RStudio/GitHub Setup

Charlie Labuzzetta

Spring Semester 2017

Chapter 1

Setup

This section includes details on creating a functioning RStudio workspace, connecting with GitHub, and installing Dr. Niemi's data structure template for the Strips2 project from a GitHub repository.

1.1 Integrate RStudio and GitHub

RStudio can be linked to a GitHub account using SSH keys. Connecting the workspace with GitHub allows for direct access to commit and push capabilities.

1. Generate SSH keys in RStudio
   - In RStudio, select Tools → Global Options...
   - Navigate to the Git/SVN Tab
   - Press the "Create RSA Key..." Button
   - Select "Create" and close the window that appears
   - Use the "View public key" link to display the SSH key
   - Copy this information for later use
2. Add new SSH key to GitHub account
   - Navigate to the settings on your github.com account
   - Select the tab titled "SSH and GPG keys"
   - Press the "New SSH Key" Button
   - Enter a title such as "RStudio Integration"
   - Paste the previously copied information into the textbox
3. Update git information on local computer
   - Return to RStudio and navigate to Tools → Shell...
   - Enter the following commands into the terminal:
     - git config –global user.email "clabuzze@iastate.edu"
     - git config –global user.name "clabuzze"

1.2 Install Packages in RStudio

The following packages should be installed in RStudio for use throughout the project. Packages can be installed with the following command:

- install.packages("<package name>")

List of packages to install:

- devtools
- readr
- dplyr
- tidyr
- roxygen2
- rtools
- knitr

1.3 Integrate the Data Template with RStudio

Dr. Niemi has created a template that is designed to store data in an R package. This is particularly applicable for the STRIPS2 Project, so that each Principal Investigator (PI) of the project can store their data in a standardized format. Now that RStudio has been integrated with GitHub, the template can be created in RStudio from the directions on GitHub: jarad/RDataPackageTemplate. To use the template on your personal computer and create a corresponding repository on GitHub, follow these steps:

- Navigate to Dr. Niemi's template which contains directions to create the repository: https://github.com/jarad/RDat
- Once the project's files have been created, open the .gitignore file
- Add *.Rproj under the R Studio files spot in the .gitignore file (This will tell GitHub to ignore the any files that are only relevant to the R Project)

Chapter 2

Create a Data Package

In this section, the STRIPSMeta package will be updated as an example of how data are stored in an R Package to be distributed on GitHub. The STRIPSMeta package will be recreated and updated in order to demonstrate how to create a package.

2.1 Setup the Package

Edit the structure of the package to include standardized folders and files.

- Create an ”R” directory
- Navigate to ”Tools...Project Options...Build Tools”
- Make sure that the ”Use devtools package functions if available” and ”Generate documentation with Roxygen” checkboxes are selected.
- Use the command ”devtools::create_description()” to create a DESCRIPTION file
- Update the following information:

Package: STRIPSMetaUpdate
Title: STRIPS Meta Update
Version: 0.1.2
Authors@R: c(person(”Jarad”, ”Niemi”, email = ”niemi@iastate.edu”, role = c(”aut”, ”cre”)), person(”Lisa”, ”Schulte-Moore”, email = ”lschulte@iastate.edu”, role = ”aut”))
Description: STRIPS meta data including watershed details, e.g. treatment, size, etc. and crop rotation.
Depends: R (>= 3.1.2)
License: GPL-3
Encoding: UTF-8
LazyData: true
VignetteBuilder: knitr
URL: https://github.com/ISU-STRIPS/STRIPSMeta
BugReports: https://github.com/ISU-STRIPS/STRIPSMeta/issues
RoxygenNote: 5.0.1

- Commit and push the information to GitHub
- Load the devtools library using ”library(devtools)” and then run the ”use_data_raw()” command

2.2 Add Data to the Package

Raw data is now added to the package in the form of .csv files and .R files are used to import the data.

- In the data-raw directory, add the .csv files containing the raw data, and commit the files to GitHub.
- For each of the .csv files, create an .R script of the same name.
- In each of these files, write a script that reads the corresponding csv file using the readr package and imports the data to the package using the devtools ”use_data” command as follows. Note the code may need to be edited/expanded upon in order to transform the raw data into a usable data table.

```
watersheds <- readr::read_csv(”watersheds.csv”)
devtools::use_data(watersheds, overwrite = TRUE)
```

2.3 Document the Package

- Save and run (or source) each of the .R files for the .csv's to add the variables to the environment.
- In the R directory create a file, named ”data.R” and create documentation code describing the csv files similar to the following:

```
#′ Seed info
#′
#′ Seed meta data including year, seed type, and tolerance
#′
#′ @format A data frame with 3 variables: \code{year}, \code{pioneer_seed_type},
#′ and \code{tolerance}
”seed_info”
```

- Re-Document the package, using the command found in the Build tab under ”More...Document”
- Try to ”Build and Reload” Package using the button in the Build Tab
- In the R directory create a file, named ”¡packageName¿.R” and create documentation code describing the project similar to the following:

```
#′ STRIPSMetaUpdate: A package update for STRIPS meta data.
#′
#′ The STRIPSMeta package contains STRIPS meta data including watershed
#′ information, e.g. treatment, size, etc., crop rotation, and seed type info.
#′
#′ @section STRIPSMeta data sets:
#′ \code{\link{watersheds}}
#′ \code{\link{crop_rotation}}
#′ \code{\link{seed_info}}
#′
#′ @seealso \code{\link{watersheds}}, \code{\link{crop_rotation}},
   \code{\link{seed_info}}
#′
#′ @docType package
#′ @name STRIPSMetaUpdate
NULL
```

- Re-document the package similar to above
- Build and Reload the package
- Now try loading the package: library(STRIPSMetaUpdate)
- Commit and push changes to GitHub