Paper CT06

Make GitHub Your Web-based Version-controlled Code Repository

Spencer Childress, Rho®, Chapel Hill, NC, United States
Shane Rosanbalm, Rho, Chapel Hill, NC, United States

ABSTRACT

Downloading code from GitHub® manually is straightforward: navigate to the repository website, download the ZIP file, and extract it to your working directory. However, because this process is manual it needs to be repeated whenever the repository changes, such as when the developer applies bug fixes or incorporates new features. SAS® and R provide users the tools to programmatically download and source repositories housed on GitHub. This paper demonstrates how to automate the download of code from GitHub using both SAS and R, saving you time and keeping your code up to date.

INTRODUCTION

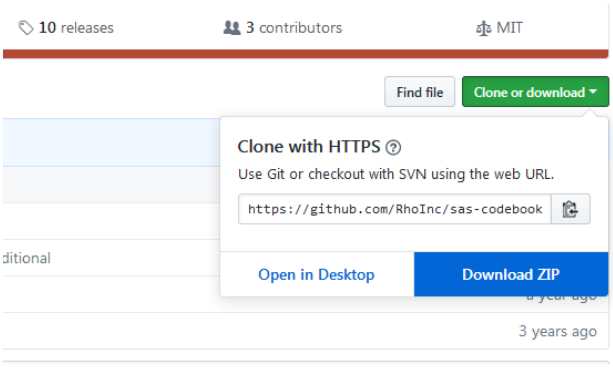
SAS and R both have a base set of functionality, but they differ in that R provides access to user-created packages, code bundles that extend R's functionality, hosted on services like the Comprehensive R Archive Network (CRAN) and GitHub. CRAN hosts established, vetted packages while GitHub tends to host more developmental packages. GitHub is not limited to R code, however, so the macro described in this paper extends to SAS access to remote code repositories so ingrained in R development.

SAS users typically store reusable SAS programs in directories known as AUTOCALL libraries. These code repositories might reside on the programmer's computer or on a network drive, and are generally developed in isolation, by a single programmer or within a single organization. R users, meanwhile, pull reusable code down from CRAN. Hosting services provide easy and universal access to codebases, and require only an internet connection.

GitHub is a programming language-agnostic hosting service that allows users to store and access code in the cloud. It leverages Git, an open-source version control system, to track and control changes to the code. Storing code remotely allows the programmer to access, modify, and run it from any workstation with an internet connection. Additionally Git tracks every change to the code and makes available every version, tracked in a commit history. If a newer version causes problems the user can easily revert to an earlier version.

MANUAL CODE DOWNLOAD

Code can be manually downloaded from GitHub with a few mouse clicks. Click on the *Clone or download* button on the repository home page and select the *Download ZIP* option.



Once the ZIP file has been saved locally, extract the contents and point to it from within your program.

In R the pointing looks like:

```
install.packages(lib="my/download/path/")
```

And in SAS like:

```
%include "my/download/path/fancymacro.sas";
```

Manual download from GitHub is straightforward to implement but time-consuming, and it is not robust to code updates. By programmatically downloading remotely-hosted code this process can be automated and future-proofed to new versions of the codebase.

AUTOMATED CODE DOWNLOAD

A more robust solution is to skip the manual download and use statements within SAS and R to access the code directly from the remote repository. This approach avoids any manual steps and also ensures that you are always using the latest version of the code.

In R, the *devtools* package contains a function named *install_github*. This function allows users to automate the installation of other packages directly from GitHub.

```
# Install and source the devtools package.
install.packages("devtools")
library(devtools)

# Call install_github to download a repository directly from GitHub.
install_github("someuser/endswithR")
library(endswithR)
```

SAS lacks built-in functionality to simultaneously download and install code from GitHub, which prompted the creation of the SAS macro %install_github (available at RhoInc/sas-install-github). This macro behaves much like the corresponding R package. After a one-time manual download and install of the %install_github macro itself, SAS users are henceforth able to use the macro to automagically download and install other SAS code directly from GitHub.

```
*--- point to the manually downloaded install_github macro ---;
%include "my/utility/macros/install_github.sas";

*--- use install_github to install SAS code directly from GitHub ---;
%install_github(
    repo = rhoinc/violinPlot,
    file = src/violinPlot.sas
);
```

HOW SAS ACCESSES GITHUB

SAS accesses GitHub via the URL access method of the FILENAME statement that associates a *fileref* with the URL of an external file:

```
FILENAME fileref URL 'external-file' <url-options>;
```

A %INCLUDE statement that references the *fileref* makes the external file available to the SAS session:

```
%INCLUDE fileref;
```

The code to access the *violinPlot* repository referenced in the previous section calls these two statements to read in the *violinPlot* macro:

```
filename fileURL url
    '//raw.githubusercontent.com/RhoInc/sas-violinPlot/master/src/violinPlot.sas';
    %include fileURL;
filename fileURL; * clear fileref ;
```

GitHub also exposes an API that simplifies access to the entire repository. The PROC HTTP statement can issue requests to the API and return the response to the SAS session. Consider the sas-codebook repository, which houses a collection of files that all need to be available to produce a codebook. The following statements request information from the GitHub API about the Macros folder:

```
filename inFolder temp;
proc http
    url = 'https://api.github.com/repos/RhoInc/sas-codebook/contents/Macros'
    method = 'GET'
    out = inFolder;
run;
```

The GitHub API sends the response in a format called JavaScript Object Notation, or JSON. The LIBNAME statement knows how to parse the response, making it available to the data step. By iterating over the response SAS generates statements that access each file in the Macros folder sequentially:

```
libname inFolder json fileref = inFolder;
data _null_;
    set inFolder.root;
    call execute('filename fileURL url "' || strip(downloadURL) || '";');
    call execute('%include fileURL;');
    call execute('filename fileURL;');
run;
```

SAS makes accessing individual files and collections of files relatively simple with the FILENAME URL access method and the LIBNAME JSON engine. R has similar capabilities but the internals of the *devtools* library are beyond the scope of this paper.

CONCLUSION

To access R or SAS code directly from GitHub without the hassle of a manual download, use the install_github function from the R package *devtools* or the SAS macro %install_github to help automate the process. With these functions, programmers can access and continue their work anywhere with an internet connection, effectively becoming workstation-agnostic. GitHub's API extends the advantage of remote code repositories to SAS users, a capability R users have enjoyed for years.

RECOMMENDED READING

Wikipedia. *GitHub: Scope*. Retrieved from https://en.wikipedia.org/wiki/GitHub#Scope

Chacon, S. and Straub, B. *Git – About Version Control*. Retrieved from https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

RDocumentation. *devtools: install_github*. Retrieved from https://www.rdocumentation.org/packages/devtools/versions/1.13.6/topics/install_github

Wikipedia. *Hypertext Transfer Protocol: Request methods*. Retrieved from https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Spencer Childress
Rho, Inc.
6330 Quadrangle Dr
Chapel Hill, 27517
spencer_childress@rhoworld.com
www.rhoworld.com

Shane Rosanbalm
Rho, Inc.
6330 Quadrangle Dr
Chapel Hill, 27517
shane_rosanbalm@rhoworld.com
www.rhoworld.com