R to create the tartan rug plots is very simple and can also be displayed in the form of a table visual in Power BI. The selected topic and related binary variables and values (from the bVarsVals table), along with any cross-tabulation variable (from the bydat table) is used to tabulate distinct counts of the respondent Id from both bdat1 and bdat2. If a school user is logged in, then the bdat1 table will be automatically filtered to their school's data. Any additional information can also be passed to R, such as the comparator choice, any filter values (based on bydat) and any auxiliary information about the questions in the topics (stored in the topics table).

The R script is essentially a copy of the code used in Pupilinform to perform the same tasks:

- the tabulated data is processed to conduct statistical tests and calculate percentages,
- hex colour codes are added depending on the results of the tests,
- results based on low numbers are suppressed, and
- the plot is constructed and displayed.

The code in Pupilinform and the POC work is fairly complex and would require someone with the appropriate skills in R to understand and develop it further. The advantage of using R is that it is a very flexible programming language, and so there would be scope to develop the analysis and presentation if necessary. Two examples of tartan rug plots from the R script visual in Power BI are shown in Figure 6 and Figure 7. The comparison between the two raised a potential issue which may need addressing and is discussed in the section on Plot display issues.

Figure 6: First example – demographics Summary (tartan rug) plot in Power BI



| | School | | Comparator | |
|---|---|---|---|---|
| | Female | Male | Female | Male |
| % with parents in armed forces (PSF) | 3.1 | 3.7 | 1.7 | 1.2 |
| % whose family has a social worker (PSF) | 5.9 | 6.4 | 4.3 | 3.7 |
| % that live with their parents (PSF) | 95.1 | 95.1 | 97.7 | 97.3 |
| % that have a disability (PSF) | 4.3 | 7.0 | 4.5 | 5.7 |
| % that are young carers (SF) | 7.4 | 6.4 | 7.3 | 5.3 |
| % that are in care, looked after or fostered ~(PSF) | 1.7 | 2.6 | 0.8 | 1.5 |
| % identifying as heterosexual (SF) | 69.2 | 76.6 | 69.2 | 80.7 |
| % identifying as cis gender (SF) | 81.0 | 86.0 | 81.9 | 92.1 |
| % have speech and language therapy (PSF) | 2.5 | 4.2 | 1.4 | 2.3 |
| % have SEN or educational healthcare plan (PSF) | 6.2 | 9.8 | 4.6 | 5.5 |
| % female (PSF) | 100.0 | 0.0 | 100.0 | 0.0 |
| % BME background (PSF) | 14.6 | 15.3 | 14.8 | 18.7 |

Figure 7: Second example – physical activity Summary (tartan rug) plot in Power BI

| | School | | Comparator | |
|---|---|---|---|---|
| | Female | Male | Female | Male |
| % would like to do more exercise (SF) | 75.0 | 70.4 | 75.2 | 74.4 |
| % that actively travel to school (PSF) | 39.4 | 42.6 | 64.2 | 75.1 |
| % find it easy to be physically active (SF) | 51.5 | 65.4 | 52.5 | 67.8 |
| % feel they do enough exercise for health (PSF) | 57.3 | 66.9 | 48.7 | 66.2 |
| % enjoy PE lessons (PS) | 58.2 | 72.1 | 42.0 | 74.8 |
| % don't exercise because too expensive* (PSF) | 11.5 | 10.2 | 12.4 | 15.4 |
| % don't exercise because think it's not cool* (PSF) | 4.5 | 5.5 | 4.1 | 4.6 |
| % don't exercise because not good at it* (PSF) | 31.5 | 28.9 | 34.4 | 28.8 |
| % don't exercise because no-one to do it with* (PSF) | 25.6 | 21.4 | 32.0 | 24.4 |
| % don't exercise because get hot and sweaty* (PSF) | 21.9 | 23.2 | 19.5 | 18.2 |
| % don't exercise because find it boring* (PSF) | 20.0 | 20.0 | 31.9 | 29.2 |
| % don't exercise because feel embarrassed* (PSF) | 29.7 | 15.7 | 41.3 | 18.0 |
| % don't exercise because facilities not nice* (PSF) | 4.8 | 4.9 | 7.0 | 6.3 |
| % don't exercise because don't like teacher* (PSF) | 5.2 | 4.8 | 10.1 | 8.0 |
| % don't exercise because don't have time* (PSF) | 29.8 | 24.2 | 34.2 | 27.8 |
| % don't exercise because don't enjoy it* (PSF) | 29.0 | 25.6 | 45.9 | 35.8 |
| % doing 6+ hours exercise per week (PSF) | 40.3 | 51.7 | 40.2 | 56.2 |
| % do exercise to see friends (PSF) | 54.1 | 58.3 | 53.9 | 57.9 |
| % do exercise to reduce stress (SF) | 50.7 | 48.4 | 52.5 | 48.7 |
| % do exercise to learn a new skill (PSF) | 49.6 | 57.3 | 34.2 | 41.8 |
| % do exercise to get fit (PSF) | 74.3 | 76.0 | 69.4 | 73.3 |
| % do exercise to get better at it (PSF) | 58.4 | 67.1 | 45.1 | 59.7 |
| % do exercise for weight management (SF) | 54.4 | 49.8 | 55.3 | 51.7 |
| % do exercise for enjoyment (PSF) | 72.5 | 74.9 | 68.3 | 73.7 |
| % do exercise because they have to (PSF) | 45.3 | 50.0 | 32.8 | 39.1 |

## Custom plotting

The data passed to R to create the Custom plots is very similar to that used in the tartan rug plots. The main difference is that the data can have many more categories than the binary data used for the tartan rug plots and is grouped according to the top level question (with potentially many sub-questions). This data can also be displayed as a table in Power BI in order to represent what is passed to the R script visual. The selected top-level question (customtitle in the qlist table) and related variables and values (from the cVarsVals table), along with any cross-tabulation variable (from the bydat table) is used to tabulate distinct counts of the respondent Id from both cdat1 and cdat2. If a school user is logged in, then the cdat1 table will be automatically filtered to their school's data. Any additional information can also be passed to R, such as the sub-question text (qtext), comparator choice, any filter values (based on bydat) and any auxiliary information about the questions stored in the qlist table (such as the question type – categorical or binary – stored in the qtype column).

The R script is essentially a copy of the code used in Pupilinform to perform the same tasks:

- the tabulated data is processed to calculate percentages,
- hex colour codes are added depending on the analysis (main or comparator),
- results based on low numbers are suppressed,
- label formats are created dependent on the label lengths and number of categories,
- the data and plot are adjusted if the variables are binary, and
- the plot is constructed and displayed.

As with the tartan rug plot, the code in Pupilinform and the POC work is fairly complex and would require someone with the appropriate skills in R to understand and develop it further. Two examples of Custom plots from the R script visual in Power BI are shown in Figure 8 and Figure 9. Again, there are potential issues with the plots which may need addressing (discussed in the section on Plot display issues).

Figure 8: First example - Custom plot with binary variables

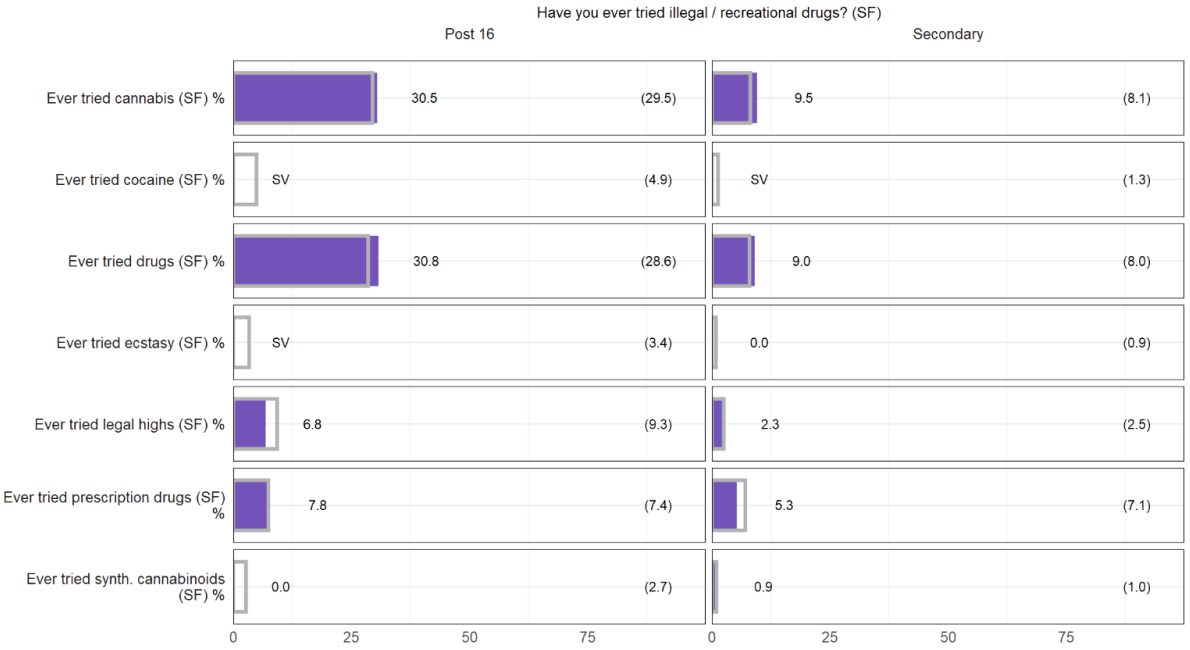
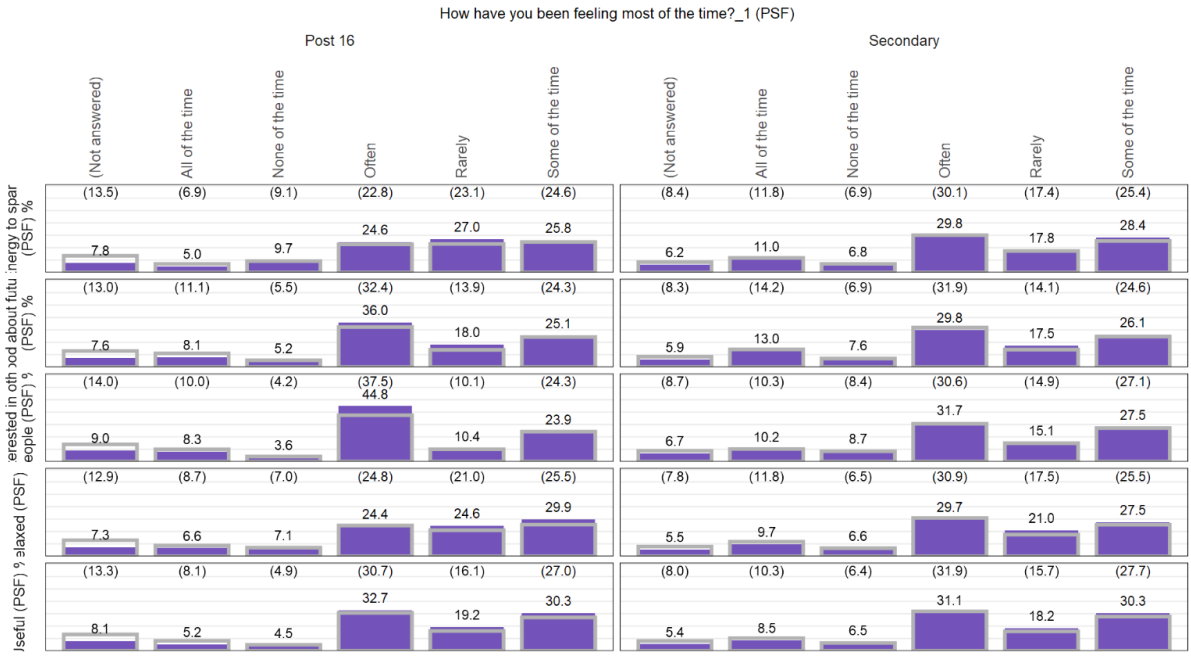



Figure 9: Second example - Custom plot with categorical variables



## Plot display issues

There were three issues identified with plot display during the POC work:

- plot sizing and aspect ratio,
- plot processing speed, and
- value category sorting.

### *Plot sizing and aspect ratio*

As can be seen in the examples of plots above, depending on the amount of data being displayed, the plots can appear squashed or stretched. This is because the plot size is determined and fixed by the embedded R script visual in Power BI. Essentially, any plot drawn in R will be sized to fit the available R script visual frame. This is not a huge problem for the tartan rug plots, as they cope fairly well with being squashed or stretched to a set frame, with only an aesthetic impact. For the Custom plots, however, the impact is much greater. This is because the Custom plots can vary considerably in the size of canvas required to display them clearly. This is due to the potentially high number of categories and sub-questions that may need to be displayed. Also, if a cross-tabulation with many categories is plotted (such as with year group or survey year), this problem will be exacerbated.

There are two potential solutions to this problem:

- limit the amount of data that can be plotted, or
- adjust the returned plots and Power BI page size to adequately display all plots.

The first of these solutions amounts to ensuring that the plot looks acceptable simply by limiting the number of plot rows and columns displayed. For tartan rug plots, this can be accomplished by changing the number of variables in each topic. Large topics could be split into groups, for example. For the Custom plots, this amounts to changing the number of sub-questions that are grouped together for any top-level question. At the moment, up to five categorical sub-questions are displayed within any grouped top-level question (hence, the suffix numbers that appear for some of the top-level questions in the customtitle column). This number could be reduced. Although fairly straightforward, this solution may not be ideal.

The alternative is to increase the Power BI report page size and the frame of the R script visual, and then use R code in the R script visual to amend the amount of the canvas used to display the plot based on the amount of data being plotted. Pages in Power BI reports have a default aspect ratio of 16:9, but can be changed to be portrait and have custom dimensions (set in pixels). This custom size, along with setting the page to view full-width, allows long pages to display online with vertical scroll bars. The R code would need to be amended to create two ‘plots’ of varying heights. One of these would contain (for example) the Custom plot, and one would be blank. The two plots would be combined vertically before being drawn. By setting the relative height of each, the vertical size of the Custom plot could be controlled.

**Limitation:** R plots fill the R script visual frame in Power BI, which is of fixed size, squashing large plots and stretching small plots (but there are ways to work around this).

### *Plot processing speed*

Typically, standard Power BI visuals respond to changes in slicer values very quickly. This is partly a reflection of the simplicity of most of the visuals and the way in which they are integrated into the software. R script visuals are non-native and involve passing data back and forth between R and Power BI. As a result, there is an additional lag between slicer values changing and the resulting plot appearing in Power BI. In the POC tests using Power BI desktop, this lag was approximately five seconds for new plots (most Power BI visuals usually update in less than one second). Interestingly, however, the nearest standard Power BI plot to the Custom plots used in Pupilinform took approximately 15 seconds to update. This suggests that the speed of processing may be related both to the passing of data between R and Power BI and the complexity of the plots themselves. Nevertheless, the speed at which plots in R script visuals update is slower in Power BI than in Pupilinform (approximately one second). Plots which have been drawn before (during the session) refresh almost instantly in Power BI desktop. This is because newly drawn plots are being cached by Power BI and are simply retrieved if the same slicer settings are chosen later in the session. Performance was similar in tests using the Power BI service, online.

**Limitation:** Plots take a few seconds to be newly drawn after a slicer value is changed. Due to plot caching, plots which have been drawn before can update almost instantly.

### *Value category sorting*

As can be seen in the plot examples above, category values are sorted alphabetically when the data passed to R is text data. This means that the values in the second Custom plot are not in the correct order. In Pupilinform, the categories are stored as factor variables (the underlying data is ordered and numeric, with value labels attached to each numeric category). Power BI does not have an equivalent data type, and so the data must be stored as numeric values or text strings. There is a way in which to sort text values non-alphabetically in Power BI standard visuals (using additional numeric sort columns), but seeing as the data is passed to R, the solution to the problem needs to be programmed in R. The way in which this can be done is to pass not only the text values, but also their encoded numeric values to R. The text string columns in the dataset can then be coerced into factors in R, using the numeric encodings. This can be accomplished with both the category values and the questions, where a particular ordering of questions is desired (such as in the Summary plot topics).

**Limitation:** Value categories based on text data are sorted alphabetically in R. These columns need to be turned into factor variables within the R script visual using the encoded numeric values, so that the preferred sort order can preserved.

# Suppressing values and data downloads

In Pupilinform, small counts in plots and data downloads are routinely suppressed for all user roles except for the Super GCC role. The aim of this suppression is to protect the anonymity of respondents. Any counts or statistics based on fewer than five respondents are hidden. Additionally, where it might be possible to back-calculate a small count from presented results (e.g. if a single category has been suppressed in an analysis of a categorical variable), an additional category is suppressed (usually the 'Not answered' category or the category with the next most frequent count).

Seeing as the plots are drawn in R, the suppression of small counts can be achieved using the same approach as in Pupilinform for the plots. However, there are some difficulties in achieving adequate suppression in Power BI table visuals and data downloads, and related data security considerations for the plots:

- Power BI 'export data' functionality usually allows data passed to a visual to be downloaded,
- R script visuals cannot be used to create tables (only images), and
- although simple suppression can be achieved in Power BI table visuals, complex suppression rules are difficult to programme (in Data Analysis Expressions – DAX).

## Power BI 'export data' functionality

Typically, with any Power BI visual, there are header icons in the top right corner where a user can choose to export the data on which the visual is built. In the case of an R script visual, this will be the dataset prior to any processing by R, and thus small counts will not be suppressed (the full dataset is required to calculate percentages and conduct statistical tests). Thus, it is necessary to prevent users from being able to download this data (except for Super GCC users). For visuals, this is done by turning off the header icons for the visual. Additionally, there are controls at the report level where the ability to query the underlying data source can be controlled and all data export prevented. This means that although the plots will be adequately suppressed, there is no possibility of using this functionality for users to download adequately suppressed data.

## R script visual tables

A logical alternative might be to suppress counts in R and pass data tables to Power BI for download by the user. However, R script visuals can only display images and so any table presented would be a graphic rather than a data table that can be downloaded and used for other purposes.

## Power BI table visuals

The remaining option is to use Power BI table visuals to build the suppressed data tables. The POC work includes tables that show the data which is sent to the R script visuals. Adding suppression of small counts is easy to achieve. However, during the POC work, we found that more complex suppression rules are difficult to programme in the DAX language used by Power BI. This is because

DAX works using ‘row contexts’ and ‘filter contexts’. The suppression rules used by Pupilinform necessarily work across different DAX contexts, making them fairly difficult to translate into DAX. As an example, the steps involved in suppression of a categorical variable in Pupilinform are:

1. where the category is substantive (i.e. not including ‘Not answered’) suppress small counts which are greater than zero and less than five,
2. count how many categories are suppressed,
3. if only one category is suppressed, identify any zero count categories or low count ‘Not answered’ category and suppress them too, and
4. if there were no zero or low count ‘Not answered’ categories, rank the categories from smallest to largest count and suppress the next largest count.

This method was designed in an attempt to retain as much substantive information in an analysis as possible, whilst protecting respondents’ identities.

Ultimately, we were unable to replicate the suppression rules using DAX in Power BI. This is not to say that it is not possible, however. An expert DAX consultant could be employed in an attempt to find a method by which to produce an adequate suppression approach using Power BI table visuals. These could then be used as the method by which suppressed data could be downloaded by users.

**Limitation:** We could not reproduce the suppression rules used in Pupilinform using DAX in a Power BI table. Nevertheless, an expert DAX consultant may be able to programme an adequate solution.

## Report downloads

The report downloading functionality of Pupilinform allows users to download a copy of the analysis they have just run as a single page and, for the tartan rug plots, also a multi-page Word report of the analysis using the same settings but re-running the analysis for all non-Ofsted topics.

In Power BI, there is PDF export functionality. This usually allows the user to save a PDF document of the current page or all pages of a report. However, a limitation of this export functionality is that it currently does not support exporting R or Python script visuals. This also means that any discussion of multi-page (all topic) reports in Power BI is moot (although there is an additional paginated report designer available for Power BI that may offer suitable functionality for standard Power BI visuals).

The only simple and satisfactory way in which R script visuals can be downloaded from the Power BI service is by screen capture. This would mean a user would need to be comfortable either using the print screen key on their keyboard or using something like the snipping tool in Windows. It is possible to print R script visuals to PDF using Power BI functionality, but we found that the results were not predictable (size and position of visual on the page was often not ideal). This lack of adequate support for exporting R script visuals in Power BI is a major limitation.

**Limitation:** R script visuals cannot be exported to PDF using Power BI functionality. They can be printed to PDF, but we found results to be unpredictable. The only reliable way to export R script visuals is to use screen capture functionality in Windows. There is no automated way to create report downloads in Word.

# Miscellaneous

There are a few miscellaneous aspects of Pupilinform that don't fit neatly within the core functionality of the software. This section aims to cover these for the sake of completeness.

## Mean WEMWBS score

In Pupilinform, the mean WEMWBS score is included in the Summary plot tab, despite not being calculated in the same way as all other variables and having a different statistical test applied. In Pupilinform, the calculations are performed entirely separately and then added to the main results. We could not identify a way to do this in Power BI that would not be immensely complicated.

**Limitation:** We could not identify a way to add the mean WEMWBS score to the data passed to the R script visual for the tartan rug plot or display it in a Power BI table visual alongside counts.

## Keys to plots

Keys to plots in Pupilinform describe what the plot colours and various tags to questions mean (phase-specific questions, routing, wording changes, etc.) It would be easy to add these as graphics to Power BI. Alternatively, there is a custom visual called 'HTML content' that enables the loading of HTML code, which could be used to build the keys.

## Guidance and glossary

The guidance and glossary tabs in Pupilinform could also be replicated fairly easily using a combination of text boxes and graphics.

## Routing info

The routing info would need to be added to Power BI as an additional data source, and then a Power BI table visual could be used to display the information.

## Webinars

Adding the webinars is also possible. Video can be embedded into Power BI reports and displayed by the Power BI service. Currently, the webinars in Pupilinform are hosted on YouTube. However, it seems that YouTube blocks access to videos requested by the Power BI service. As a result,

alternative hosting would be required, such as on Vimeo or a custom server/storage (such as Azure). A possible approach is described here: https://www.discoverei.com/blog/how-to-embed-videos-in-power-bi

## Summary and conclusions

The POC work was challenging and resulted in both good solutions and limitations in transferring Pupilinform to Power BI. Much of the core functionality of Power BI is possible to replicate using Power BI, but there are some key gaps and issues that need to be considered.

Core functionality that **could be incorporated** included:

1. controlling filtering and cross-tabulation of data using table joins and slicers,
2. producing the correct data summarisations and passing them to visuals so that plots could be constructed,
3. controlling the data that users can pass to the visuals, using row-level security, slicer values, and role-specific versions of the reports, and
4. creating the same plots as available in Pupilinform using R script visuals.

Core functionality that **could *not* be incorporated** included:

1. the suppression of small counts (beyond simply hiding values <5) to prevent the possibility of back-calculation from the remaining values and totals in table visuals,
2. a straightforward method of creating downloadable reports that are correctly formatted, and
3. incorporation of the mean WEMWBS score in the Summary plots.

For the suppression of small counts, it may be worth seeing whether a DAX expert may be able to incorporate improved suppression rules into the table visuals used for data download.

There were also some **limitations it is possible to work around**:

1. enforced phase filtering for specific (Ofsted) Summary topics,
2. squashing and stretching of plots dependent on the amount of data being displayed, and
3. correct sorting of questions and values in visuals and tables.

There was also **one other important consideration** that relates to:

1. the way in which users access Power BI from external organisations and any costs associated with this. Power BI subscriptions may need to be paid unless GCC can add external users to its organisational Power BI user groups.

Ultimately, the POC work has been very successful in prototyping solutions for moving Pupilinform onto the Power BI platform. GCC will need to carefully consider the limitations that have been identified. It may be necessary for GCC to undertake additional work on these before a final decision can be made and work can begin on building a Power BI-based solution for release to users.

# Appendices

The ‘Appendices’ folder included with the POC work contains the following:

1) *Power BI reports folder – dashboard/report files for each user role:*
   a) **dev_school.pbix** – the Power BI report for POC work for School users of Pupilinform,
   b) **dev_gcc.pbix** – the Power BI report for POC work for GCC/Partner users of Pupilinform, and
   c) **dev_supergcc.pbix** – the Power BI report for POC work for Super GCC users of Pupilinform.
2) *Data preparation folder – code, raw data and prepared datasets used in POC work:*
   a) **POC_data_preparation.Rproj** – RStudio project file for the data preparation work in R,
   *b) Data folder – raw data used in Pupilinform:*
      i) **dat.rds** – R data file containing GCC Pupil Wellbeing Survey data, used in Pupilinform,
      ii) **qgroups.csv** – CSV file of binary variables listed by Summary topic, used in Pupilinform,
      iii) **qlist.csv** – CSV file of questions and auxiliary information, used in Pupilinform, and
      iv) **schools.csv** – CSV file of schools and auxiliary information, used in Pupilinform.
   *c) R folder – R code used to prepare datasets for POC work:*
      i) **01_data_prep.R** – R script file used to prepare datasets needed for the POC work.
   *d) Outputs folder – prepared datasets resulting from running the R code:*
      i) **topics.csv** – lookup table of binary variables and auxiliary information listed by topic, used to filter data in the Summary plot page,
      ii) **bVarsVals.csv** – the lookup table of binary variables and values to their encoded values in CSV format,
      iii) **bdat_long.rds** – the encoded survey response data for the binary variables used in Summary topics, in long-format, saved in R data file format,
      iv) **bydatwide.csv** – a wide-format table of survey responses for variables used for ad hoc filtering or cross-tabulation (i.e. ‘plot by’ variables),
      v) **schools_long.csv** – the auxiliary information for schools in long-format for school users’ comparator choice and filtering of main/comparator analyses by GCC/Partner users,
      vi) **qlist_c.csv** – the lookup table of survey questions and auxiliary information by top-level question, used to filter data in the Custom plot page,
      vii) **cVarsVals.csv** – the lookup table of variables and values to their encoded values in CSV format,
      viii) **cdat_long.rds** – the encoded survey response data for the variables used to create Custom plots, in long-format, saved in R data file format,
      ix) **schools_long_superGCC.csv** – the same as schools_long.csv, except also including values for schools listed by name, so that Super GCC users can filter by school, and
   e) **school_users.csv** – a manually created table of pretend school users, used to test school filtering by logged-in user (using Power BI row-level security, RLS).