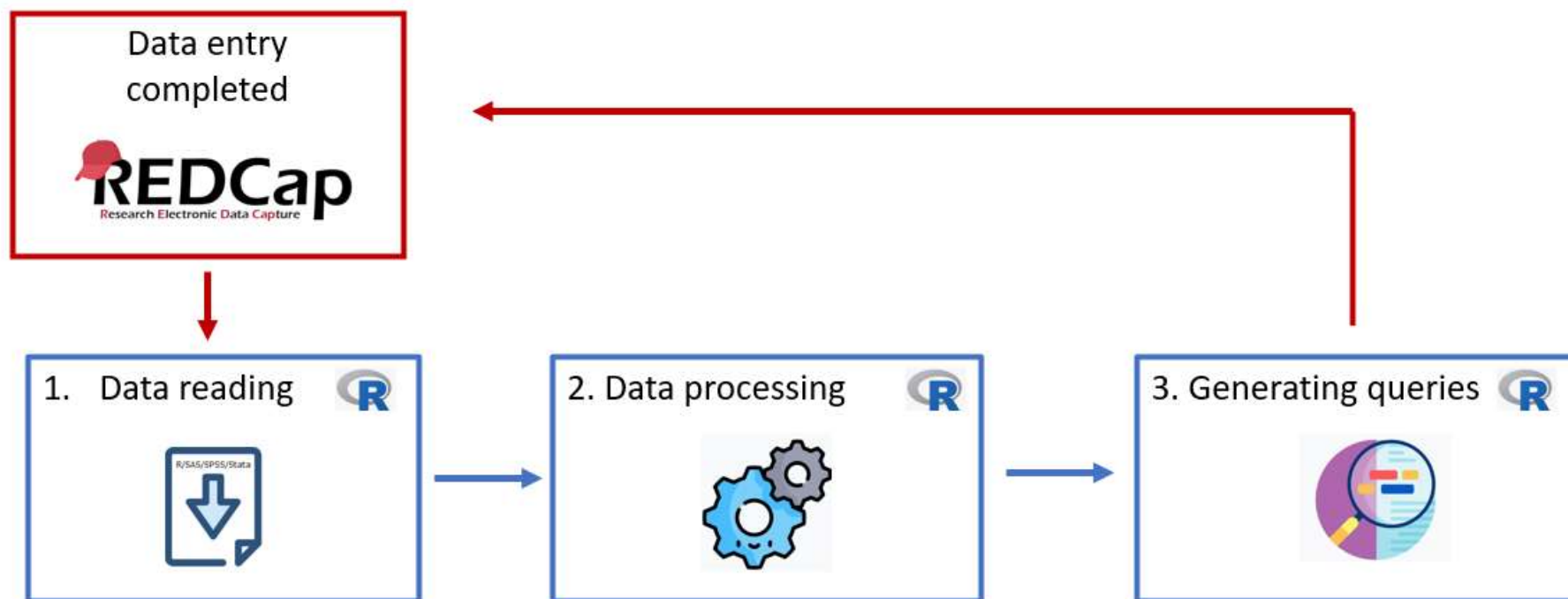


Managing REDCap Data: The R package REDCapDM

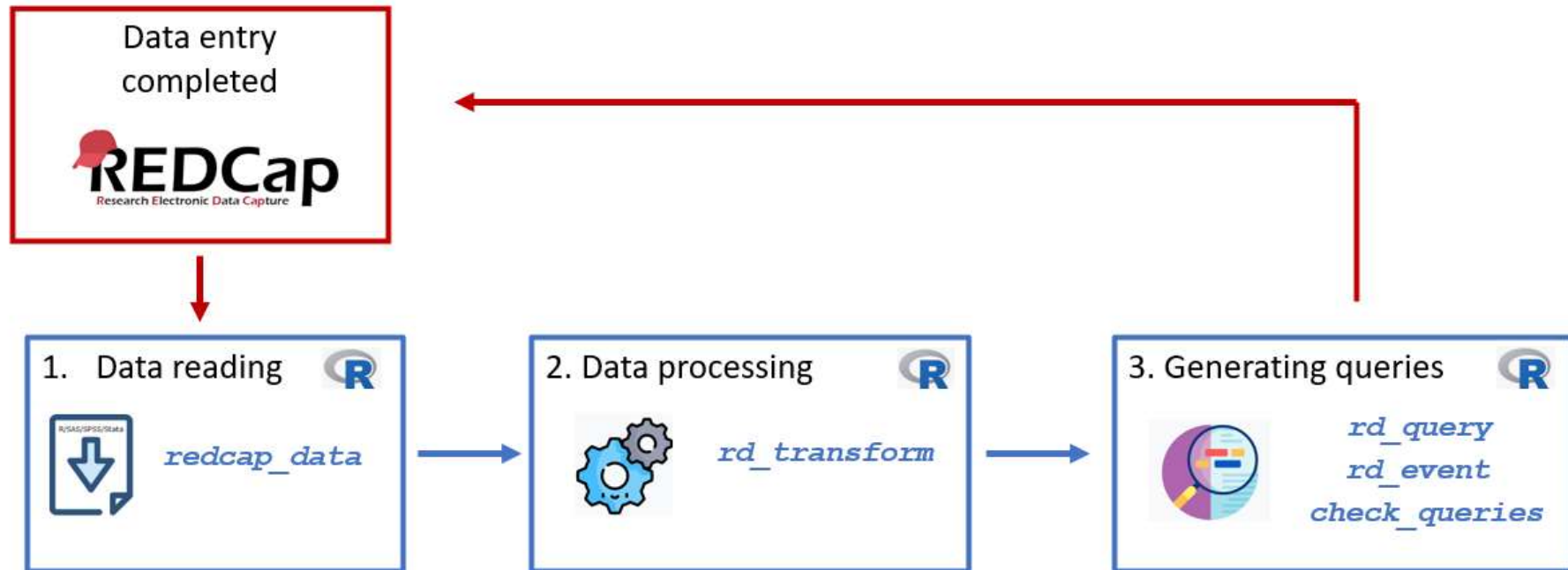
João Carmezim, Biostatistics Support and Research Unit
Germans Trias i Pujol Research Institute and Hospital (IGTP) - Badalona, Barcelona
R Medicine - Miami, June 2023

Motivation

Streamline and automate the processes that occur between completing data entry and obtaining a database ready for statistical analysis in :



Structure



1. Data reading

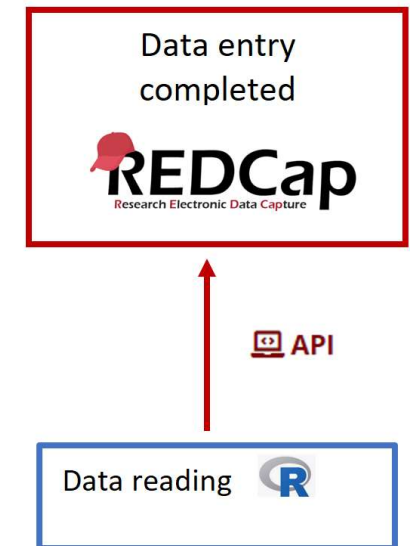
Data reading

(API connection)

1. Generate an API token for the project in REDCap
2. Use the `redcap_data` function:

```
dataset <- redcap_data(uri = "https://redcapdemo.vanderbilt.edu/api/",  
                      token = "1234567890ABCDEFGHIIJ")
```

Please note that the API token gives you special access to the REDCap project and should not be shared carelessly.



2. Process

Process

- As an example, we will use this initial dataset that was imported into R.
- We will then call the `rd_transform` function:

```
dataset_transformed <- rd_transform(data = dataset$data,  
                                   dic = dataset$dictionary)
```

```
str(dataset_transformed, max.level = 1)
```

List of 3

```
$ data      : 'data.frame': 99 obs. of  318 variables:  
$ dictionary: tibble [316 × 18] (S3: tbl_df/tbl/data.frame)  
$ results   : chr "1. Recalculating calculated fields and saving them as..."
```

3. Queries

Output

An R list object composed of two elements ↴

- Data frame with all identified queries:

Identifier	DAG	Event	Instrument	Field	Repetition	Description	Query	Code
268	.	.	Enrollment	is_homeless	Daily-1	Are you currently homeless?	The value is NA and it should not be missing	268-1

- Summary of the total number of queries per variable:

Variables	Description	Event	Query	Total
is_homeless	Are you currently homeless?	.	The value should not be missing	1

Generate queries

(Identifying missings)

Information required to properly use the `rd_query` function:

```
example <- rd_query(dataset_transformed,  
  variables = "da_is_mobile",  
  expression = "is.na(x)")
```

List of 2

```
$ queries: tibble [34 × 9] (S3: tbl_df/tbl/data.frame)  
$ results: 'kableExtra' chr [1:5] "" | __truncated__ "-" "-" "-" ...  
..- attr(*, "format")= chr "html"
```

`example$results`

Variables	Description	Event	Query	Total
da_is_mobile	Daily Location	.	The value should not be missing	34

Generate queries

(Specific queries)

If, instead of missing values, we want to identify inconsistent values:

```
example <- rd_query(variables = "age_first_drug_injection",
                    expression = "x < 35 | x > 80",
                    dic = dataset_transformed$dictionary,
                    data = dataset_transformed$data)
```

example\$results

Variables	Description	Event	Query	Total
age_first_drug_injection	How old were you the first time you injected an...	.	The value should not be less than 35 or greater than 80	29

Generate queries

(Function versatility)

So far, we have only seen examples of a single query for a single variable. However, we can make multiple queries at the same time:

```
example <- rd_query(variables = c("da_narcans_total", "age_at_enrollment", "race"),  
  expression = c("x < 2", "x > 50 & x < 75", "is.na(x)"),  
  dic = dataset_transformed$dictionary,  
  data = dataset_transformed$data)
```

Or apply the same query to several variables:

```
example <- rd_query(variables = c("da_narcans_total", "age_at_enrollment", "race"),  
  expression = c("is.na(x)"),  
  dic = dataset_transformed$dictionary,  
  data = dataset_transformed$data)
```

Generate queries

(Arguments - filter)

The **filter** argument allows us to apply a filter to the data set:

```
example <- rd_query(dataset_transformed,  
  variables = "is_vaccinated_covid",  
  expression = "is.na(x)",  
  filter = "is_homeless == 'Yes'")
```

example\$results

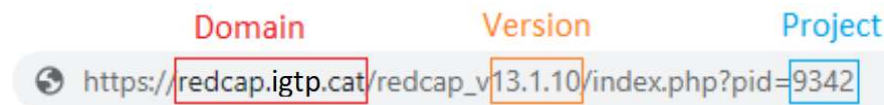
Variables	Description	Event	Query	Total
is_vaccinated_covid	Have you been vaccinated for COVID-19?	.	The value should not missing	9

Generate queries

(Other arguments)

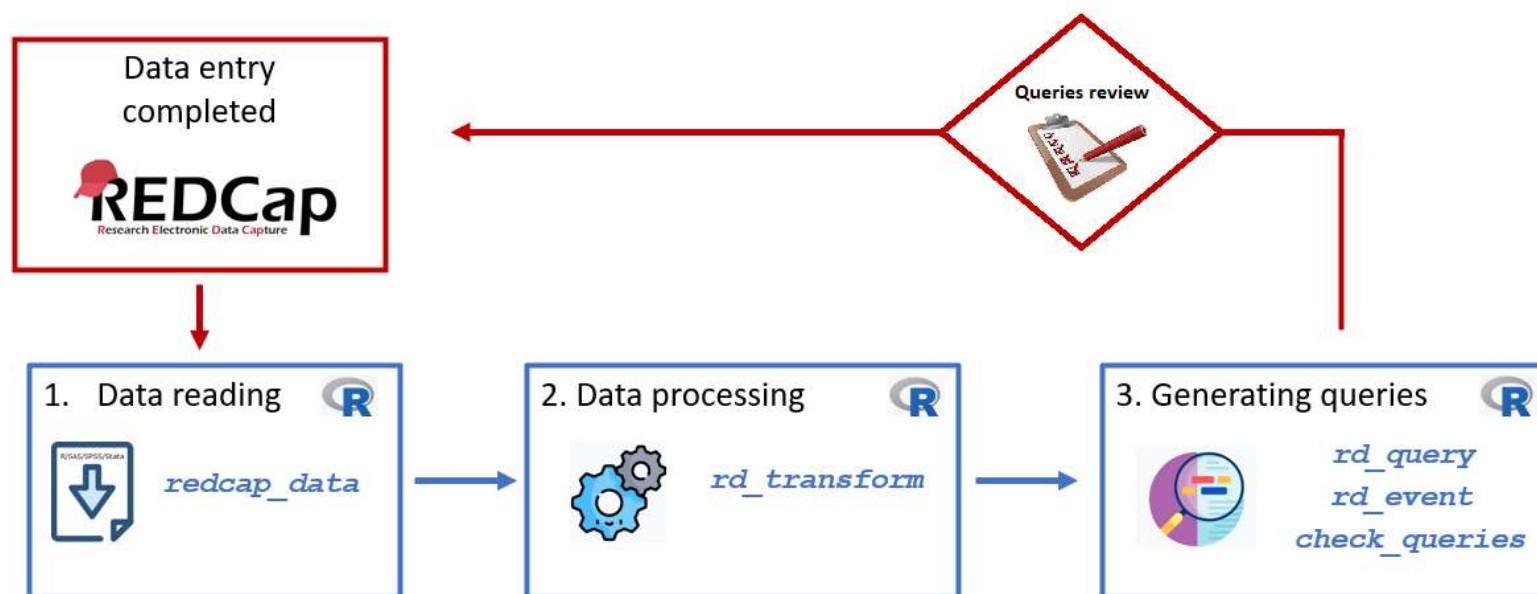
Arguments	Description
negate	Negation of the defined expression
addTo	Previous query data frame to which we will add the newly generated one
report_zeros	Includes variables with no queries in the summary element
by_dag	Group both elements of the output based on the data access groups (DAGs) of the REDCap project
link	List containing project information used to create a web link to each query

Link:

The image shows a browser address bar with the URL `https://redcap.igtp.cat/redcap_v13.1.10/index.php?pid=9342`. Above the URL, three labels are positioned: 'Domain' in red above 'redcap.igtp.cat', 'Version' in orange above '13.1.10', and 'Project' in blue above '9342'. Each of these three components in the URL is enclosed in a thin rectangular box of the corresponding color (red, orange, and blue respectively).

Follow-up of queries

Once the identification process is finished, the typical procedure would be to review all queries and, once reviewed, re-run the generation step, thus creating a new dataset of queries.



Follow-up of queries

To perform a follow-up on the generated queries, we can use the `check_queries` function:

```
check <- check_queries(old = example$queries,  
                       new = new_example$queries)
```

The function compares the old query dataset with the newly created one and classifies each query as new, solved, miscorrected or pending.

Warning: queries classified as *Miscorrected* belong to the same combination of record identifier and variable name in both the old and new datasets, but with a different reason.

Follow-up of queries

(Output)

Still a list with the same 2 elements, but ↴

- There is an additional column in the query data set that shows the classification of each query:

Identifier	DAG	Event	Instrument	Field	Repetition	Description	Query	Code	Modification
7	.	.	Enrollment	race	Daily-1	Race	The value is NA and it should not be missing	7-1	Pending

- The summary now shows the number of queries by category:

State	Total
Solved	105
Pending	9
Miscorrected	1
New	0

Limitations and prospects

Limitations and prospects

Limitations:

- The latest version of the package was tested on REDCap version 12.4.17.
- It was not tested on all possible REDCap project structures.
- Data processing may not be able to handle complex REDCap logic.

Prospects:

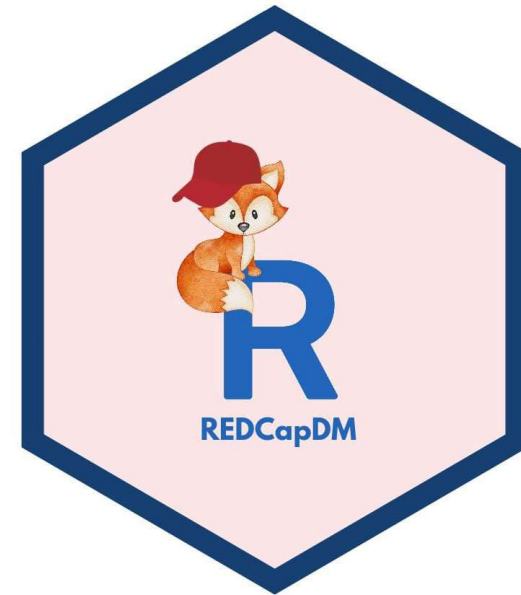
- Attempt to connect the query system generated in R to the existing REDCap system.
- Develop a Shiny application to complement the package.

Conclusions

Conclusions

The REDCapDM package:

- Provides a set of useful functions for importing, organising and checking the quality of REDCap data.
- Facilitates a study's data management workflow and helps ensure reliable, high quality data that is ready for analysis.
- Fills a gap in the tools available in R for query generation of REDCap data.



Vignette

Link: <https://ubidi.github.io/REDCapDM/articles/REDCapDM.html>

REDCapDM 0.7.0 Get started Reference Changelog

Search for



REDCapDM

Source: [vignettes/REDCapDM.Rmd](#)



On this page

Introduction

Functions

Built-in dataset

Examples

Introduction

The REDCapDM package allows users to read data exported directly from REDCap or via API connection. It also allows users to process the previously downloaded data, create reports of queries such as outliers or missing values and track the identified queries.

References

- Carmezim J, Peñafiel J, Satorra P, García E, Pallarés N, Tebé C (2022). REDCapDM: 'REDCap' Data Management. R package - version 0.7.0, <https://CRAN.R-project.org/package=REDCapDM>
-
- R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, - Vienna, Austria. URL <https://www.R-project.org/>.
 - Wickham H, François R, Henry L, Müller K (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10, - <https://CRAN.R-project.org/package=dplyr>.
 - Will Beasley (2022). REDCapR: Interaction Between R and REDCap. R package version 1.1.0, - <https://CRAN.R-project.org/package=REDCapR>
 - Wickham H (2022). stringr: Simple, Consistent Wrappers for Common String Operations. <https://stringr.tidyverse.org>, <https://github.com/tidyverse/stringr>.
 - Wickham H, Henry L (2023). purrr: Functional Programming Tools. <https://purrr.tidyverse.org/>, <https://github.com/tidyverse/purrr>.