**Workshop – Using ODK**

**30th July 2018**

**Peter Smyth (University of Manchester)**

[Peter.smyth@manchester.ac.uk](mailto:Peter.smyth@manchester.ac.uk)

Contents

[Section 1 3](#_Toc515367694)

[What are the eSurvey Options? 3](#_Toc515367695)

[SMS based surveys 3](#_Toc515367696)

[PC web based. 3](#_Toc515367697)

[Smartphone based 3](#_Toc515367698)

[Sections 2 4](#_Toc515367699)

[Overview of ODK 4](#_Toc515367700)

[Components and general workflow 4](#_Toc515367701)

[Things we are not going to cover 9](#_Toc515367702)

[Section 3 10](#_Toc515367703)

[Using XLSForm 10](#_Toc515367704)

[Creating forms with XLSForm 13](#_Toc515367705)

[A simple form 14](#_Toc515367706)

[A more complex form 19](#_Toc515367707)

[Section 4 23](#_Toc515367708)

[Demonstration of Collect 23](#_Toc515367709)

[Demonstration of Aggregate 23](#_Toc515367710)

[Section 5 24](#_Toc515367711)

[Comparison between CSV and JSON data formats 24](#_Toc515367712)

[CSV 24](#_Toc515367713)

[JSON 24](#_Toc515367714)

[Section 6 28](#_Toc515367715)

[Processing the JSON output 28](#_Toc515367716)

# Section 1

## What are the eSurvey Options?

eSurveys can take on many forms, some possibilities are given below.

SMS based surveys

These are essentially simple questions and responses sent to and received from

Probably more suited to customer satisfaction than serious research. Unless you have an automated process for collected, and sending the next question, this approach is not really very viable.

### PC web based.

These can come in two forms; A survey embedded in an existing web page, tends to be simple market research (related to the page) or quizzes. Quizzes aren’t really surveys but they can be operated in a similar way. There would certainly be potential ethical issues if you wanted to use this approach for research.

The second approach to Web based surveys is to have a web page dedicated to the survey. Respondents have been asked to participate and sent a URL. Using dedicated web pages allows you to use a full range of question types, skip logic and repeating sections. You can also provide almost limitless amount of help and back ground on the questions. If the web pages are individually crafted, for a specific survey, then the technical requirements of this might approach might be excessive.

### Smartphone based

Again there are variations possible.

If you had a dedicated smartphone app. for a given survey, then the problems would be much the same as with the dedicated web page approach.

By using a generalised app with specific survey forms for each survey, the problem is reduced to creating the appropriate survey form. This is essentially what we will be doing using ODK.

You could use this approach and leave the respondent to complete the survey by themselves, but it means that you would have to provide instructions on how to have the app send the collected data returned to the correct cloud/server based repository for that survey.

One of the advantages of the dedicated web page approach was that we are able to provide limitless help and interpretations of the questions. For a smartphone app that is perhaps not quite so practical.

As an alternative, the smartphone app with a specific survey form can be administered by an in-person Interviewer. The Interviewer will also be able to direct the collected data to the appropriate repository. This approach does however negate some of the cost benefits of using an electronic survey approach in the first place.

# Sections 2

## Overview of ODK

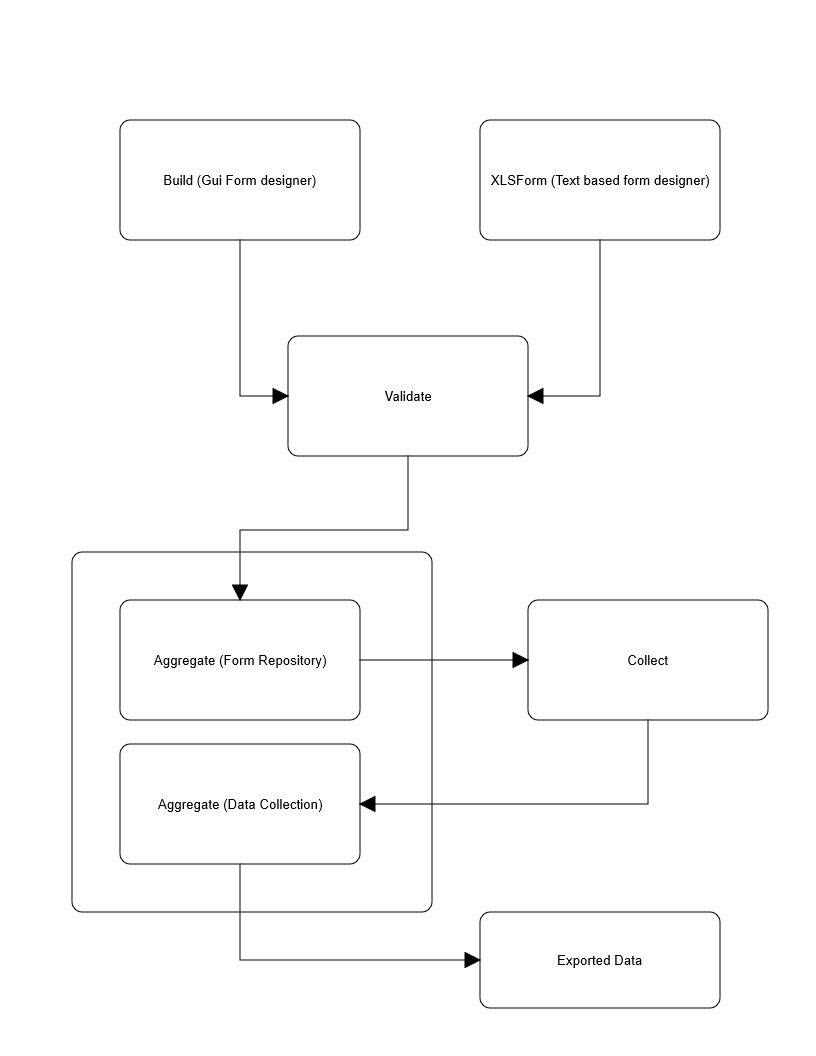
ODK (Open Data Kit) is a set of open source applications which work together to allow users to perform all of the necessary actions to create eSurvey forms, deploy them and collect data from completed forms.

ODK specifically targets the Android Smartphone as the method of completing and collecting data from forms. The forms themselves are based on the more generic XLSForm standard which in turn is based on the XForm standard supported by the [Javarosa Project](https://bitbucket.org/javarosa/javarosa/wiki/Home).

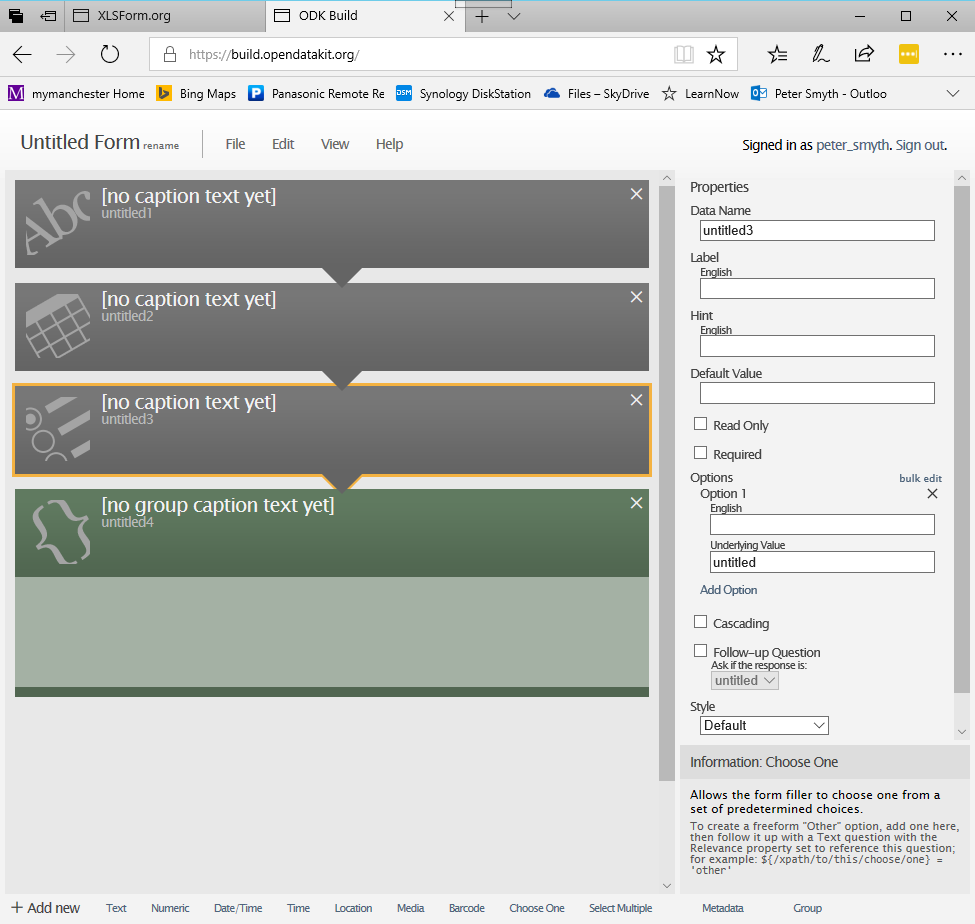
XLSForms are supported by several data collection platforms apart from ODK. Some are free to use, others are more commercially based by providing ‘add-on’ functionality. A partial list is provided in the [XLSForms](http://xlsform.org/) documentation.

## Components and general workflow

The diagram below shows the various ODK components and how they relate to one another



**Build:** This is a GUI (Graphical user Interface) which allows you to create simple forms. Although quite flexible it can be limiting for more complex forms. It is available as both an on-line (browser based) application or can be run as a standalone application on a PC



The screenshot above shows a form with four different elements in it. Elements are chosen from the list along the bottom of the screen. On the righthand side you can provide information related to the selected element. The on-line and off-line version look and work exactly the same.

**XLSForm:** is simply a translation program. It takes a specially constructed Excel spreadsheet and creates a file of XML (eXtensible Markup Language) which is used to represent your survey form.

Much of this workshop is about the creation of a suitable Excel spreadsheet for XLSForm to translate.

XLSForm can be downloaded and used off-line or ODK provides an on-line version available from their website. We will make use of this facility as it also provides us with the ability to see our forms rendered on a PC. Not everything is rendered, nor is it all rendered as it will appear on the smartphone but it is very useful to allow us to check the validity of the form and the logic flow as we develop it.

The link for the online version is [here](http://opendatakit.org/xiframe/) . We will go into more details of how XLSForm works in the next section of the workshop.

**Validate:** Both Build and XLSForm produce a file of XML as output. The validate program is designed to check whether or not the generated XML does represent a valid form.

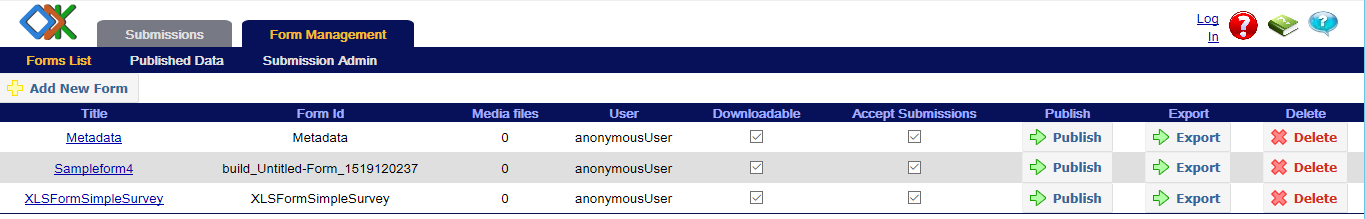
In this workshop we will be making use of the online version of XLSForm which will automatically run the produced XML through the validate program so we don’t have to. There is an offline version of validate as well for people using the offline version of XLSForm. Although the survey forms we create and download to the smartphone are XML files, we don’t need to interact with the XML directly.

**Aggregate:** The Aggregate program is managed in a web server. This could be your own private server, or it could be cloud based. There are many 3rd party organisations who host such servers and may allow you to use them for a fee. ODK itself has a freely available Web server from which you can download test forms to your smartphone, complete them and send the resulting data back to the server. It does not allow you to upload your own forms however.

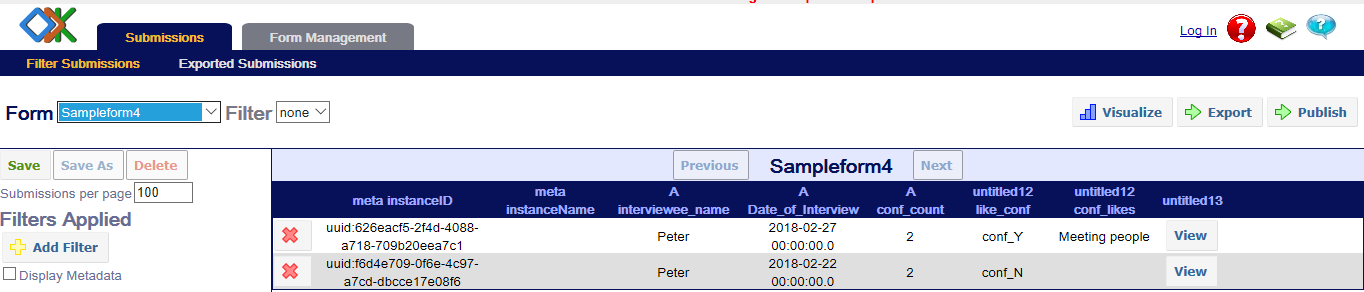
We don’t really need to use Aggregate for this workshop however a pre-built VM of Aggregate will be demonstrated for completeness.

There are really two parts of Aggregate. Aggregate is used as a repository for the eSurvey forms. They are stored in their native XML format. Aggregate is also used to store data collected from completed forms. This data can be downloaded from the server to a PC or laptop for further processing to take place. When you download the data, you will in general, download all of the data associated with a particular form. I.e. all of the data relating to your survey. It is possible to apply filters to the data before download should you only want part of it.

You can download the data in either csv or json format. Which one you choose will depend on the complexity of the data structures (which in turn will depend on the complexity of your eSurvey form). We will look at how downloaded data can be further processed in the last part of this workshop.

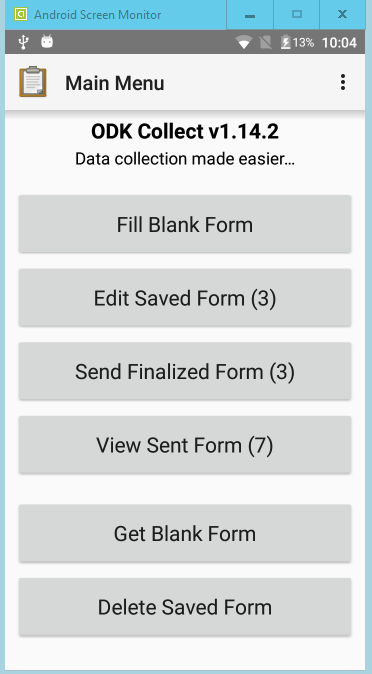


A list of Forms in ODK Aggregate

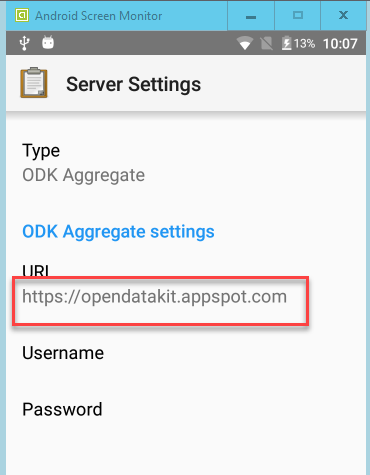


Uploaded data from a form in ODK Aggregate

**Collect:** Collect is the Android application used to download the form template from an Aggregate server, render the XML file into a proper form, allow the user to complete the form and finally transfer the data contained with the complete form back to the Aggregate server where it is stored with all of the other completed form data from the same form.



When collect is first installed it ‘points’ to the ODK default server. You can use this to download test forms and try it out. Typically, you will change the server URL to point to your own Aggregate server.



### Things we are not going to cover

For simplicity in this workshop we are taking a rather non-existent approach to data security.

In reality, for real research surveys, data that you collect may be of a sensitive nature. Interviewees might reasonably expect their data to be kept secure at all times.

Forms can be created that are encrypted. That is, the data collected when a form is completed will be encrypted. It is important to note that data associated with a form on the Android device which has not been marked as completed will not be encrypted.

The Aggregate server can be setup so that it uses the https rather than the http protocol. This will ensure that data transferred from the Android device to the Aggregate server will be encrypted in transit

On the Aggregate server, the data will be unencrypted, however the Aggregate server can (and normally would be) set up to make use of ordinary user access controls. I.e. access to the forms and data associated with a given project can be restricted by user account.

**Danger after download!**

Once the data has been downloaded, it is the user’s responsibility to keep it secure.

# Section 3

## Using XLSForm

In this session we are going to create an Excel spreadsheet which can be converted by XLSForm into an XML file which we will then load into Aggregate.

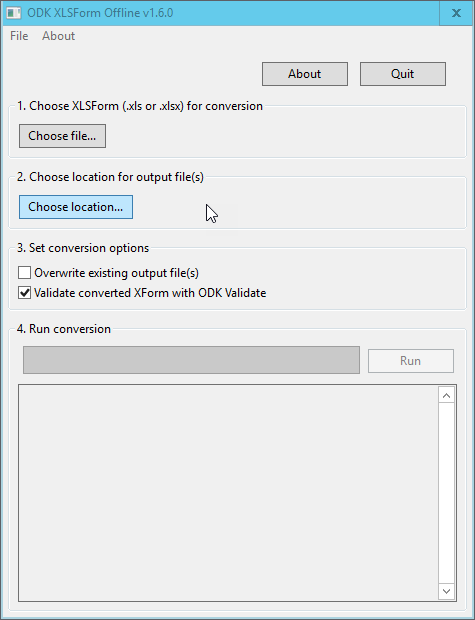
Before we start constructing the spreadsheet we will have a quick look at the different ways of how we can use XLSForm.

XLSForm can be downloaded as a standalone program. The XLSForm offline program can be downloaded from this github [link](https://github.com/opendatakit/xlsform-offline/releases). You download a .zip file, the unzipped file has the XLSForm executable and a short readme file. The executable file includes the version number in the name so it might vary depending on when you download it.

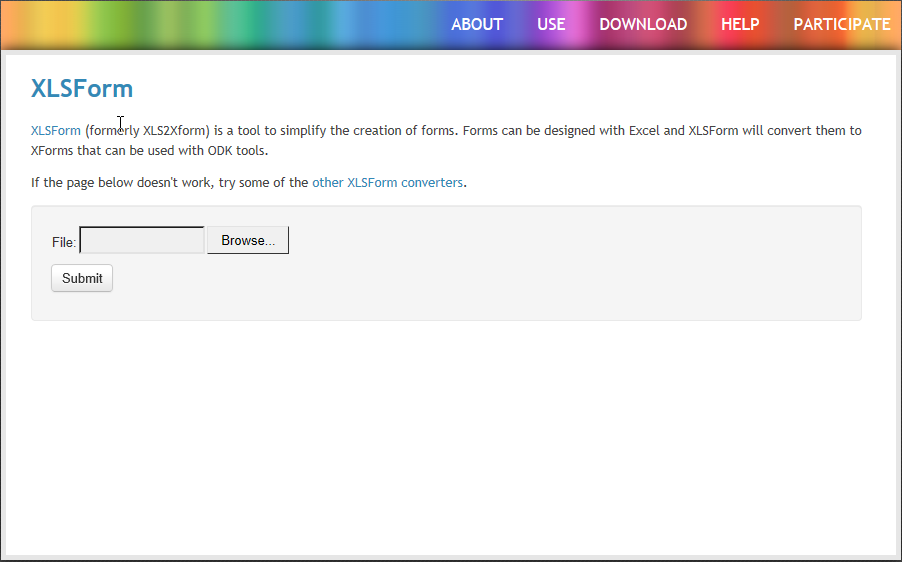
The Readme file explains possible problems you might encounter. One key point however is that the Validate functionality will only work if you have Java installed on your PC. This is in fact similar to the standalone Validate program which also requires Java.

To use

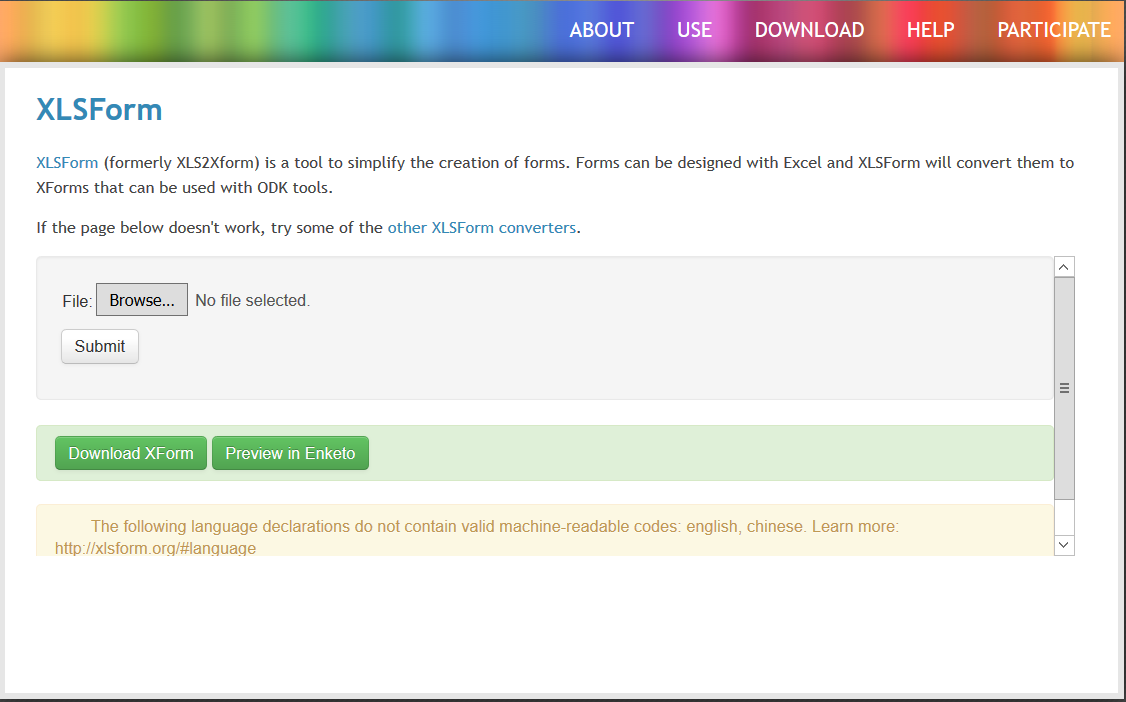
When you open the XLSForm offline program, you are presented with a single Window which allows you to choose the name and location of the XLS or XLSX file you wish to convert into an XLSForm (i.e. to an XML format file) and the location of where you want the output file to be saved. Optionally you can request that your form be Validated. If there are any warning or error messages produced from the Validate process, then they displayed in the bottom half of the Window.



As an alternative to this ‘offline’ version of XLSForm there is also an ‘online’ version which can be accessed from this [link](http://opendatakit.org/xiframe/).



Here you only have to provide the location of the XLS or XLSX file you want to convert. I t will automatically be converted and validated. You are then given the options of downloading the resulting XML file and the option to display the form online using the EnKeto system (Enketo is a 3rd party organisation which provides ODK form and data storage and processing)



When you are developing a form being able to see the results in this way can be very useful as you don’t have to the form into the Android phone using Collect. There are some restrictions on what the Enketo system will display from you form. For example it will not show audio or video and it cannot show data items which are relevant only to a real phone, like the phone number.

**Exercise**

Using the ODK provided xlsform-sample.xls file, upload it to XLSForm online and view the results in Enketo.

## Creating forms with XLSForm

The title is slightly mis-leading. We are actually going to create an Excel spreadsheet which the XLSForm program will convert into an XLSForm.

Our Scenario – Cinema Habits

We are going to create a survey form which will ask people about their Cinema going habits.

We will construct the survey in two parts; We will start off by asking for simple information about the respondent and their cinema likes and habits. This will create a simple linear form but will allow us to look at several of the possible question types available.

We will then modify the form to request additional details about a number of films which have been seen, or would like to see etc. This will allow us to see how repeating groups work. We will also introduce questions which are only asked depending on a previous response.

**Exercise**:

Consider how question flows like repeating questions and conditional questions handled in a paper survey?

Starting our Spreadsheet

In order for the XMLSForm program to understand our spreadsheet, we must lay it out in a well-defined manner. It requires specifically named tabs. The tabs need to have specific column names, some are mandatory, some are optional.

Because the layout is so well defined we can start our own spreadsheet based on a template, which has many of the column headings and tabs already included.

[screen shot of the template]

Our template has two tabs; ‘survey’ and ‘choices’. We use the ‘survey’ tab to record the details of the form and the ‘choices’ tab to provide lists of choices for any question that needs them.

**The ‘choices’ tab**

This tab has three columns.

‘list\_name’ : This the name that you give to a list, you will refer to the list by name when you need to reference it in a question.

You can have as many entries in a list as you want. The list is defined by all of the entries sharing the same value in the ‘list\_name’ column.

‘name’ : This is the values stored when this particular list item is selected in the form. The end user (respondent) will not see this value.

‘label’ : This is the text which actually appears in the form. You use this to allow the respondent to make a choice.

**The ‘survey’ tab**

The ‘survey’ tab contains three required columns and several optional columns

‘type’ : This refers to the type of question being asked. It is also used to denote the start of groups or repeating groups.

‘name’ : This is an internal variable name used to store the value provided by the respondent. The name doesn’t appear on the form itself, but the value it stores can be used internally by the form, for example in deciding whether or not an optional question should be asked

‘label’ : This is the text which actually appears in the form. You use this to help the respondent to understand the question.

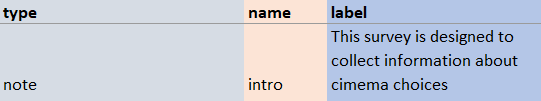
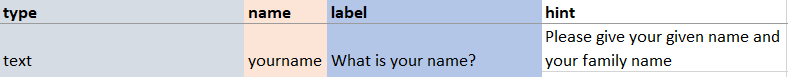
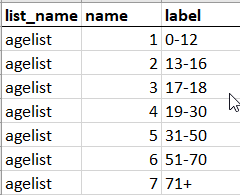
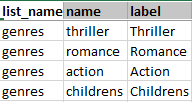
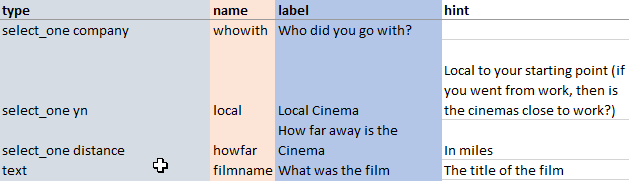
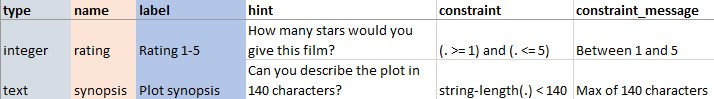
Several other column names are also possible which we will describe as we need to use them.

In general XLSForm expects the column names and tab names to be lower case. Apart from that there is no real restrictions on the formatting of the spreadsheet. In the template the columns which are required have been coloured and the column heading have been made bold and a bit of formatting of the cells has been added. This is all entirely optional and would normally be used simply to aid (human) readability.

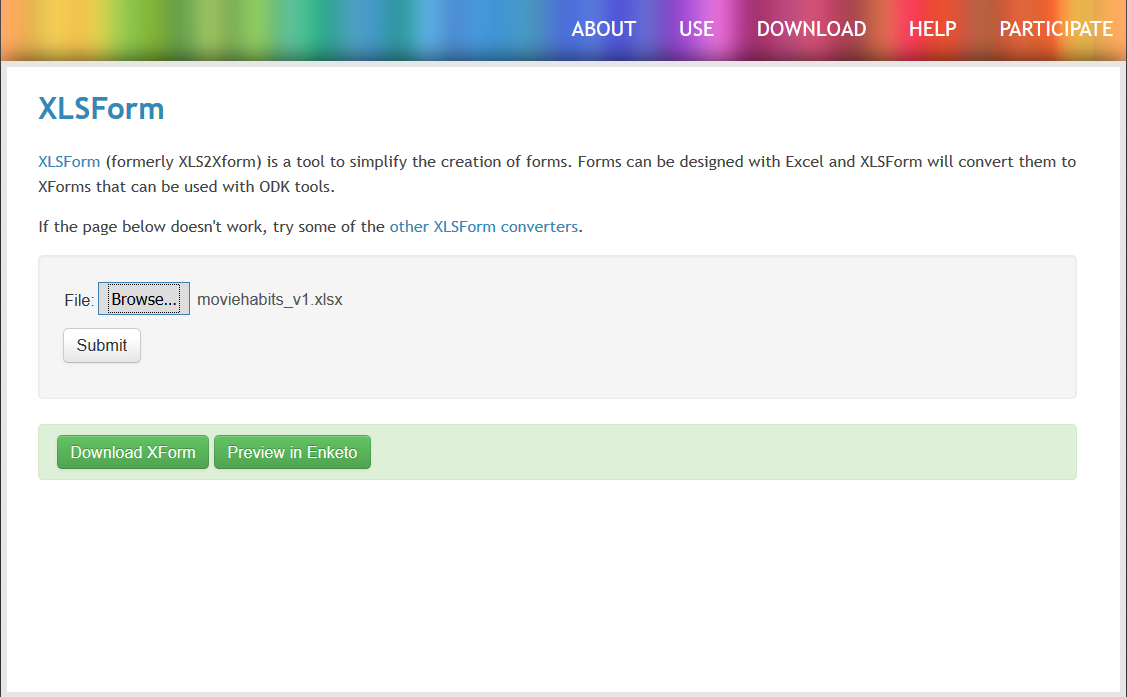
## A simple form

Step 1: Create a new spreadsheet called ‘moviehabits\_v1’ by renaming a copy of the template spreadsheet; XLSForm Template.xlsx

Step 2: Add some simple questions to our form.

1. The first thing that we are going to add, isn’t a question at all. It is a comment. This will be displayed to the user so that you can provide explanations of the survey or of particular questions or groups of questions. You can add a note any where you like but at the beginning to explain the overall survey and at the beginning of each group of questions is most usual.  
     
   To add a note;   
     
     
     
   The type is ‘note’ and the label is the text of the note. In this example I have specified a name of ’intro’. This is essential the name of an internal variable which we will never reference. You can call it whatever you like but bare in mind that variable names have to be unique, so if you have other notes, you will have to create new names for them.
2. We now start with the questions proper. We will start by collecting some information about the respondent. Their names, sex and the age group to which they belong.  
     
   To ask the persons name;  
     
     
   ‘type’ refers to the type of response we are expecting. In this case we are expecting text ans so on the form the respondent will be presented with a text box into which they can write their answer. The value of the response will be stored in the ‘yourname’ variable. The hint acts a bit like a note and appears immediately below the question.  
     
   To ask for an age group;  
     
     
   Here the type of question is ‘select one’. In order to select one, you must be given a choice to select from. The choices are defined in ‘agelist’ list in the choices tab.  
     
     
     
   The ‘agelist’ list has seven entries. The user will see the label values and be able to select only one. The value given in the name column is the value that will be placed in the ‘agegroup’ variable.  
     
      
   To ask for the sex of the respondent;  
     
     
   This works in much the same way as the age group. We are again using the ‘select’ one’ type but this time with a different list ‘sex’. This list is defined in the choices tab  
     
     
     
   The variable name ‘sex’ will be given the value of either 1 or 2 depending on the choice made. We could just as easily have assigned the values of ‘M’ and ‘F’ by specifying these in the name column of the sex list.
3. Next we want to ask what film genres the respondent likes.  
     
   To ask for the film genres liked;  
     
     
   Unlike the previous two questions, here the respondent can select more than one option. So the question type changes from ‘select\_one’ to ‘select\_multiple’. We still use a list of the choices in this case ‘genres’.  
     
     
   It is difficult to make a list like this to be inclusive, i.e. include all possible genres, so we include in the question type the ‘or\_other’ clause. The effect of doing this is that when the ‘genres’ list is displayed to the respondent on the device, an additional choice of ‘other’ is added. If this is selected, then a textbox is presented for the respondent to record what there ‘other’ genre is.  
     
   For the genres question the label and hint work the same way as for the other question types. The name ‘genres’ is still a variable name, but in this case it needs to store not a single value, but the values associated with all of the choices the user selected. It does this by storing the choices as a comma separated string. If the ‘other’ option is taken, then the value that the respondent gave will be stored in a variable called ‘genres\_other’. We will look at these types of variables in more detail in the last section of the workshop when we look at processing the collected data.
4. Next we want to ask the date of their last cinema visit;  
     
   To ask for a date;  
     
     
   The question type is simple ‘date’. Name, label and hint all work as for the other question types. On the device a ‘select date’ button is presented, when clicked a standard (for the device) date picker. The selected date is stored in the ‘lastvisit’ variable.
5. More questions about the cinema visit.  
     
   These questions are of types we have seen before;  
     
     
   Providing a hint is always optional. You need to decide if the information in the label is sufficient or whether you need to add clarification. For the ‘How far away is the Cinema’ question, the hint ‘in miles’ clearly adds clarification.
6. Getting more information about the film.  
     
   Getting a star rating and a plot synopsis;  
     
     
   For the rating we have a new question type of ‘integer’. As the name suggests we are expecting an integer value to be provided.  
     
   Of more interest is not the question types but the fact that we are specifying constraints on the values of the responses.  
     
   The constraint condition is given in the constraint column. It is an expression which when evaluated yields either True, in which case everything is OK and we can move on to the next question, or False, in which case the ‘constraint\_message’ appears as a pop-up and the respondent has to provide a new response that does satisfy the constraint. You can not move on to the next question until you provide a valid response. In theory the ‘constraint\_message’, like the ‘hint’ is optional. However, it is highly unlikely that you would not provide a constraint message.  
     
   We need to consider the format of the constraints.   
     
   1. The ‘.’ In an expression represents the current variable. So, in these examples it will be either the ‘rating’ variable or the ‘synopsis’ variable depending on which of the constraints you are looking at.   
     
   2. Much of the constraint expression is constructed in a similar way to an Excel expression would be. Importantly however you do not start with the ‘=’ sign, they are just simple text.  
     
   3. You can use brackets to enforce precedence, you can use ‘and’ and ‘or’ to build more complex expressions, you can use the usual range of arithmetic operators.  
     
   4. In the plot synopsis question, we are making use of an XLSForm built-in function called ‘srtring-length’ which returns the number of characters in a string. In this case the current variable ‘synopsis’. A list of the available built-in functions can be found at this [link](https://opendatakit.org/help/form-design/binding/).

This completes our initial form – it is time to try it out. To do this we will use the online version of XLSForm at this [link](http://opendatakit.org/xiframe/).



Use the browse button to navigate to where you have stored your form and then click the submit button. If there are errors in the spreadsheet they will be listed in red on the bottom part of the window. If all is well, the screen will look the same as above but instead of the file name it will say ‘no file selected’. You can then click the Preview in EnKeto button and your form should load in a new tab of your browser.

In the browser the form looks like a continuous web page. On a smartphone there would be only one question on a screen and you would swipe left to get the next question.

Exercise

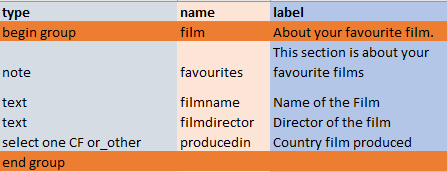
On the web page; check the ‘other’ option in the Genres question and see what happens. Check that the constraints on the Ratings and Synopsis questions are working as expected.

## A more complex form

We will now continue the development of our form by adding a few more questions, including repeating groups of questions, questions which are only asked depending on a previous response and grouping questions together. We will also add some further refinements on how the survey is displayed on the device and gather some smartphone specific information.

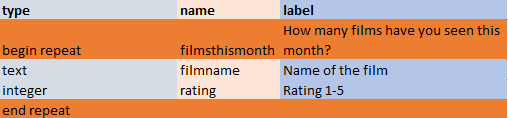
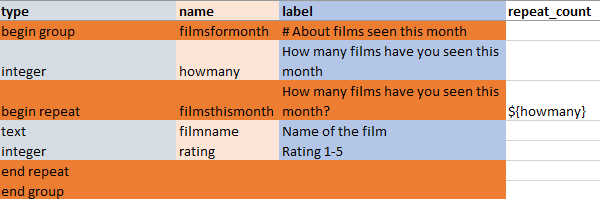
Step 1: Creating groups of questions.

Groups are logical divisions of questions in the survey used by the survey designer to keep questions on a similar topic together. They provide the opportunity to give additional information to the respondent, but the questions are displayed in much the same way as before.

1. In the spreadsheet, purely for our convenience, we will colour code the start and end of the group.   
     
     
   The label associated with the ‘begin group’ type is displayed above each question when it is presented on the phone. In the ‘select one’ question we have included the ‘or\_other’ option. This works in the same way as it does for the ‘select\_multiple’ type, if it is selected, a text box appears to allow the respondent to enter a value. In our example, the value will be stored in a variable called ‘producedin\_other’. You don’t need a ‘name’ or ‘label’ for the ‘end group’.

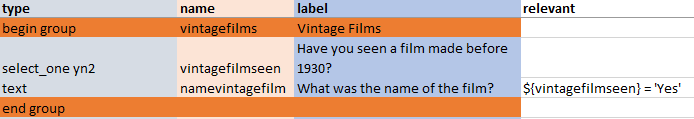
Step 2: Adding repeating groups

There are two main ways of adding repeating groups that we will look at.

1. Adding the groups as needed.  
     
   We use the ‘begin repeat’ and ‘end repeat’ to delimit the set of questions which are to be repeated. We do not indicate how often the they should be repeated. On the phone, we will be asked if we wish to create a group for these questions and every time we complete the group of questions we will be asked again until we say no. Like the ‘end group’ the ‘end repeat’ does not require a ‘name’ or a ‘label’.
2. We can however anticipate how many times the repeat group is shown by either explicitly providing an integer value, or more commonly using the given response from an ‘integer’ type question.  
     
     
   We really need the ‘begin group’ and ‘end group’ but it is convenient to use them to show that the ‘howmany’ variable is to be used within the ‘repeat’ group. When the ‘howmany’ question is answered, the ‘howmany’ variable will contain an integer value. The variable ‘howmany’ is specified in the repeat\_count column. We have previously referred to a the current variable, i.e. the variable associated with the current question as ‘.’. Now , however we need to refer to the variable associated with another question. To do that we use the variable name in curly braces and prefixed with ‘$’. The repeat\_count value is used to decide how often the repeat group should be presented. If the respondent had entered 0 for the ‘howmany’ question then the repeat group would not be presented at all.

Step 3: Conditional questions

It is quite common to have questions which only become relevant depending on answers to previous questions. On paper surveys all of the questions need to be included in case they are needed and you end up with lots of ‘go to’ or ‘skip’ type messages. In an eSurvey, this can be done a lot more cleanly. Questions are only presented if they are relevant.

1. ‘Skip’ questions  
     
     
   The ‘begin group’ and ‘end group’ types are optional. The ‘select\_one’ question has a list ‘yn2’ associated with it. The value selected is stored in the ‘vintagefilmseen’ variable.  
     
     
   In the ‘relevant’ column of the ‘namevintagefilm’ question the value of the vintagefilmseen’ variable is compared to the value ‘Yes’. Only if they match, i.e. the expression ${vintagefilmseen} = 'Yes' evaluates to True will the question be presented to the respondent.

Exercises

1. Save your spreadsheet and then submit it to the online XLSForm utility. Have a look at the Enketo form produced. Notice how it deals with the Groups and repeating groups.
2. Change the Vintage films groups to use a repeating group so that you can record all of the pre-1930 films that the respondent can remember. What kind of repeating group do you think it best to use here?

Step 4 : Adding some extras

1. Currently all of the questions in our questionnaire are optional, they can be skipped by swiping left. In Enketo you can click the validate button at the end of the form and it will say that the form is valid. We can make selected fields ‘required’  
     
     
   By putting ‘yes’ in the ‘required’ column we are making a response to the ‘agegroup’ and ‘sex’ questions mandatory. On the smartphone and Enketo they are marked with an ‘\*’. On the smartphone you will not be able to move on without providing an answer and in Enketo the form will not validate unless they are completed.
2. Having to answer a question and then swipe left to get the next question is a bit tedious. It would be nicer if the form automatically moved on after you provided an answer.  
     
     
   It is not always possible to tell when an answer is complete, however for the ‘select one’ questions it is. By adding ‘quick’ in the appearance column we can make the form move on to the next question as soon as a choice has been made.

This is the end of our form development. Save the spreadsheet and submit it to the online version of XLSForm one last time to see the effects of the extras. If you view the form using Enketo you will see the ‘\*’ for the manadatory fields, but it cannot show the automatic question movement.

If you have an Android smartphone with ODK Collect installed connected to your laptop you can download the XML version of your form from XLSForm online by right mouse clicking the ‘download XForm’ button and saving the file.

You can then copy this downloaded XML file onto your phone by copying it to the ‘This PC\<your-phone-name>\Internal storage\odk\forms’ folder. We will look at this in the next section.

# Section 4

## Demonstration of Collect

Demonstration of using collect

* download form, complete, send data back

## Demonstration of Aggregate

Show how the data is accessible on Aggregate and how the different components are represented

Specifically :

* ‘other’ options
* Groups
* Repeating groups
* Skip questions

Downloading completed forms from Aggregate in:

* csv format
* JSON format

# Section 5

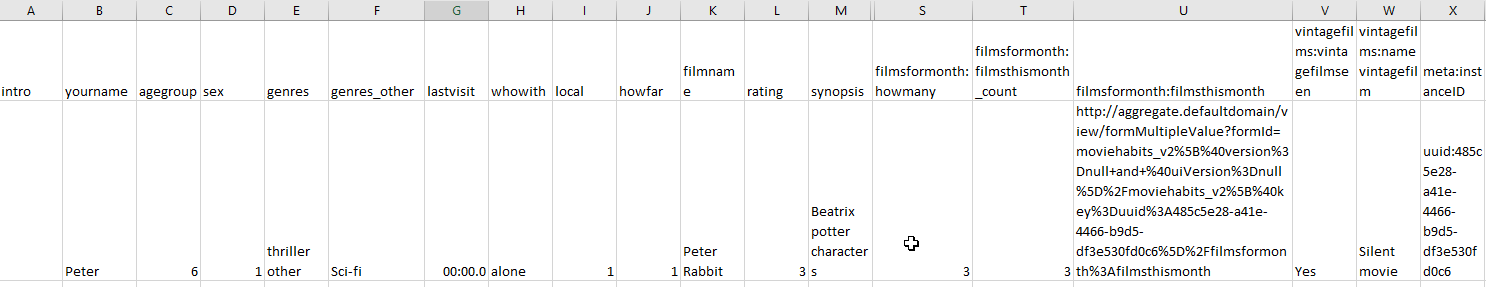
## Comparison between CSV and JSON data formats

Both CSV and JSON are data storage formats. In both cases the data is stored in a human-readable format in a text file.

### CSV

CSV – comma separated values is a format used to store tabular data. A table has rows and columns. Quite often the first row is a set of column headings so that you know what meaning the data in subsequent rows should be given. The column values are separated in each case by a comma, hence the name CSV.

You don’t have to use a comma, any single character will do. Often the non-printable ‘tab’ character is used, such files are often referred to as TSV files. In practice as long as you know what the separator character is, you can process the data in the file in a similar way and the generic name of CSV is used.



[Example of CSV file – taken from aggregate]

The screenshot above is of a csv file downloaded from Aggregate. It is data which has been collected using the movie\_habits form. It has a first row of column headers matching our variable names. If the variable occurs in a group, then it is prefixed with the group name. Subsequent rows represent the data for each completed survey form.

This works fine until we get to the section on ‘How many films seen this month’. The number is recorded, but for the details you are only given a rather obscure link. This is not much use as it stands. We have seen that in Aggregate we get a ‘View’ button which opens a new window with the details of the repeating group in it.

### JSON

JSON – Java Script Object Notation was in fact designed as a way of transferring data between two computer programs using APIs (Application Programming Interface). Because the format is human-readable, it is also very widely used by APIs returning information to end users (people like you).

If data can be represented in a CSV format, it will always be possible to represent the same data in JSON format. However data which can be represented in JSON format cannot always be represented in CSV format so easily as we have just seen with our movie\_habits data.



You can see that in JSON, every ‘record’ includes the equivalent of the column names and for a particular ‘column’ the value it takes in the particular record. In JSON these are referred to as Key Value pairs and the ‘records’ are known as documents. Each key in a document has to be unique. Each document is enclosed in ‘{}’ and the entire set of documents is enclosed in ‘[]’



If we look at the JSON file produced when we downloaded out movie\_habits data we can see this format. However some of the ‘values’ are not simple types (numbers, strings etc).

In the ‘genres’ question the respondent could make multiple selections. These selections are represented by a list, something in square brackets. Because each item in the list is itself a simple type, these can still relatively easily be represented in the csv version of the data.

In the repeating group ‘filmsthismonth’ the responses are still represented by a list, but here, as each iteration of the group involved answering two questions, these answers are themselves represented as a separate document in ‘{}’ brackets with key, value pairs for the questions. Notice that each of these documents is allowed to have its own set of keys.

Because of this nested structure, documents within documents, it is possible to represent everything in the form response as a single JSON document. This is why for more complex documents the JSON format can be preferable, although as we will see, after it has been downloaded, it need more careful processing.

We will look at processing this JSON file in the next section.

# Section 6

## Processing the JSON output

The code and notes for this section is contained in the Jupyter notebook:

**movie\_habits.ipynb**