Adverse Events in REDCap

Authors: B. de Veer (ITHS) & Andy Arenson (IU)

Overview

REDCap was built on the assumption that it would be used to implement a known research protocol. REDCap assumes that the user knows exactly which variables are going to be collected when designing the project.

As a result, Adverse Events (or AE's) are hard to capture due to their unpredictable nature. Any participant could have one, five, 20, or no adverse events. Anticipating this in REDCap can be done in a number of different ways. We will outline the three most commonly used methods.

1. Pre-determined set of Adverse Events in a form

This method is by far the simplest and easiest to implement. You design a set of variables needed to capture all relevant information about the adverse events and repeat that set a number of times. A good rule of thumb is to multiple the expected number of AE's by two and put in that many sets. Label each set as Adverse Event 1, Adverse Event 2, and so forth. Older versions of REDCap tend to use this method. The main disadvantage of this method is that you will need to add additional sets of AE's if you exceed your allotted number of AE's.

Please see appendix A for an example of a set of Adverse Events variables.

2. Pre-determined set of Adverse Event forms in arms/events

The second method is most suited for longitudinal REDCap projects. It calls for a separate form or data collection instrument dedicated to AE's. This form can contain one set of AE variables or multiple ones depending on need. This AE form will then be associated with the relevant events in the REDCap project or warrant their own custom events reserved just for AE tracking.

Custom AE events have the disadvantage that you will need to put in variables to track where in your study protocol the AE occurred.

However, it is easier to add additional AE's by creating extra AE Events in any given arm and linking the AE form to it.

3. Separate AE project

The last method is the technically the most complex, but will give the user the most flexibility in the long run.

Instead of trying to integrate the AE tracking within the same project, you create a separate project dedicated solely to AE's. You create a simple form that contains only one AE variable set. Each record in this project will track a separate AE. A single participant can have

multiple records in the AE tracking project when they have multiple AE's over the course of a study.

The tricky part of this method is tying each AE record back to the correct study participant in the main study database. A simple solution could be to just enter the participants study ID manually in the AE tracking form.

A more elegant solution makes use of the Dynamic SQL field available in REDCap. Dynamic SQL fields use custom SQL code that looks at other projects in the REDCap installation and create a custom dropdown every time the form or survey gets loaded.

In this context, that means that adding another participant in the main study database creates a new entry in the Dynamic SQL dropdown in the AE tracking database.

Note: Please be aware that Dynamic SQL fields need to be added by a user with super user privileges.

Please see Appendix B for an example of the needed custom SQL code for such a dropdown.

Appendix A: Common AE Variables

- Start date
  Include time if the AE's are short-lived
- Stop date
  Include time if the AE's are short-lived
- Status (Ongoing or Resolved)
- Description
  This can be a free text field or an advance decision tree based on an existing terminology
- Protocol Related (Yes or No)
  Some AE's are caused by the study protocol, others are the result of existing comorbidities
- Level of Severity
  Most studies will need to report the most severe AE's to a local organization (usually an Institutional Review Board)
- Additional Notes

Appendix B: Dynamic SQL code Example

Example code of a Dynamic SQL pull, that looks at REDCap project 101 and creates a dropdown with the record ID from 101 as the raw value and label that combines the record ID and the last name. *(Ex. 1, 1, Doe ; 2, 2, Smith ; 3, 3, Johnson)*

```
select a.value, concat_ws(', ',a.value,b.value) as label
from
(select record, value from redcap_data where project_id=101 and field_name='record_id') a
JOIN
(select record, value from redcap_data where project_id=101 and field_name='last_name') b
ON
a.record=b.record
order by a.value
```