

MENU

Joy Payton

November 29, 2018

TAGS r, python, REDCap

Using the REDCap API

We've [talked before on this site](#) about the usefulness of APIs (Application Programming Interfaces) in data analysis. Let's apply those benefits to data stored in REDCap!

API Overview

There are two principal benefits to using APIs:

- Data freshness
- Scripted reproducibility

Data Freshness

With API calls, you get your data fresh every time, instead of relying on potentially stale downloads. Reaching into your file system to get a .csv or .json file to analyze can be tricky if you have multiple versions, and it's easy to do analysis on an obsolete file if you're not careful. For example, let's say you have to run some analysis on data you're collecting in REDCap

obscure me if you're not careful. For example, let's say you have to run some analysis on data you're collecting in REDCap, and you want to re-run this analysis every couple of weeks to see the latest figures. One way to do that is to manually export data to a .csv and save it to a file that your R or Python script will then open and work with. But REDCap likes to download files with a date stamp as part of the file name, so you have to remember to either change the name of the file to whatever standard file name you want your script to use, or change the file name each time in your script. You may also end up collecting multiple .csvs in various directories, each of which has a particular version of the data in REDCap. This can easily become overwhelming and cause confusion.

What's a better approach? Reach into the REDCap database directly each time you run your script, so that you know you're using the most up-to-date data. Read on to learn how to do this!

Scripted Reproducibility

Another problem with using downloaded files from REDCap is that this method requires unscripted, point-and-click manual work. If you were to document this carefully, you'd have to give several steps, like where to log in to REDCap, which project to use, the .csv download settings (which fields or forms to download), file naming conventions, and where to put the file. Most of us don't go into this level of detail in our manual workflows, for good reason! It's tiresome, and we know that sometimes things change in the look and feel of a website, so including screenshots and detailed instructions about where to look for a link or how to highlight multiple fields is a lot of work for something that might have slightly different steps next week or next month.

A better approach is to use a script that uses an API call. First of all, it's scripted, which means no manual steps to write up in a Word document or add to a GitHub repo. Also, the typical API has a standard interface that will rarely, if ever, change. API access may improve over time, adding new features, but it's very infrequent that an API will radically change and remove options, rendering your script unusable. The same half-dozen lines of code you use to access your data will almost always be stable for months or years, and if you do need to change it, you're only changing that small chunk of code, instead of a step-by-step document with words and images that guide a manual effort.

API Keys

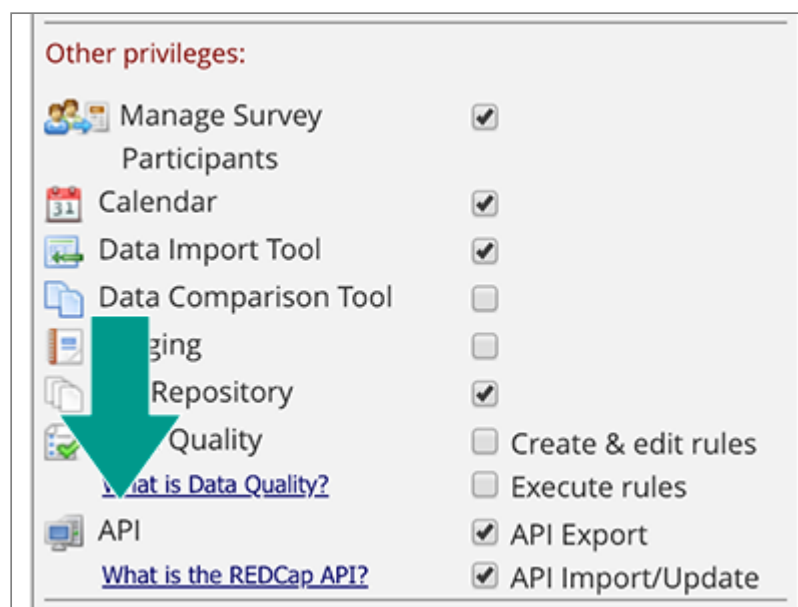
APIs are automated, which means they won't rely on you logging in manually, adding a user name and password. API calls have to run without human intervention, which means you need to provide your script with credentials that show you are allowed to log in and see the data you're accessing. But obviously you don't want to put your user name and password in a script. Your user name and password open a lot of doors at CHOP, from your email to your payroll information to Epic. You want to isolate just your access to this particular data, and using your all-powerful login information to access REDCap data is

...to create just your access to the particular data, and using your API key to get information to access REDCap data is far too powerful. What if your credentials fell into the wrong hands, because they were in a script on a drive that many people have access to? This is where API keys or tokens come in. API credentials give very specific access to very specific things, and can be regenerated easily, in case you suspect they may have been lost or misused. If API credentials do fall into the wrong hands, it's not great, but it's much better than accidentally sharing your CHOP username and password! Different APIs have different methods for giving you the key to your data or application, and different methods for supplying that credential when your program tries to make contact. Here, we'll talk specifically about how REDCap handles this.

In REDCap, the project owner (or people with user rights) have to explicitly give permission to use the REDCap API. In fact, when you create a new REDCap project, REDCap does not provide you with these permissions, even though you're the project owner. Take a look at your REDCap projects and look on the left hand side of the screen to look at the menu of options. Chances are, you won't see anything that says "API". If that's the case, don't worry. We're going to take a project you own (or, you could quickly create a project and put some fake data in it) and walk you through how to give yourself API access.

Giving Yourself Access

Pick a REDCap database that belongs to you or in which you have the ability to change user rights. When you go into User Rights, look up your name (or your role) and edit the user rights. Give yourself both kinds of API access: API Export and API Import/Update, as shown in the image below.



Then hit “Save Changes”. Refresh your browser (reload the page) so that your new permissions are included in what you get shown in your project. Now, on the left hand side of the project, you should see something new under “Applications” – “API” and “API Playground”.



So now you have the **right** to use the API, but you can't start using it yet. You have to generate an API token. Click on the “API” application and then click on the button to “Generate API Token.”

Obtain API token for **Your Project Name**

Use the button below to generate an API token for this project. You will need a different token for each project you would like to access.


Generate API token


The API token is unique to the combination of user and project. It's a code that allows access **only** to the data in a single project and **only** the data that the person who generated the API token is allowed to see. Importantly, if you feel like you may have accidentally given your API token away, it's a good idea to regenerate it, which is a single-click operation. It makes your old token invalid and creates a new token.

Soon, your API token will show up in that same "API" Application. It will look something like this (note, I've obscured my token – sharing this in a post like this one would be a terrible idea!).

Your API token for project **Your Project Name**

The API token below is **ONLY for you** and will work **ONLY with this project**. This token allows special access to REDCap data and **should NOT be shared with others**. If you think your token has been compromised, then please contact your REDCap administrator immediately **AND** either delete or regenerate your token by using the buttons below.

 **API Token:**



Delete token

Finished using the API for this project? If so, please delete your token for security reasons.

Regenerate token

Think someone else knows your token? If so, please regenerate your token for security reasons.

You'll use this token when you request data in your R or Python script.

Rehearsing your API Calls in the API Playground

So, you have some data in a REDCap database, and you have an API key that gives you access to that data. How do you **use** this key, though? Learning how to use an API can be tricky, because it requires you to pass not only the key, but your actual **request** (like "give me only the field `beck_depression_inventory` or "give me only the form called `inclusion_criteria` "). Luckily, REDCap provides you an "API Playground" that gives you not only a view into what the API can do, but provides a good start to giving you the actual code. Let's take a deeper look

good start to giving you the actual code. Let's take a deeper look.

The API Playground has several parts:

- The menu-driven selection box
- Raw Request Parameters
- Response
- Code Examples

Let's look at how each part of the API Playground helps you learn the API.

In your REDCap project, head over to the “API Playground” – click on that phrase in the Applications pane on the left. At the top, you'll see the menu-driven selection box. The first selection you have to make is API Method. For now, choose “Export Project Info”, and in the next two menus, choose “CSV”.

Select an API method from the drop-down list below, after which it will load any other options that are specific to that method.

API Method:

Production - Some APIs have been disabled for the playground because they affect data

Format:

Errors:

The “Raw Request Parameters” box below your selection will change to reflect whatever you chose. Here's an example of what you should see (note, I've obscured my actual key):

Raw Request Parameters

Displayed in the box below are all the POST parameters that would be sent in the API request based on the selections above.

```
token:   
content: project  
format: csv  
returnFormat: csv
```

This “Raw Request Parameters” gives you a quick look at the information you’re passing to REDCap, to make sure you selected what you really wanted in the menu part above.

Look further down the page, in the “Response” area, and click on the button that says “Execute Request”. You might get a “waiting” spinner, and then a box will appear below the button with the data that REDCap returned from your request.

You requested your data to be in a .csv, so you should get some data that’s “comma separated” – a bunch of fields that are separated by commas, with each new line of data being separated by a line break. In the “Response” box, all of this is presented in plain text, not in a table, so it might look confusing or overwhelming. If you want to, you can copy that plain text and paste it into a text editor, saving it with the .csv extension. That will allow you to then open it in Excel to see if the .csv is what you intended. Below, here’s the “before” (comma separated text in plain text) and the “after” (text saved as a .csv and then opened in Excel) of one of my own projects:

↓ **Response**

Click the Execute Request button to execute a real API request, and it will display the API response in a text box below.

↑ **Execute Request**

```
project_id,project_title,creation_time,production_time,in_production,project_language,purpose,
purpose_other,project_notes,custom_record_label,secondary_unique_field,is_longitudinal,s
urveys_enabled,scheduling_enabled,record_autonumbering_enabled,randomization_enabled,ddp_en
abled,project_irb_number,project_grant_number,project_pi_firstname,project_pi_lastname,dis
play_today_now_button,has_repeating_instruments_or_events
24740,"Arcus Training Feedback","2018-12-07 14:19:34","2018-12-07
15:07:55",1,English,3,,,,,0,1,0,1,0,0,,,,,1,0
```

	A	B	C	D	E	F	G	H	I	J	K
1	project_id	project_title	creation_time	production_time	in_production	project_language	purpose	purpose_other	project_notes	custom_record_label	secondary_unique_field
2	24740	Arcus Training Fee	12/7/2018 14:19	12/7/2018 15:07	1	English	3				
3											
4											
5											



Below that box, you'll see an HTTP status. You want that to be 200, which means no errors occurred.

This is an example of “trying out” the API without having to write code. You experiment with various methods, like “Export Records”, to get data, add the appropriate criteria, such as the form(s) you want to download, and set some parameters for how you want REDCap to give you that data back (for example, in .csv format, or in .json format). In some API methods (like “Export Records”, you’ll have several drop down menus you can choose from, including forms and fields. You can make multiple choices by using control or command + click, or, in some cases, if you want every item listed (say, you want every form and every field), you don’t click anything at all, and REDCap assumes you mean everything. Practice makes perfect! Go

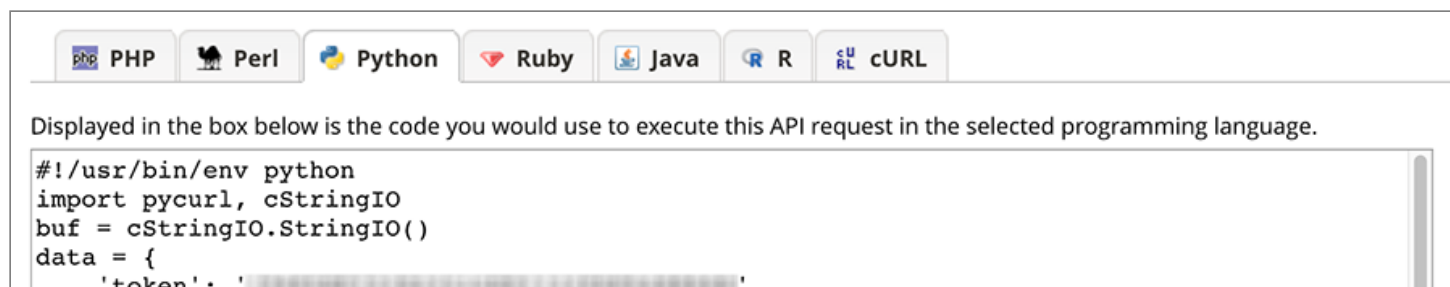
ahead and try out a few API calls using the Playground, including trying to access the data you want to pull into an analysis script. Once you feel comfortable with the various API methods and their output, go ahead to the next section.

Writing Code to Access Data

So, you've played around in the API Playground and you think you know the kind of API call you want to make. Now, how do you automate this selection so that your R or Python code will issue exactly the same request using code like you just did using menus? REDCap's API playground comes to the rescue again, with code snippets that show how to write the code to do automatically what you just did manually.

Let's take a look at some simple code snippets for a really easy REDCap API request – “Export Project Info.” That just gives me some metadata about a REDCap project, like the owner of the project and the project number. As before, I've chosen the API Method and set both the output and errors to be .csv files. When I do this, some code snippets are generated below. Let's look at the Python and R versions (my token is obscured for obvious reasons!).

Here's Python:



(take this from the `setopt ... URL` line). It'll probably be the same, but if it changes, trust what REDCap suggests.

Template:

```
import requests
data = {
    # Some stuff here
}
r = requests.post(# Some URL here #, data)
r.text
```

So here's the "before" code that REDCap provides, and the "after" code I propose:

```
1 #!/usr/bin/env python
2 import pycurl, cStringIO
3 buf = cStringIO.StringIO()
4 data = {
5     'token': '[REDACTED]',
6     'content': 'project',
7     'format': 'csv',
8     'returnFormat': 'csv'
9 }
10 ch = pycurl.Curl()
11 ch.setopt(ch.URL, 'https://redcap.chop.edu/api/')
12 ch.setopt(ch.HTTPPOST, data.items())
13 ch.setopt(ch.WRITEFUNCTION, buf.write)
14 ch.perform()
15 ch.close()
16 print buf.getvalue()
17 buf.close()
```

The diagram illustrates the transformation of a pycurl script into a requests script. Two large green arrows point from the left code block to the right one. The first arrow points from line 12 of the left code (`ch.setopt(ch.HTTPPOST, data.items())`) to line 8 of the right code (`r = requests.post(...)`). The second arrow points from line 13 of the left code (`ch.setopt(ch.WRITEFUNCTION, buf.write)`) to line 9 of the right code (`r.text`).

```
1 import requests
2 data = {
3     'token': '[REDACTED]',
4     'content': 'project',
5     'format': 'csv',
6     'returnFormat': 'csv'
7 }
8 r = requests.post("https://redcap.chop.edu/api/", data)
9 r.text
```

What you see as a result of this is just a string that you get back (similar to how you just saw the string in the “results” box when you were practicing in the API Playground). So, in a Python notebook, you might see something like this:

```
In [1]: 1 import requests
2 data = {
3     'token': 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
4     'content': 'project',
5     'format': 'csv',
6     'returnFormat': 'csv'
7 }
8 r = requests.post("https://redcap.chop.edu/api/", data)
9 r.text
```

```
Out[1]: 'project_id,project_title,creation_time,production_time,in_production,project_language,purpose,purpose_other,project_
notes,custom_record_label,secondary_unique_field,is_longitudinal,surveys_enabled,scheduling_enabled,record_autonumber
ing_enabled,randomization_enabled,ddp_enabled,project_irb_number,project_grant_number,project_pi_firstname,project_pi
_lastname,display_today_now_button,has_repeating_instruments_or_events\n24740,"Arcus Training Feedback","2018-12-07 1
4:19:34","2018-12-07 15:07:55",1,English,3,,,,,0,1,0,0,,,,,1,0\n'
```

How can we get that into a format we can work with? I suggest using `pandas` to make this string into a dataframe! We'll use `pandas` and `io` to read that long string into a dataframe:

```
import pandas as pd
from io import StringIO
df = pd.read_csv(StringIO(r.text))
df
```

And this is what we find:

```
1 import pandas as pd
2 from io import StringIO
3 df = pd.read_csv(StringIO(r.text))
4 df
```

	project_id	project_title	creation_time	production_time	in_production	project_language	purpose	purpose_other	project_notes	custom_record_label	...	sch
0	24740	Arcus Training Feedback	2018-12-07 14:19:34	2018-12-07 15:07:55	1	English	3	NaN	NaN	NaN	...	

Now, you're ready to work with the data! Obviously, in a more realistic situation, where you're importing research records, you'll have more than just one row like I do here.

R

The R Code provided by REDCap is almost perfect ... here's how to get a data frame in R from the data you're pulling in from REDCap. Take the code provided by REDCap (you can optionally remove the first line, `#!/usr/bin/env Rscript`, if you're working directly in R or RStudio) and this is what you get after issuing the `print(result)` command. Again, it's the string that makes up the .csv:

```
Console Terminal x R Markdown x
~/
> print(result)
[1] "project_id,project_title,creation_time,production_time,in_production,project_language,purpose,purpose_other,project_notes,custom_record_label,secondary_unique_field,is_longitudinal,surveys_enabled,scheduling_enabled,record_autonumbering_enabled,randomization_enabled,ddp_enabled,project_irb_number,project_grant_number,project_pi_firstname,project_pi_lastname,display_today_now_button,has_repeating_instrument"
```

```
nts_or_events\n24740,\"Arcus Training Feedback\", \"2018-12-07 14:19:34\", \"2018-12-07 15:07:55\", 1, Engl
ish, 3, , , , , 0, 1, 0, 1, 0, 0, , , , , 1, 0\n"
attr(,"Content-Type")
      charset
"text/csv"   "utf-8"
>
```

Just like in Python, we have to bring that string into the dataframe. To do that, we'll replace the line that reads `print(result)` and instead add these lines:

```
con <- textConnection(result)
df <- read.csv(con)
df
```

This gives you a data frame in R that you can analyze! See below, where I use RStudio's `view` command to make the data frame appear in an easy to read format in a source pane:



```
1 library(RCurl)
2 result <- postForm(
3   uri='https://redcap.chop.edu/api/',
4   token='[REDACTED]',
5   content='project',
6   format='csv',
7   returnFormat='csv'
8 )
9 con <- textConnection(result)
10 df <- read.csv(con)
11 View(df)
```

6:16 (Top Level) R Script

Untitled1*

df

↩

↪

🔍

Filter

🔍

	project_id	project_title	creation_time	production_time	in_production	project_language	purpose	purpose_other	p
1	24740	Arcus Training Feedback	2018-12-07 14:19:34	2018-12-07 15:07:55	1	English	3	NA	A

Now you know how to reach in directly to REDCap to get data in or about your project and bring it into R or Python!

Like this article? Click "Like" to let us know.

Like

Previous

← *Date Pairing in R*

Next

Descriptive Statistics: The Bullet →

Give us feedback or request a topic! [User survey](#) now available.

About DBHi

The [Department of Biomedical and Health Informatics \(DBHi\)](#) explores how technology can impact both research and patient care. DBHi's talented team of data scientists, programmers, and bioinformatics scientists combine both technological and scientific expertise to transform research and clinical data into innovative solutions that directly impact patient outcomes. Additionally, DBHi provides an academic home for all research informatics activities, including the development and deployment of intellectual, technical, and educational resources in biomedical computing.

About Arcus

Arcus is an initiative by the [Children's Hospital of Philadelphia \(CHOP\) Research Institute](#) which aims to connect clinical and research data. We allow biomedical researchers to conduct highly innovative, data-driven, computationally complex, and reproducible research. We join researchers in collaborative endeavors that promote science and scientists and we stand at the forefront of new research practices that promote science that is transparent, ambitious, and nimble.

Educational Focus

The ongoing formation of biomedical researchers in the fields of statistics, data manipulation, computation, and reproducible research methods is a focus of the NIH as well as the CHOP Research Institute. Arcus aims to educate researchers in the use of Arcus-specific tools for research at scale, as well as provide fundamental education for all researchers in the use of data science principles.

[Email Arcus Education](#)

© 2021 Children's Hospital of Philadelphia. Powered by [Jekyll](#) & [So Simple](#).