REDCapSync: Encapsulated REDCap projects for pipelines, functions, and applications in R

by Brandon Rose and Natalie Goulett

Abstract Many R packages exist for working with the various API endpoints but none have been established for an encapsulated "get-everything" work flow that produces a standardized R object. REDCapSync accomplishes this and can serve as a dynamic input object for downstream functions.

1 Abstract

- Brief description of REDCap as a dominant data capture platform in clinical research

- Limitations of existing R packages interfacing with the REDCap API

- Introduction of *REDCapSync* as an encapsulated, state-aware representation of REDCap projects in R

- Key contributions:
  - Project-level encapsulation of metadata and data
  - Incremental synchronization rather than full re-downloads
  - Export of structured, audit-friendly Excel workbooks with advanced features (e.g., deidentification)
  - Designed for pipelines, analysis functions, and Shiny applications

- Summary of current capabilities and future roadmap (bi-directional sync)

2 Introduction

Some REDCap API packages include **REDCapR**, **redcapAPI**, **tidyREDCap**…

Figure 1: REDCapSync: Encapsulated REDCap projects for pipelines, functions, and applications in R

One of the reasons people do not choose more complicated structures in REDCap, such as repeating instruments is the data exports require a manual or automated process to merge the data into a usable format. For example if you have non-repeating data in one form...

Enter REDCapSync ...

2.1 REDCap in Clinical and Translational Research

- Widespread use of REDCap for:
  - Clinical trials

  - Observational cohorts

  - Registries and quality improvement

- Strengths of REDCap (governance, compliance, structured metadata)

- Common downstream needs in R:
  - Analysis pipelines

  - Reporting

  - Visualization and applications

2.2 The Gap Between REDCap and R Pipelines

- Typical REDCap-to-R workflows:
  - Stateless API pulls
  - Flat data frames
  - Manual post-processing
- Consequences:
  - Redundant API calls

  - Loss of project structure

  - Fragile downstream code

  - Poor support for reproducibility and automation

3 Key Design Choices

- No API token stored anywhere except user-defined renviron, not in cache, project object, global environment, or code.
- deidentify Excel outputs by default

4 Hypothetical

You are working on a research project collecting surveys from hundreds of participants. You are writing code that makes a visual.

5 Who is REDCapSync for?

6 Related Work and Limitations of Existing Packages

6.1 Existing REDCap R Packages

- API wrappers

- Data-frame–oriented imports

- Metadata access utilities

6.2 Common Design Pattern: Stateless Data Access

- Existing tools primarily:
    - Fetch data on demand
    - Return tables or lists
    - Leave project structure implicit
- Lack of:
    - Encapsulation

    - Persistent project state

    - Awareness of prior syncs

6.3 Why Encapsulation Matters

- Contrast with object-oriented or stateful designs in:
    - Database interfaces

    - Data pipelines

    - ETL frameworks

- Motivation for a higher-level abstraction representing a **REDCap project as an object**

7 Design Philosophy of REDCapSync

7.1 Encapsulated REDCap Projects

- Core concept: a REDCap project is represented as a **single structured R object**
- Encapsulation of:
    - Project metadata

    - Instruments and fields

    - Records and events

    - Sync state and provenance

7.2 Separation of Concerns

- Clear distinction between:
  - Data retrieval
  - Transformation
  - Export and downstream usage

- Benefits for:
  - Reproducibility
  - Testing
  - Modularity

7.3 Designed for Pipelines, Not Just Data Access

- Object persists across:
  - Scripts
  - Functions
  - Shiny applications

- Enables consistent behavior across contexts

8 Incremental Synchronization

8.1 Motivation

- Large projects and frequent data updates
- Inefficiency of full data re-downloads

8.2 Selective Updating

- REDCapSync tracks:
  - What has already been retrieved
  - Which components need refreshing

- Supports:
  - Metadata-only updates
  - Data-only updates
  - Targeted form or event updates

8.3 Benefits

- Reduced API load

- Faster iteration

- Improved stability in automated workflows

9 Structured Excel Export as a First-Class Feature

9.1 Excel as an Interoperability Layer

- Excel remains a dominant format for:
    - Clinical teams
    - Data managers
    - Regulatory workflows

9.2 Custom Excel Workbooks

- REDCapSync exports:
    - Multiple sheets reflecting project structure
    - Consistent naming and formatting

- Preserves:
    - Metadata context
    - Variable labels
    - Choice mappings

9.3 Advanced Features

- Built-in support for:
    - Deidentification strategies
    - Derived fields
    - Project-specific annotations

- Designed for:
    - Review
    - Sharing
    - Archival

10   Applications

10.1 Clinical Trials

- Ongoing data monitoring

- Interim analyses

- Regulatory-compliant exports

10.2 Observational and Registry Studies

- Longitudinal data handling

- Incremental updates

- Consistent downstream datasets

10.3 Shiny Applications and Dashboards

- Live, synchronized project objects

- Reduced logic duplication

- Stable data interfaces

11   Implementation

11.1 Package Architecture

- Internal object structure

- Use of R lists / classes

- Dependency philosophy

11.2 Reproducibility and Transparency

- Explicit sync steps

- Inspectable internal state

- Alignment with reproducible research practices

12   Comparison with Existing Approaches

12.1 Feature Comparison

- Stateless API wrappers vs REDCapSync

- Encapsulation

- Incremental sync

- Structured Excel export

- Pipeline integration

12.2 When to Use REDCapSync

- Not a replacement for all REDCap tools

- Best suited for:
    - Long-lived projects
    - Repeated analyses
    - Applications and automated pipelines

13 Future Directions

13.1 Bi-Directional Synchronization

- Uploading data from previously exported Excel workbooks

- Validation against metadata

- Controlled updates back to REDCap

13.2 Extended Deidentification and Governance Features

- Project-level policies

- Role-based exports

13.3 Ecosystem Integration

- Downstream analysis packages

- Shiny app wrappers

- File-system–based project organization

14 Conclusion

- REDCapSync introduces a new abstraction for working with REDCap in R

- Encapsulation enables:
    - More robust pipelines
    - Better reproducibility
    - Reduced cognitive and technical overhead

- Positions REDCap projects as durable, inspectable R objects rather than transient API responses

15 Acknowlegements

Brandon Rose
University of Texas Southwestern Medical Center
Hematology/Oncology Department
Dallas, TX, USA
https://www.thecodingdocs.com/
ORCiD: 0009-0009-7813-1960
TheCodingDocs@gmail.com

Natalie Goulett

fakeexample@gmail.com