Missing Data Detection with `exportRecordsTyped`

2023-11-09

Contents

Introduction

The addition of `exportRecordsTyped` opened a great deal of flexibility and potential for customization when exporting data from REDCap and preparing them for analysis. The tasks of preparing data are broadly categorized into three phases

1. Missing Value Detection
2. Field Validation
3. Casting Data

This document will focus on missing data detection and customizations to fit the user's preferences.

```
library(redcapAPI)
url <- "https://redcap.vanderbilt.edu/api/" # Our institutions REDCap instance

unlockREDCap(c(rcon = "Sandbox"),
             envir = .GlobalEnv,
             keyring = "API_KEYs",
             url = url)
```

```
## <environment: R_GlobalEnv>
```

Missing Data Detection

Missing data detection operates in a similar manner to casting data (see `vignette("redcapAPI-casting-data", package = "redcapAPI")`). When data are exported from REDCap, the field type of each field is determined and an appropriate function is applied to the field to determine which, if any, values are missing. This section describes the default behavior of missing data detection and discusses how to customize these behaviors.

Default Detection Behavior

When testing fields for missing values, all field types are tested using the `isNAorBlank` function. This function identifies two values as missing data

- The R missing value `NA`
- Empty strings, i.e. `""`

The empty strings are treated as missing values because this is how REDCap represents missing values in the data export. When exporting data `exportRecordsTyped` will convert empty strings to `NA`. Conversely, `castForImport` will convert `NA` values to empty strings to prepare the data for import.

Customizing Missing Data Detection For a Field Type

Circumstances may arise where the user wishes to treat other values as missing. Consider a situation where medical records are being reviewed to identify the number of days elapsing between the date of a diagnosis and the date of the associated surgery. During the records review, researchers may be unable to determine the number of days because either the date of diagnosis is not recorded in the record or the date of surgery is not recorded. The researchers then decide to record `-99` when the date of diagnosis is not recorded and `-98` when the date of surgery is not recorded. While this information may be relevant to researchers in understanding why data are missing, the analysis data set is unconcerned with why the data are missing and needs to code both values as `NA`.

Detecting these values as missing data is done by first defining a function to include `-99` and `-98` as missing values. The function must return a logical value where `TRUE` indicates a missing value.

```
isMissingSpecial <- function(x, ...){
  is.na(x) | x == "" | x %in% c(-98, -99)
}

#########################################################################
# Use the default missing data detection
Rec <- exportRecordsTyped(rcon,
                          fields = c("days_between"))
Rec
```

```
##   record_id days_between
## 1         1           10
## 2         2           22
## 3         3          -98
## 4         4          -99
```

```
#########################################################################
# Use the custom missing data detection
Rec <- exportRecordsTyped(rcon,
                          fields = c("days_between"),
                          na = list(number = isMissingSpecial))
Rec
```

```
##   record_id days_between
## 1         1           10
## 2         2           22
## 3         3           NA
## 4         4           NA
```

Customizing Missing Data Detection For a Specific Field

Additional arguments may be passed to the missing value detection function in order to single out specific fields. This allows for missing value detection to be customized for each field, even if there are multiple fields within a field type. In this example, the `isMissingSpecialField` function only identifies `-98` and `-99` as missing values for `days_between`, but not for `days_between_duplicate`.

```
isMissingSpecialField <- function(x, field_name, ...){
  if (field_name == "days_between"){
    is.na(x) | x == "" | x %in% c(-98, -99)
  } else {
    isNAorBlank(x, ...)
  }
}

#########################################################################
# Use the custom missing data detection
Rec <- exportRecordsTyped(rcon,
                          fields = c("days_between",
                                     "days_between_duplicate"),
                          na = list(number = isMissingSpecialField))
Rec
```

```
##   record_id days_between days_between_duplicate
## 1         1           10                     10
## 2         2           22                     22
## 3         3           NA                    -98
## 4         4           NA                    -99
```

It is possible to customize missing data detection for any field type listed in the appendix. In this example, the `isMissingSpecialField` is extended to include the `dropdown_example` field in identifying the values `-98` and `-99` as missing data.

```
isMissingSpecialField <- function(x, field_name, ...){
  if (field_name %in% c("days_between",
                        "dropdown_example")){
    is.na(x) | x == "" | x %in% c("-98", "-99")
  } else {
```

```
    isNAorBlank(x, ...)
  }
}

#########################################################################
# Use the custom missing data detection
Rec <- exportRecordsTyped(rcon,
                          fields = c("days_between",
                                     "days_between_duplicate",
                                     "dropdown_example",
                                     "dropdown_example_duplicate"),
                          na = list(number = isMissingSpecialField,
                                    dropdown = isMissingSpecialField))
Rec
```

```
##   record_id days_between days_between_duplicate dropdown_example
## 1         1           10                     10         One week
## 2         2           22                     22      Three weeks
## 3         3           NA                    -98             <NA>
## 4         4           NA                    -99             <NA>
##   dropdown_example_duplicate
## 1                   One week
## 2                Three weeks
## 3       Missing Surgery Date
## 4     Missing Diagnosis Date
```

Frequently Asked Questions

Change the Default Missing Data Detection for All Field Types

How do I change the default missing data detection for all field types?

`redcapAPI` has an obscure function that will create a list of overrides for every field type. Use the `na_values` function to create the override list as illustrated below. (Yes, `na_values` takes a function as an argument)

```
customMissingDetection <- function(x, ...){
  is.na(x) | x == "" | x %in% c(-98, -99)
}

Rec <- exportRecordsTyped(rcon,
                          fields = c("days_between",
                                     "days_between_duplicate",
                                     "dropdown_example",
                                     "dropdown_example_duplicate"),
                          na = na_values(customMissingDetection))
Rec
```

```
##   record_id days_between days_between_duplicate dropdown_example
## 1         1           10                     10         One week
## 2         2           22                     22      Three weeks
## 3         3           NA                     NA             <NA>
## 4         4           NA                     NA             <NA>
##   dropdown_example_duplicate
## 1                   One week
## 2                Three weeks
## 3                       <NA>
## 4                       <NA>
```

Appendix

Missing Data Detection Field Types

- `calc`: Calculated fields.
- `checkbox`: Checkbox fields.
- `date_`: Text fields with the "Date" validation type.
- `datetime_`: Text fields with the "Datetime" validation type.
- `datetime_seconds_`: Text fields with the "Datetime with seconds" validation type.
- `dropdown`: Drop down multiple choice fields.
- `float`: Text fields with the "Number" validation type.
- `form_complete`: Fields automatically added by REDCap indicating the completion status of the form.
- `int`: Text fields with the "Integer" validation type. This appears to be a legacy type, and integer appears to be used by more recent version of REDCap.
- `integer`: Text fields with the "Integer" validation type.
- `number`: Text fields with the "Number" validation type.
- `number_1dp`: Text fields with the "number (1 decimal place)" validation type.
- `number_1dp_comma_decimal`: Text fields with the "number (1 decimal place - comma as decimal)" validation type.
- `number_2dp`: Text fields with the "number (2 decimal place)" validation type.
- `number_2dp_comma_decimal`: Text fields with the "number (2 decimal place - comma as decimal)" validation type.
- `radio`: Radio button fields.
- `select`: Possible alias for `dropdown` or `radio`.
- `sql`: Fields that use a SQL query to make a drop down tools from another project.
- `system`: Fields automatically provided by REDCap for the project. These include `redcap_event_name`, `redcap_data_access_group`, `redcap_repeat_instrument`, and `redcap_repeat_instance`.
- `time_mm_ss`: Text fields with the "Time (MM:SS)" validation type.
- `time_hh_mm_ss`: Text fields with the "Time (HH:MM:SS)" validation type.
- `truefalse`: True - False fields.
- `yesno`: Yes - No fields.