Whitepaper

GitHub and RStudio

February 10, 2018

Tags: github

This tutorial teaches you to create R Markdown documents with RStudio and publish them via GitHub, using GitHub Pages.

RStudio is a popular integrated development environment for R. It integrates the tools you use with R into a single environment. GitHub Pages allows you to host websites directly from your GitHub repository.

In this tutorial, you will learn how to:

- Create a new repository.
- Work with your repository on your local machine using Git and RStudio.
- Use the GitHub.com website or GitHub Enterprise to complete the GitHub workflow.
- Publish and share your R Markdown documents using GitHub Pages.

Prerequisites

- For this tutorial you will need an account on GitHub.com or GitHub Enterprise.
- If you are new to Git, GitHub and GitHub Pages it is recommended to complete the GitHub Pages from the command-line course first.
- Some programming knowledge with R will be helpful but is not required.

Install Git and RStudio

For this tutorial you will use Git and RStudio to work with your GitHub repository.

1. Download and install Git.
2. Download and install RStudio (1.1.383 or higher).
3. Open RStudio.

Create the remote repository on GitHub

Now that you have what you need installed locally, let's create the repository that will hold your new website.

1. On GitHub.com, create a new repository. **Note**: if your organization uses GitHub Enterprise you can also create the repository there.
2. Name your repository `cars`. You can use a different name.
3. Enter a description for your repository.
4. Choose Public visibility.
5. Select Initialize this repository with a README.
6. Click Add `.ignore` and select `R`.
7. Click Create repository.

Clone the repository with RStudio

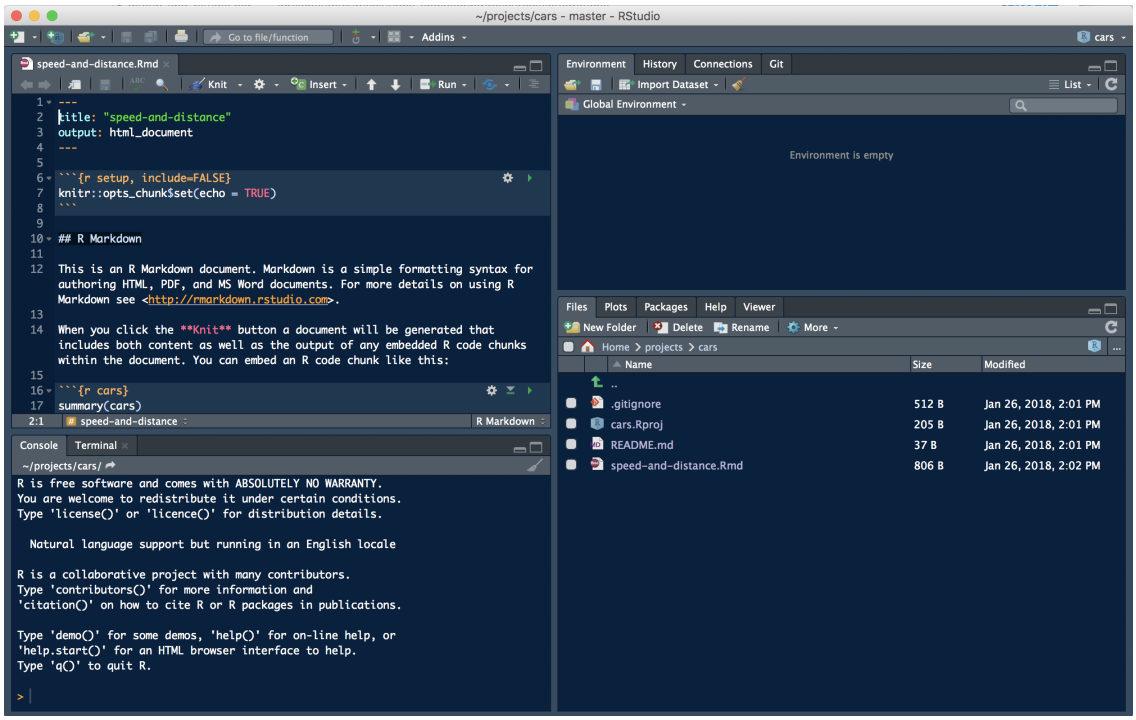
After you've created a repository on GitHub (the remote repository), the next step is to clone it to your local environment.

1. On GitHub, navigate to the Code tab of the repository.
2. On the right side of the screen, click `Clone or download`.
3. Click the `Copy to clipboard` icon to the right of the repository URL.
4. Open RStudio on your local environment.
5. Click `File`, `New Project`, `Version Control`, `Git`.
6. Paste the repository URL and enter TAB to move to the `Project directory name` field.
7. Click `Create Project`.

Create an R Markdown document in RStudio

Now that you have a local copy of the repository, let's add an R Markdown document to your project.

1. In RStudio click `File`, `New File`, `R Markdown`.
2. Choose HTML output as default output.
3. Click `File`, `Save` to save the document.
4. Enter `speed-and-distance` and click `Save`.



Commit and push the changes to GitHub

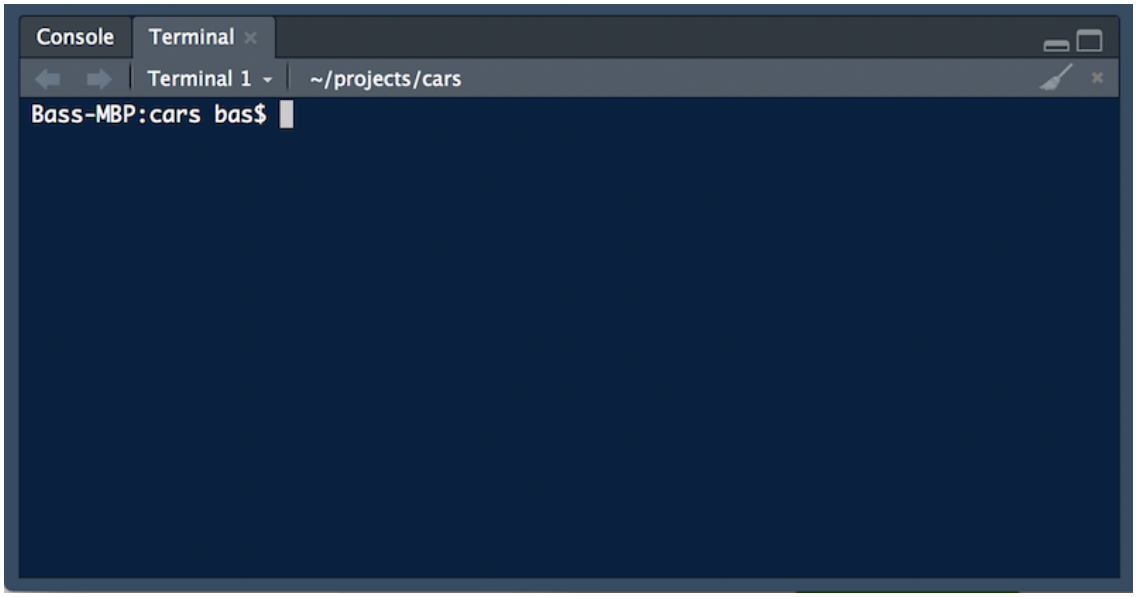
After you have created the R Markdown document and finished making your changes, it is time to commit them.

1. In RStudio click the `Git` tab in the upper right pane.
2. Click `Commit`.
3. In the `Review changes` view, check the staged box for all files.
4. Add a commit message, for example `Add initial speed and distance report`.
5. Click `Commit`.
6. Click the `Pull` button to fetch any remote changes.
7. Click the `Push` button to push your changes to the remote repository.
8. On GitHub, navigate to the Code tab of the repository to see the changes.

Create local branches with Git

Let's make a couple of more changes in your project using the steps of GitHub Flow. As RStudio currently does not support local branches very well, we will use Git from the command-line in RStudio.

1. In RStudio click the `Terminal` tab in the lower left pane. The Terminal tab is next to the Console tab.



Note: if you cannot find the Terminal tab, check if you use RStudio version 1.1.383 or higher. Click `Help`, `About RStudio` to check the current version.

2. Create a new branch. Replace `<BRANCH-NAME>` with a descriptive name, for example `speed-and-distance-report`:

```
git branch <BRANCH-NAME>
```

3. Check your repository's status:

```
git status
```

   Notice that although you created a new branch, you are still checked out to master, as indicated by the in-line response from Git.

4. Check out to your new branch:

```
git checkout <BRANCH-NAME>
```

5. Verify that you are now checked out to your new branch:

```
git status
```

Make local changes with Git

Let's generate the HTML output in a separate directory called `docs`. To do this we add a function in the heading of the R Markdown document to 'knit' the output in the desired output directory.

1. In RStudio open `speed-and-distance.Rmd` and add the following lines in the heading under the `title` field:

```
knit: (function(input_file, encoding) {
  out_dir <- 'docs';
  rmarkdown::render(input_file,
```

```
encoding=encoding,
output_file=file.path(dirname(input_file), out_dir, 'index.html'))})
```

2. Click `File`, `Save` to save the changes.
3. Click `Knit` in the document pane for `speed-and-distance.Rmd`.

Commit local changes with Git

After you have created the HTML output, it is time to commit the changes.

1. Determine your file's status. Remember that `git status` allows us to see the status of the files on our branch at any given time. Your file is listed under the heading `Untracked files`:

```
git status
```

2. Add your file to the staging area so it's prepared to become part of the next commit:

```
git add .
```

3. See your file's current status. Your file is now listed under the heading `Changes to be committed`. This tells us that the file is in the staging area. It also indicates this is a new file:

```
git status
```

4. Commit your file. Replace `<COMMIT-MESSAGE>` with a log message describing the changes, for example `Knit output to a docs folder`. A commit tells Git to collect all of the files in the staging area and store them to version control as a single unit of work:

```
git commit -m "<COMMIT-MESSAGE>"
```

5. See the history of commits:

```
git log --oneline
```

6. See the changes between the master branch and the current branch (HEAD):

```
git diff --stat --summary master..<BRANCH-NAME>
```

Open a pull request on GitHub

Now that you have made some local commits, it is time to send your changes to the remote copy of your repository on GitHub and create a Pull Request.

1. Push the changes to the remote repository:

```
git push -u origin <BRANCH-NAME>
```

2. Create a Pull Request on GitHub.
3. Fill out the body of the Pull Request with information about the changes you're introducing.

Merge your pull request on GitHub

Since this is your repository, you probably don't have anyone to collaborate with (yet). Go ahead and merge your Pull Request now.

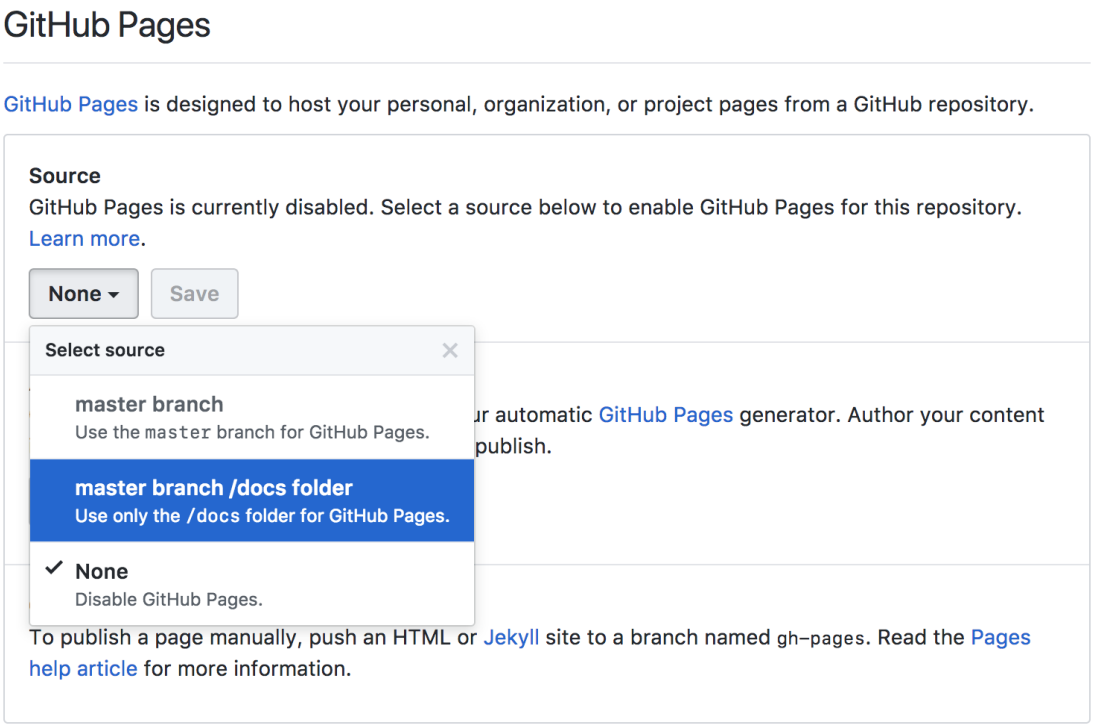
1. On GitHub, navigate to the Pull Request that you just opened.
2. Scroll down and click the big green `Merge Pull Request` button.

3. Click Confirm Merge.
4. Delete the branch `<BRANCH-NAME>`.

Create the pages site

This tutorial is going to use GitHub Pages to publish the HTML output. To initialize GitHub Pages we need to perform a few more steps:

1. In your repository, click the Settings tab.
2. Scroll down to the GitHub Pages section.
3. Under `Source`, select `master branch /docs folder`.



4. Click `Save` to save the changes.
5. Click the generated GitHub Pages URL to view the rendered R Markdown document.

Note: GitHub Pages sites are always public when hosted on GitHub.com. If you want to share a site with a select number of people you can use Jekyll Auth. On GitHub Enterprise users need to authenticate to access GitHub Pages sites when private mode is enabled.

Update local repository

After you merge your Pull Request, you will need to update your local copy of the repository.

1. In the RStudio Terminal pane, type:

```
git checkout master
```

2. Type:

```
git pull
```

3. Delete the local branch :

```
git branch -D <BRANCH-NAME>
```

4. See the history of your commits in a graph:

```
git log --oneline --graph --all
```

You can also view the history of your commits in RStudio. Click `Commit` in the Git pane to open the Review Changes panel and then click `History` .

Add welcome page, theme and navigation

Let's add a welcome page, theme and navigation to allow you to publish a collection of R Markdown documents.

1. First create a new branch. Again replace `<BRANCH-NAME>` with a descriptive name, for example `add-theme-and-navbar` :

```
git checkout -b <BRANCH-NAME>
```

2. Add a file `_site.yml` with the following contents:

```
name: "cars"
output_dir: "docs"
navbar:
  title: "Cars"
  left:
  - text: "Home"
    href: index.html
  - text: "Speed and Distance"
    href: speed-and-distance.html
output:
  html_document:
 theme: cosmo
```

3. Add a new R Markdown document `index.Rmd`. This is your welcome page.
4. Open `speed-and-distance.Rmd` and remove the `knit` and `output` fields from the heading. The heading will now look like this:

```
---
title: "Speed and Distance"
---
```

5. Save the changes.
6. Open the `Console` in the left bottom pane and render the site using the following command:

```
rmarkdown::render_site()
```

7. Commit the local changes with Git:

```
 git status
git add .
git commit -m "Add home page, navigation and theme"
```

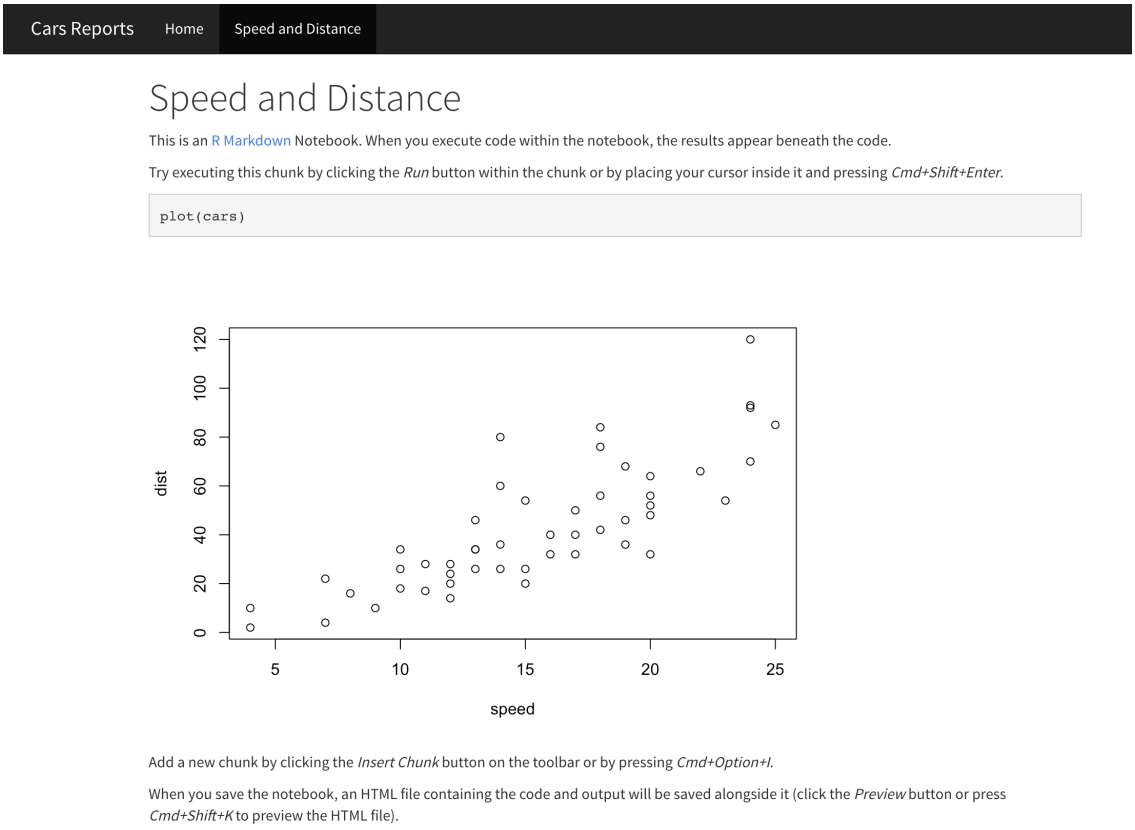
Open a pull request

Now that you have added the home page and navigation, it is time to send your changes to the remote copy of your repository on GitHub and open a new pull request.

1. Push the changes to the remote repository:

```
git push -u origin <BRANCH-NAME>
```

2. Open a Pull Request on GitHub.
3. Go ahead and merge your Pull Request.
4. View your updated site. The site should look similar to this image.



5. Add the GitHub Pages URL to the repository description to make it easier to discover.

If you can't remember the GitHub Pages URL you can always visit the URL in the GitHub Pages section on the settings page.

Congratulations!

You have completed the GitHub workflow with RStudio and added a GitHub Pages site to your project to publish your rendered R Markdown documents.

Links

- The R Markdown Websites article on the R Markdown website is a good resource to build GitHub Pages websites that host collections of R Markdown documents.
- Use GitHub Desktop if you prefer to use a graphical user interface rather than typing Git commands in the terminal when working with branches in RStudio.

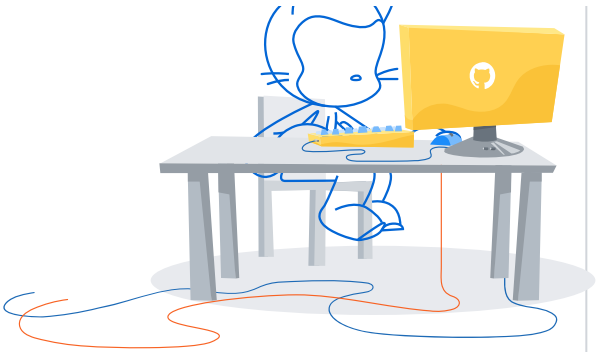
See all whitepapers →

Acme, Inc

What can we help you with?

Tell us how we can help. For support questions, head to github.com/contact

Contact GitHub



Related

Whitepaper

Perspectives on software development - lessons on accelerating innovation

June 2, 2020

GitHub is more than a DevOps platform—it's also a community. In May, the community came together virtually at GitHub Satellite 2020 to share insights, best p...

View →

Video

GitHub Advanced Security: Secure your software development lifecycle

May 27, 2020

While security measures in the software development process are becoming more important every year, they are still often put on the back burner within the ac...

View →

Video

GitHub Advanced Security: Secure your software development lifecycle

May 27, 2020

While security measures in the software development process are becoming more important every year, they are still often put on the back burner within the ac...

View →