Data collection in on-farm trials

# Some general principles

* Standardization – everyone collects the data in the same way following the same protocols; good design of the ODK form is important to achieve that.
* Consistency – design your form with a consistent method for labelling fields, trials, plots and treatments. Use barcodes if possible, or a consistent naming.
* Prompt submission – ensure data is submitted on the day that data is recorded. Avoid forms where data needs to be recorded on different days and is stored on the tablet or phone in draft. Mobile devices can get lost or break down, and then data stored in draft forms is lost. When data is submitted on the day it is recorded, analysts can evaluate the data and indicate possible issues which can be corrected at a subsequent data collection event.
* Traceability – always know who collected data when and where.
* Flexibility – allow your field teams to work efficiently, independently and simultaneously. Avoid repeat long repeat loops that require one person to record all data. Design your ODK tools so that field teams can distribute different field measurements, or plots.
* Generalisation – design your forms so that these are generally applicable across trial types, regions/countries. Avoid needing to design multiple versions and minimize maintenance and update requirements.

We will provide examples of a toolset to record data for two types of on-farm experiments: (1) a multi-locational validation exercise with a simple one farmer = one replicate design, and (2) an on-farm RCBD researcher-managed experiment.

# On-farm validation exercise

## Before you start developing your ODK form…

* Start from a comprehensive experimental protocol, which needs to provide all the necessary details on (1) the experimental design and treatment structure, and (2) what data needs to be collected when and how.

Let’s start with the experimental design and treatment structure. In our example, the design is a multi-locational on-farm validation exercise. A total of 200 farmers will evaluate a new fertilizer recommendation for potato, against the current standard fertilizer recommendation and a control without fertilizer application. Each farmer will lay out 3 plots; treatments are identified by a colour for easy reference.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Treatment nr** | **Description** | **Colour** | **NPK 17:17:17 rate [kg/ha]** | **Urea rate [kg/ha]** | **Application regime** |
| 1 | Control | Black | 0 | 0 | N/A |
| 2 | Current recommendation | Blue | 300 | 0 | Two equal splits at 0 and 6 WAP |
| 3 | New recommendation | Red | 400 | 100 | 300 kg/ha NPK at 0 WAP; 100 kg/ha NPK + 100 kg urea at 6 WAP |

* Next, extract a data collection schedule from the protocol:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **When?** | **Data** | **Variable descriptions** | **Variable name** | **Variable type** |
| 0 WAP (at planting) | Trial location | GPS location | geoloc | geopoint (vector of latitude, longitude, altitude and accuracy) |
|  | Field history | Nr of years since last fallow | yearsFallow | integer |
|  |  | Crop grown in preceding season | lastCrop | categorical |
|  |  | Nr of years since last cultivated with potato | yearsPotato | integer |
|  | Land preparation | Land clearing method | clearing | categorical |
|  |  | Management of cleared vegetation | vegetation | categorical |
|  |  | Number of tillage operations | nrTillage | integer |
|  |  | Ridging operation | Ridging | categorical |
|  | Planting date | Date of planting | plantingDate | date |
|  | Variety grown | Name of the variety grown | variety | categorical |
|  |  | Source of the planting material | varietySource | categorical |
|  | Planting density | Between row distance (m) | betweenRow | decimal |
|  |  | Within row distance (m) | withinRow | decimal |
|  | Organic inputs applied | Type of organic inputs applied (if any) | organicInputs | categorical |
|  | Fertilizer applied | Confirmation that the first split was applied as per the protocol for each plot | fertilizer1 | categorical |
| 4 WAP | Plant stand | Plot width (m) for each plot | plotW | decimal |
|  |  | Plot length (m) for each plot | plotL | decimal |
|  |  | Nr of plants germinated (for each plot) | nrPlants | integer |
|  | Weeding | Weeding dates during 0-4 WAP | weedingDate | date |
|  | Spraying | Spraying dates during 0-4 WAP | sprayingDate | date |
|  | Trial rating | Scores for possible issues in the trial: drought, waterlogging, erosion and diseases (esp. potato late blight (PLB) and potato bacterial wilt (PBW)) | rateDrought, rateWaterLogging, rateErosion, ratePLB,  ratePBW | categorical |
| 6 WAP | Fertilizer applied | Confirmation that the second split was applied as per the protocol for each plot | fertilizer2 | categorical |
|  | Vigour scoring | Scoring of plant vigour for each plot | rateVigour | categorical |
| 8 WAP | Weeding | Weeding dates during 5-8 WAP | weedingDate | date |
|  | Spraying | Spraying dates during 5-8 WAP | sprayingDate | date |
|  | Trial rating | Scores for possible issues in the trial: drought, waterlogging, erosion and diseases (esp. potato late blight (PLB) and potato bacterial wilt (PBW)) | rateDrought, rateWaterLogging, rateErosion, ratePLB,  ratePBW | categorical |
|  | Vigour scoring | Scoring of plant vigour for each plot | rateVigour | categorical |
| 10 WAP | Vigour scoring | Scoring of plant vigour for each plot | rateVigour | categorical |
| 12 WAP | Weeding | Weeding dates during 9-12 WAP | weedingDate | date |
|  | Spraying | Spraying dates during 9-12 WAP | sprayingDate | date |
|  | Trial rating | Scores for possible issues in the trial: drought, waterlogging, erosion and diseases (esp. potato late blight (PLB) and potato bacterial wilt (PBW)) | rateDrought, rateWaterLogging, rateErosion, ratePLB,  ratePBW | categorical |
|  | Vigour scoring | Scoring of plant vigour for each plot | rateVigour | categorical |
| ~16 WAP (harvest) | Weeding | Weeding dates during 9-12 WAP | weedingDate | date |
|  | Spraying | Spraying dates during 9-12 WAP | sprayingDate | date |
|  | Trial rating | Scores for possible issues in the trial: drought, waterlogging, erosion and diseases (esp. potato late blight (PLB) and potato bacterial wilt (PBW)) | rateDrought, rateWaterLogging, rateErosion, ratePLB,  ratePBW | categorical |
|  | Tuber yield | Plot width (m) for each plot | plotW | decimal |
|  |  | Plot length (m) for each plot | plotL | decimal |
|  |  | Nr of marketable tubers in each plot | nrTubers | integer |
|  |  | Total weight (kg) of marketable tubers in each plot | tuberFW | decimal |
|  | Tuber price | Farmer’s current expected sales price (local currency/kg) at his/her outlet market | priceTubers | decimal |
|  | Farmer evaluation | Farmer’s rating for performance of new versus current recommendation in terms of plant vigour, yield and tuber quality | farmerScoreVigour, farmerScoreYield, farmerScoreQuality | categorical |

## Principles of designing the xlsform

* Uploading and testing: As a rule, always upload and test yourXLSform as many times as you can in between building. Do not wait until it is complete.This will minimise repetition of errors in different groups/sections of the form.
* Naming: *Worksheet* names should be named appropriately and in lower case: i.e “survey”, “choices” & “settings”. *Column headers* should be in lowercase and the first 3 are mandatory in the “survey” sheet i.e. “type”, “name”, “label”. “Choices” sheet columns as well: i.e “name”, “list name” & “label”. *Field names* should be unique and should not contain any spaces or special characters (only ‘-’ and ‘\_’ are allowed.
* Legibility of the XLSForm: On the layout of the questionnaire, it is good practice to: Separate different sections/groups of the questionnaire using *different cell colours,* *freeze the top row cell* and add *automatic filters* to quickly navigate through your questions (survey) and answers (choices).

Please refer to these online references for further information: ([xlsform.org](file:///C:\Users\User\Documents\ACAI\ODK%20training\xlsform.org), <https://docs.getodk.org/xlsform/>, <https://help.ona.io/article-categories/form-authoring/>). You can also watch an XLSForm authoring video [here](https://www.youtube.com/watch?v=-0JCAnUQr9E).

1. Setting up the collect app and server

* Two applications needed: an application on the mobile device to collect the data, and a server application to aggregate the submissions. See: <https://docs.getodk.org/getting-started/>
* We will show examples using odk collect and [ona](https://company.ona.io/). There are quite a number of ODK compatible servers e.g. [ODK Central](https://docs.getodk.org/central-intro/) , [Kobo](https://www.kobotoolbox.org/) , [Commcare](https://www.dimagi.com/commcare/), [SurveyCTO](https://www.surveycto.com/), [DataWinners](https://www.datawinners.com/), [Secure Data Kit (SDK)](http://www.securedatakit.com/), [Tattara](http://tattara.com/), [Survey123 for ArcGIS](https://survey123.arcgis.com/) and [Community Health Toolkit](https://communityhealthtoolkit.org/) and alternatives to ODK Collect e.g. SurveyCTO Collect and KoBoCollect.
* For purposes of this guide and examples, you can create an ona account [here](https://ona.io/join).

**Now with this information, we can start building an ODK form.**

## First Exercise: Building our xlsform

To create an xlsform, open a new Excel spreadsheet and name the first 3 tabs as survey, choices and settings.

* Let’s start with the last sheet (settings) – The settings sheet and all of its items are optional, but can provide valuable information such as a title to your form as well as a unique ID for future versions identification. If no ID string is provided, the form title will be used, and if no form title is provided, then the XLSForm file name you give will be used instead.

For a start, add three columns: “form\_title” and “form\_id” and “version” and leave these blank for now.

* Now let’s go to the first sheet (survey) – assign names to the 3 columns (in order):

**type** which specifies the question type, **name** specifies the unique variable name & **label** whichspecifies how the questions or instructions appear to the user.

More columns can be added for enhanced capabilities but these first three are mandatory. For any ODK updates on new columns that you can add to your survey sheet, see [here](https://xlsform.org/en/ref-table/).

* The second sheet is **choices** – here is where we will list the multiple choice options for the categorical variables.

Assign 3 mandatory columns: **list name** is where the answer choices are listed, **name specifies the unique variable name for the corresponding answer choice**, and **label specifies how the answer choice will be presented to the user on the mobile device.**

* Save your now completed blank XLSForm template.

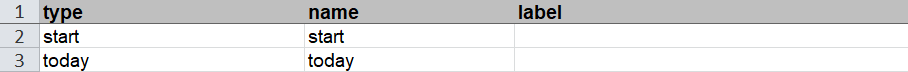
**We are now ready to start!**

## Let’s make sure all data is traceable and consistent

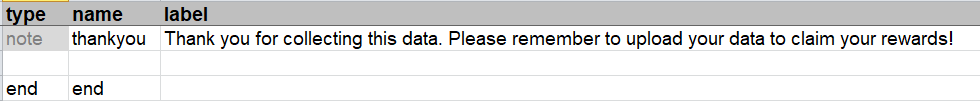
This means for all data, we need to know who collected the data when and where.

Let’s start with the when

* **Metadata** question types are captured at the back end and can be seen only when viewing collected data. They therefore do not require a label. In the survey sheet, use **start** and **today** as shown**.**



* Remember to use **end** at the end of your survey worksheet.



This allows you to automatically record on which date the data is collected, at what data entry started, and when the form was completed.

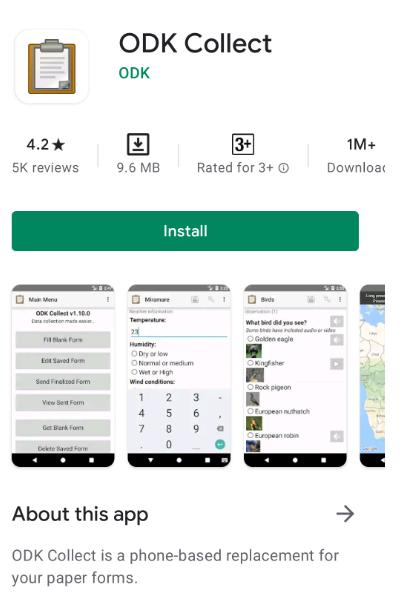
* Identification: you can use username as a metadata variable.

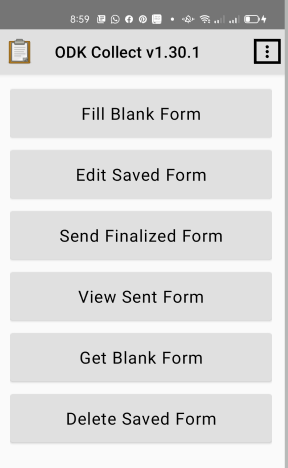
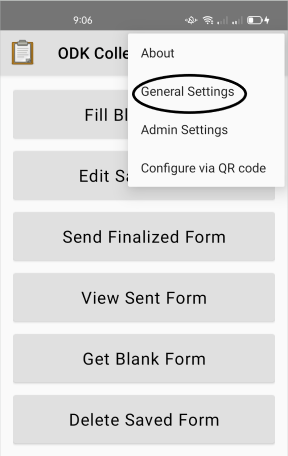


But better is to include a step to identify the enumerator. You can do this by requiring user to specify their name. But better is to use a barcode identifier. So you can provide your enumerators with ID cards that have a unique standardized barcode. You can design an ODK form to identify enumerators (and farmers) – we will come back to that later.

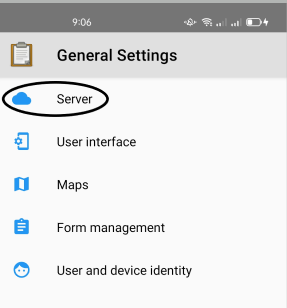
**Let’s upload this now and test…**

## Validate & Deploy the XLSForm

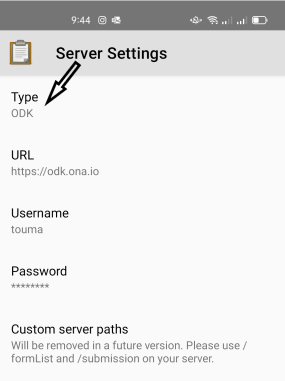
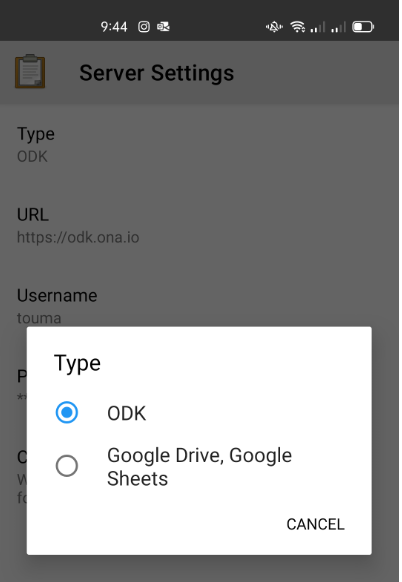
* You can upload your form to ona using the steps documented [here](https://help.ona.io/knowledge-base/how-do-i-upload-xlsforms-to-my-project/). Ona has a built-in validation & debugging check. You can find [common error messages](https://help.ona.io/knowledge-base/guide-troubleshooting-common-error-messages/) here that you may incur when deploying your XLSForm.
* To upload to ODK Central, use these [steps](https://docs.getodk.org/central-forms/).
* You can also use alternative online validation / debugging sources such as <https://opendatakit.org/xlsform/>
  1. How to install & configure ODK Collect
* On the Android device, go to the [*Google Play Store*](https://play.google.com/store?hl=en), search for ODK Collect and Install. 
* If you already have ODK Collect installed, update it so you have the newest version.
* Open the ODK Collect app
  + 1. Configuring ODK Collect with your Aggregate account
* Click on the Menu (three dots in the upper right corner)
* Click on General Settings

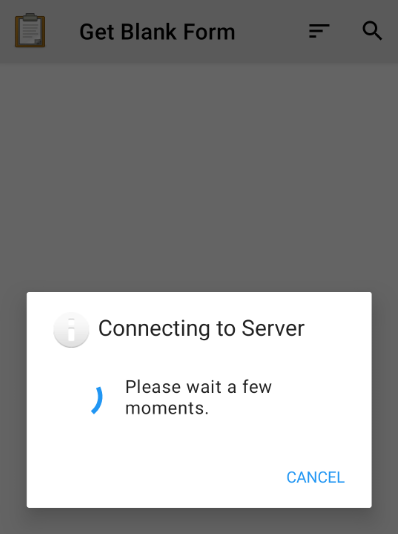
* Click on Server



* Under Server Settings, click on “Type” so you can change the destination path for where your ODK Collect data will be sent. Here you will find two options: “ODK” and “Google Drive, Google Sheets”. Select "ODK"

* Next, click on URL and type in your server URL. For example if using Ona, your URL will look like this: [**https://odk.ona.io**](https://odk.ona.io)
* Provide your username and password for your account.
* Hit the back button to return to the main menu.
* Select “Get Blank Form”.The App will then connect to the server and provide a list of all forms in your account.

* Select the relevant form and proceed.

# Building the rest of the form for more functionality

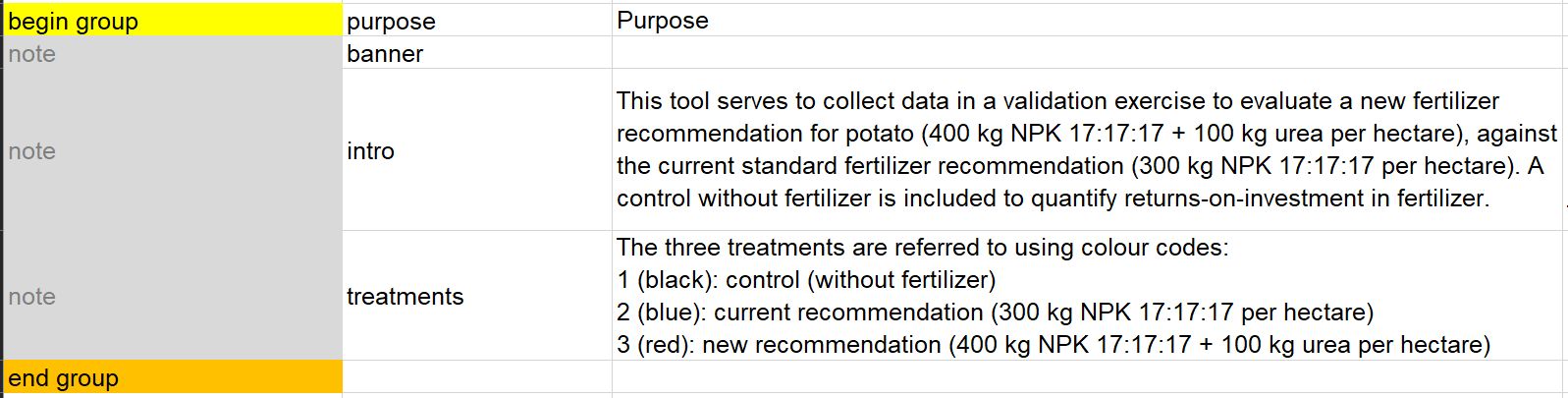
## A few tips for the ‘name’ column:

* When making up a variable name try to be both concise and descriptive.
* Avoid any variable name that begins with any special character or number or which contains any spaces in between. These will be flagged as errors by ODK Validate.
* The “name” column allows upto 32 characters but in order to anticipate data analysis with other statistical packages, limit the length of the variable names to 12.
* Consider using the «camelCase» when writing variable names and avoid underscores («\_») since some statistical packages read underscores as blanks.
* All “begin group” types must have a corresponding “end group”.

## Let’s piece together the rest of the form!

### Groups

Groups are used to put together related questions for data export and analysis. In our instance we have put together some notes to explain the purpose of the survey as seen below:

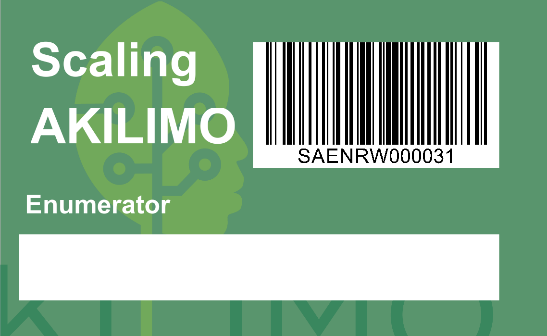


In the example, **Notes** are used to provide additional information and instructions to the user while not taking any input.

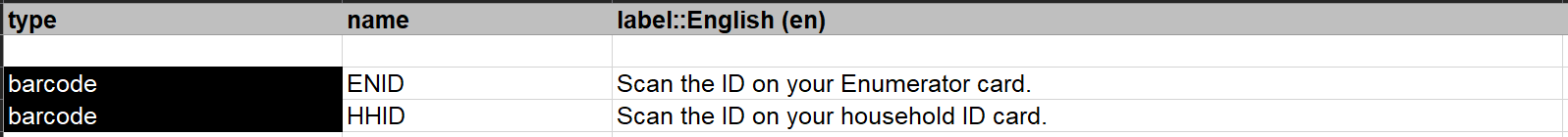
It is important to note that ‘end group’ doesn't require a name or label, as it will not be visible in the form. However, if groups don’t have an end group they will cause an error when uploading the form. Always ensure that each begin group has an end group.

### User information

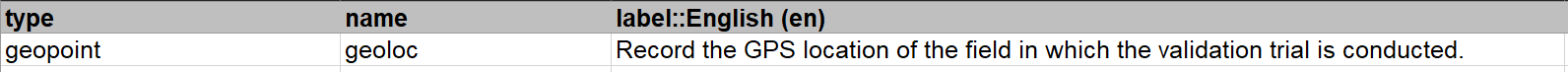
You need to record on which participating household you will collect data. You can use a barcode on an ID card.



See below how to include barcode question type to help collect user information in the ODK form



* Next you also need to always make sure where the data is being collected. We recommend that you always, at the start of the form, record the GPS location using geopoint.

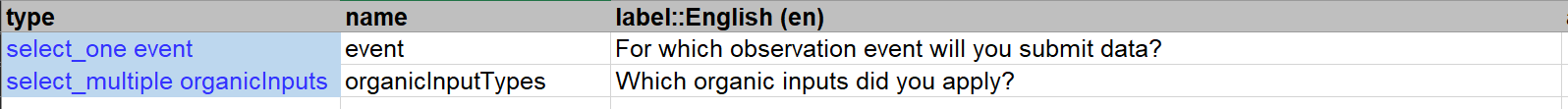


## Common question types for this kind of survey

There are many questions types in ODK but the common question types for this kind of survey are as listed here:

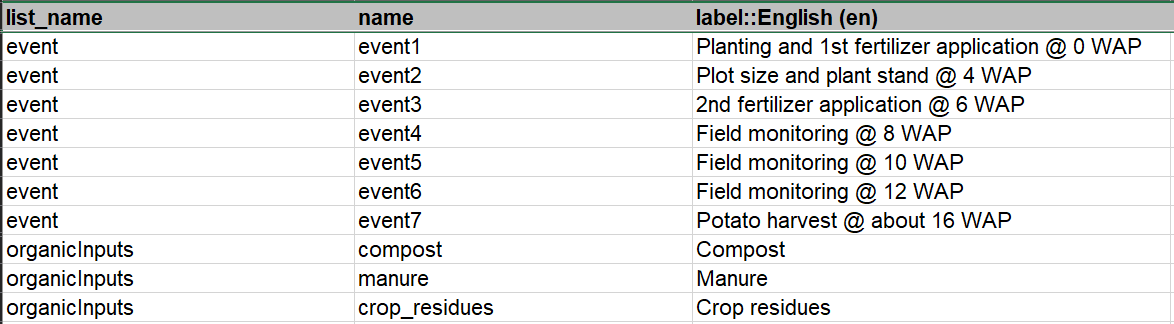
* **Multiple and single Select type.** A multiple select question type allows for selection of multiple choices for a single question whereas a single select question type restricts us to pick just one choice in the choice list.

In the ‘**survey’** worksheet under the ‘**type’** column you can write **select\_one/select\_multiple**, followed by the name of the choice list, in our case ‘event’. Then proceed to provide the variable name under ‘**name**’ and appropriate lable as below:

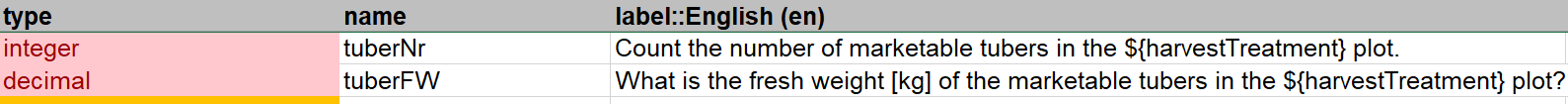


These are the three mandatory columns but additional constraints or relevancies can be added as required.

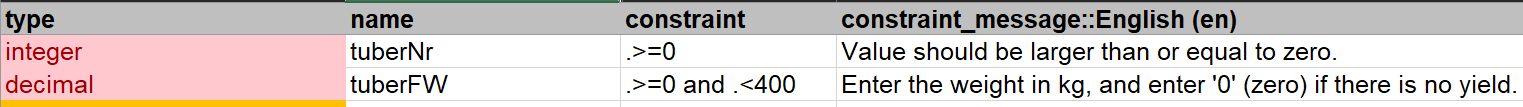
What follows is the population of the choice list with the list name we had chosen. Create a row for each of your answer choices, give variable names for each of the options and then provide appropriate labels as in the example given below:



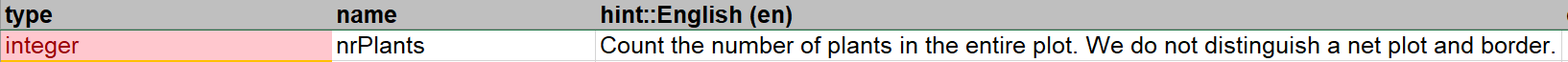
* **Integer/Decimal** are another common question type. The main difference being that integer questions only accept whole numbers while decimal allows for decimal numbers. See below example that shows differences:

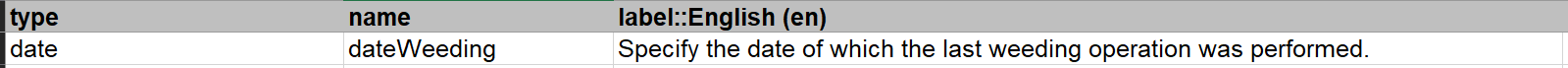


* In order to improve data quality and save time on data cleaning, you can add data **constraints column** then type in the formula specifying the limits on the answer. See example below where we constraint the above question type by providing a range of acceptable values. To note is that the ‘.’ in the formula refers back to the question variable. The constraint message gives more clarity on the constraint.



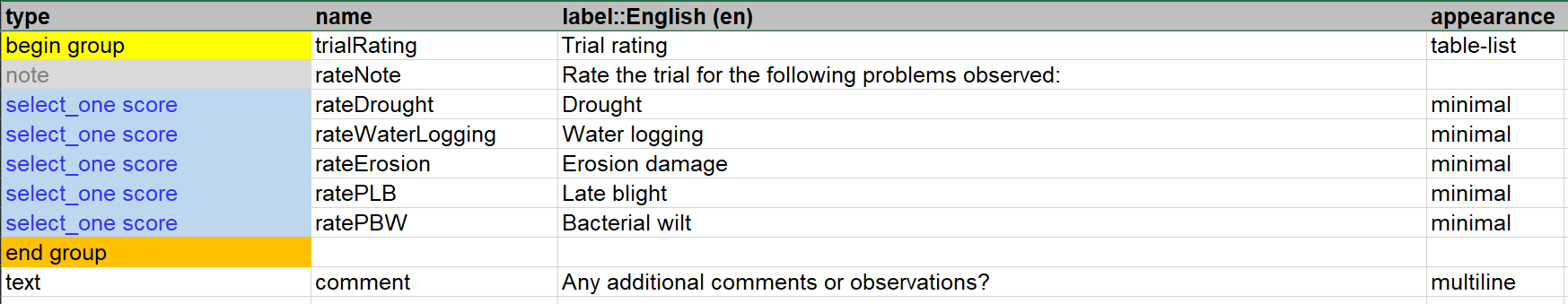
* For further clarification you can provide hints in the **hint column**:



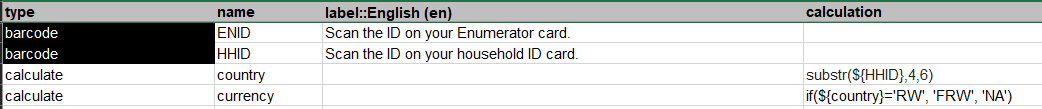
* The **date** question type is also common especially for validation exercises. See the example below, where the user is required to give a date when the second fertilizer application was conducted. 

There is an option to show a calendar format or have the question in a month- year format. In order to specify this, go to appearance column and include ‘no-calendar’ or ‘month-year’, whichever is applicable in your case.

* The **appearance** **column** allows you to alter the appearance of questions in your form. Below are more examples of the appearance column

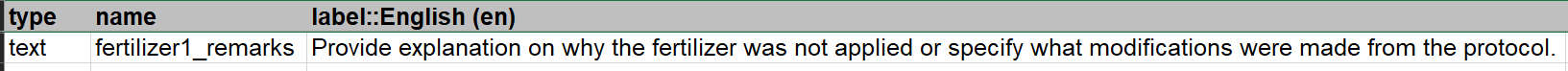


* We use the **calculate column** for performing calculations that would otherwise be conducted at the data analysis level. The calculate field can also be used to recode/create new variables as in our case where we create the variable ‘using a calculation of the unique identifiers ENID and HHID. This saves time and effort on the data analysis stage.



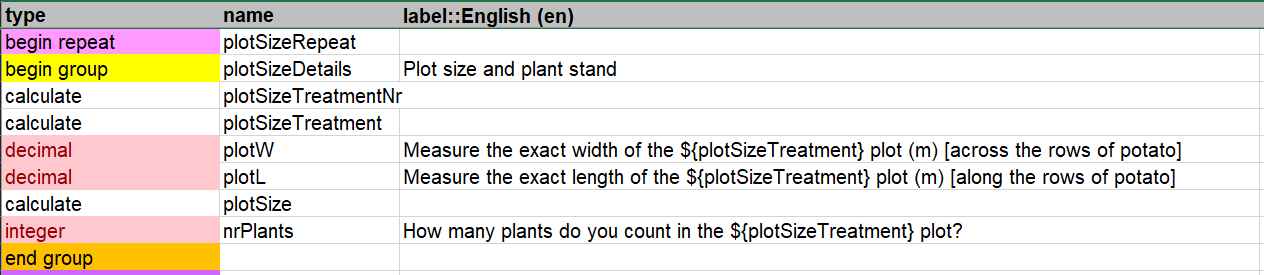
To use this column, create the “**calculate**” question type in the **type** column and then write your formula into the **calculation** column.

* **Text question type** are also a common question type although these question types should be limited unless collecting qualitative data, to avoid too much time being used for decoding in the data analysis stage:

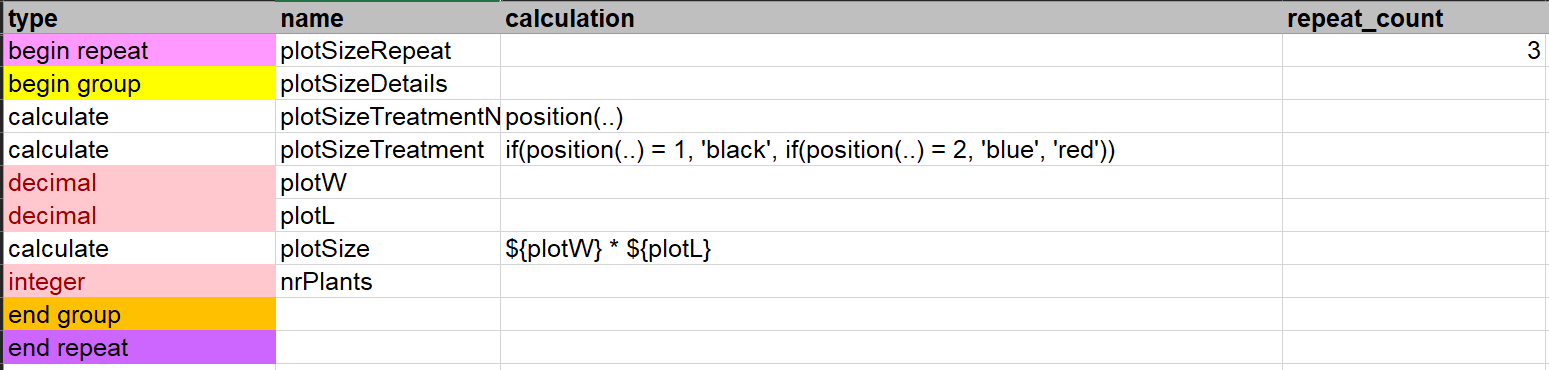


### Repeat groups

Repeat groups are used where the same set of questions is expected to be asked a number of times to collect same information for multiple people or items. Below is an example of a repeat group that counts the number of marketable tubers in several plots as well as records the fresh weight of the same.



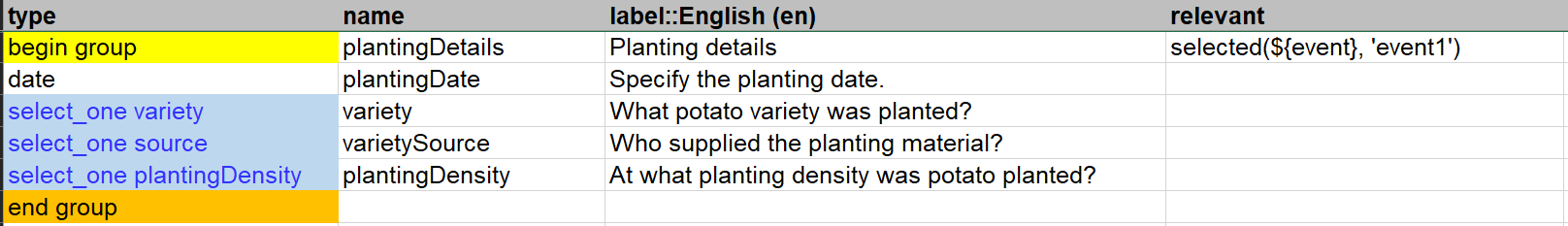
Take note of the **repeat\_count column** which specifies the number of repeats applicable. This column can also be customised to take in a value from one of the variables in case the number of repeats is not predefined.



To note also is the **calculation column** which uses the function **position(..)** placed outside the repeat group, to return the index of the last completed repeat.

### Skipping questions

You can use the ‘relevant’column to skip one question or alternatively if you want to skip a group of questions, you can put the relevant attribute at the start of a group as shown below. Here we use the selected option in the choices to show/skip some questions.



# Second exercise

## ‘MyCap’ Data Collection Tool

For the next exercise we will build an XLSForm called ‘MyCapTool’. This is an example of a data collection in an on-station 3-factorial experiment, where the plot information is read from a csv file using the **pull data function**. The information includes the plot identifier (repNr and plotNr) and the levels of the 3 factors in an RCBD with randomized plot numbers within rep.

### What does the pull data function do?

For a predefined set of data in csv format like in our case the treatMyCap.csv. If you wanted to be able to pull this information into the form you are going to use in the field you can apply this function.

### How do you create such a csv file?

You can create a csv file from scratch, but most times you may have collected this data before and with other variables that are not of interest for you. In this case you will need to trim this csv file to ensure that you don’t have unnecessary variable and so that the file does not take up too much memory in the mobile device.

In essence, the csv file should follow these rules:

1. The field names should be short and unique. See the tips we mentioned earlier for naming variables.
2. For identification of rows of interest, ensure that at least one of the columns contains unique identifiers. Tip: Use the word key in the field name as this will be allow for easier look up. In our case we used ‘lookupKey’.

We named our external csv file ‘treatMyCap’; Of interest to us are the 6 field names (lookupKey, plotNr, strain, levelP, variety). On the ona account, the csv file is uploaded under the media section of the MyCapTool survey. When you get the form on your mobile phone using ODK Collect it will come with the media file.

### Building the XLSForm

We start off by putting in place the now familiar metadata and then a note to introduce the purpose of the survey.

After this we form a group to specify the replicate number and to allow selection of parameters that will be used to collect data. All these are key variables to be used in the repeat group.

In order to determine the number of iterations, we include an integer type question to capture the number of plants on which canopy measurements will be taken. We provide relevant constraints; the number has to be more than or equal to 5. Note the relevant section which dictates that this repeat group is only applicable if the user selects the parameter ‘CM’.

### Now let’s pre-load some values!

1. The first thing we do is capture the plot number then add a calculate data type which combines (concatenates) the replicate number and plot number to form the lookupKey.
2. Next, for each field that we want pre-loaded into our tool we add a calculate data type and give an appropriate field name.

Under calculation column see how we call the pulldata() function e.g. for strain: **pulldata('treatMyCap', 'strain', 'lookupKey', ${lookupKey}).** Let’s break this down:

1. **'treatMyCap':** the csv file from which we are pulling data from.
2. **'strain':** the column that we want the data pulled from
3. **'lookupKey':** the column that acts as a key in the external csv files
4. **${lookupKey}:** the field in our tool that should match the key

In summary, we are pulling the value from the ‘strain’ column of the treatMyCap’ csv file, using the lookupKey field in our form to link to the specific row in the csv file’s lookupKey column.

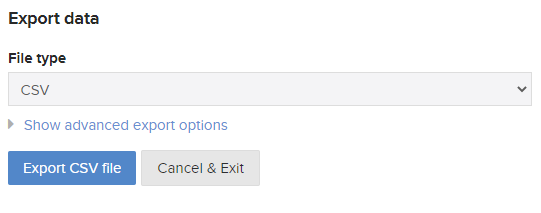
Now you can use the newly created ‘pull data fields’ in other fields in the survey; see line 19. You can also use the fields in the constraint, relevant columns etc by calling it using the $ sign as any other field in the survey.

## Downloading data from Ona for analysis

There are two ways of getting data from Ona:

### Export the data directly from ona in the Overview tab:

1. Click on ‘Prepare Data Export’
2. Select file type as .csv



1. Under ‘Advanced export options’ you can select applicable choices
2. Click Export CSV file
3. Proceed to download

### 2. Using ODK Briefcase as demonstrated [here](https://help.ona.io/knowledge-base/how-do-i-use-odk-briefcase/)