

**Continuous Integration services – Automate  
your unit tests**

# More complicated test scenarios require more advanced approaches: Stubs and Mocks

Some functions cannot be executed for testing purposes, e.g.

- Functions that access different systems, e.g., online authentication
- Persistent manipulations of databases
- Hardware controlling functions, e.g., a robot arm
- Execution of financial transactions, etc.
- Functions' dependency of non-existent code

The solution? Stubs and Mocks, see

<https://stackoverflow.com/questions/3459287/whats-the-difference-between-a-mock-stub>

# Stubs and Mocks

## Stubs

- The underlying operation is replaced by a stub for testing
- Stubs can perform primitive operations but usually return only a value
- Example: Test depends on a method that takes 5 minutes to complete, you can replace real implementation with a stub that returns hard-coded values, taking up much less time

## Mocks

- In OOP, replacements for full objects are called mocks
- Mocks additionally check if methods were called as expected
- Example: You are testing a user creation class, so you want to check after calling `createUser()`, that `sendConfEmail()` is called

# The more tests, the better... Check your code coverage (1/2)

- Code coverage shows which lines of code are tested
- Helps identify non-tested code regions
- Usually measures coverage as ratio, e.g., 60% of all lines, functions, etc.

Warning: a high coverage does not guarantee thorough testing

- As a recommendation, focus especially on the boundaries (also called edge cases) of parameter ranges (0, NA, Inf, etc.) to identify unhandled problems

R package covr

- Supports coverage only when testing full packages

Workaround is to create a dummy package

# The more tests, the better... Check your code coverage (2/2)

With covr loaded, you can run `report()` to get a report of your code coverage.

Files

Source

Coverage	File	Lines	Relevant	Covered	Missed	Hits / Line
100.00	R/hello.R	18	1	1	0	2

For more information, visit <https://covr.r-lib.org/>

# Execute all tests & calculate the code coverage automatically on every Push to GitHub

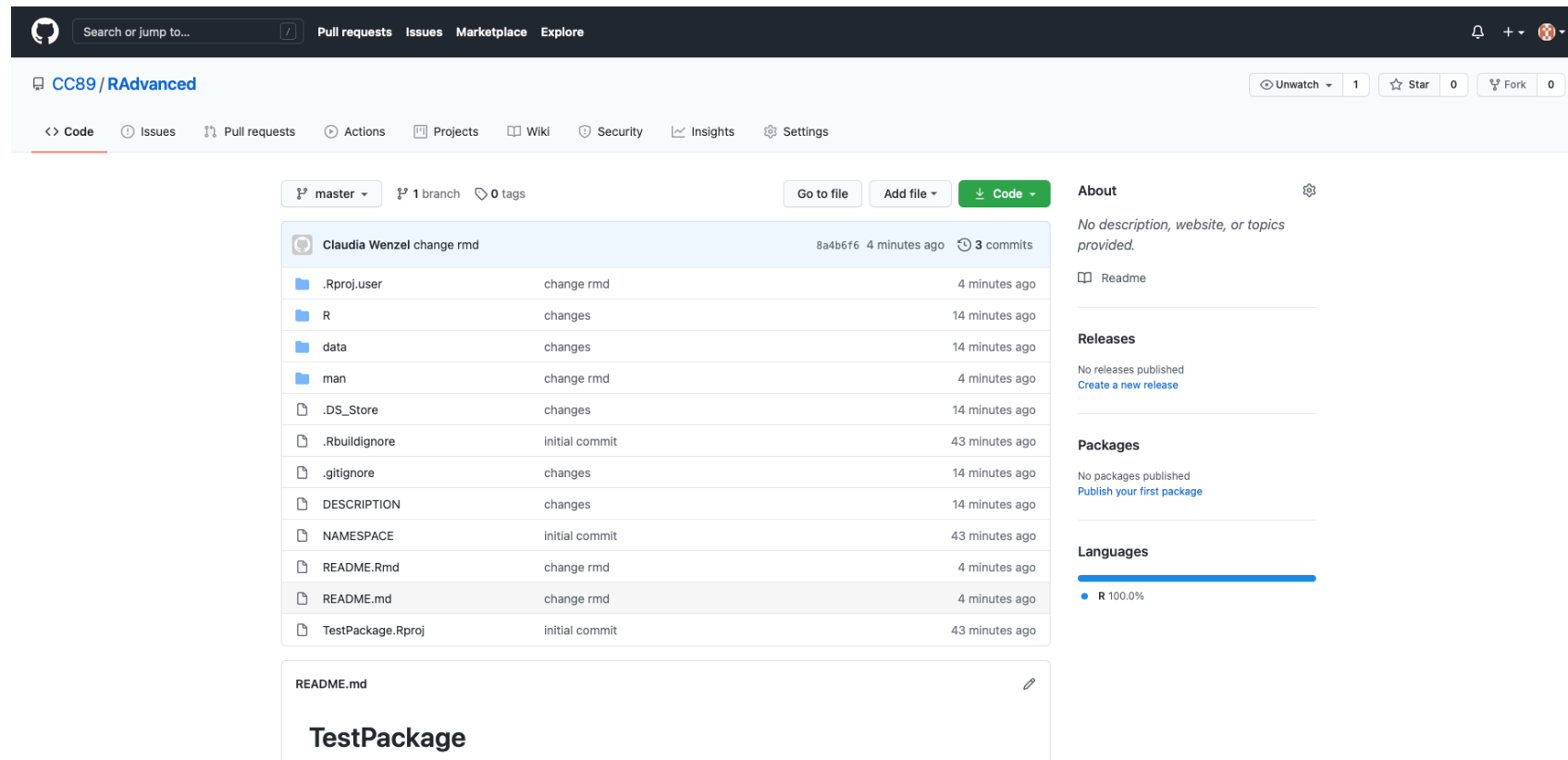
- In software engineering, continuous integration (CI) is the practice of regularly merging all changes to source code to a central repository, building and testing the source code after every change. This allows any bugs that are introduced to be identified quickly.
- Travis CI (travis-ci.org) is a continuous integration service for use with GitHub code repositories. For open-source projects it is free to use. Alternative CI software exists for use with GitHub, and with alternative version control systems and hosting services (for example, Jenkins).

# Steps to enable automatic test execution & code coverage calculation for a R package

1. Upload your package to GitHub.
2. Create an account at [travis-ci.com](https://travis-ci.com).
3. Activate the GitHub repository containing your package.
4. Add a `.travis.yml` file to the root directory of your package – the `use_travis` function in the `usethis` package will do this. The `use_covr` function will help you to set up the automatic calculation of the code coverage.
5. Commit this new file to your Git repository.

Now, whenever any commits are pushed to your online GitHub repository, Travis will automatically build and check your package. Travis will notify you if the check results in any errors, including those generated by failed unit tests.

# Step 1: Upload package to Github



Search or jump to... Pull requests Issues Marketplace Explore

CC89 / RAdvanced Unwatch 1 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

Claudia Wenzel change rmd 8a4b6f6 4 minutes ago 3 commits

.Rproj.user	change rmd	4 minutes ago
R	changes	14 minutes ago
data	changes	14 minutes ago
man	change rmd	4 minutes ago
.DS_Store	changes	14 minutes ago
.Rbuildignore	initial commit	43 minutes ago
.gitignore	changes	14 minutes ago
DESCRIPTION	changes	14 minutes ago
NAMESPACE	initial commit	43 minutes ago
README.Rmd	change rmd	4 minutes ago
README.md	change rmd	4 minutes ago
TestPackage.Rproj	initial commit	43 minutes ago

README.md

## TestPackage

About No description, website, or topics provided.

Readme

Releases No releases published [Create a new release](#)

Packages No packages published [Publish your first package](#)

Languages R 100.0%



## Step 2: Create account for Travis CI

Visit: <https://travis-ci.com/> and click "Sign in with GitHub".

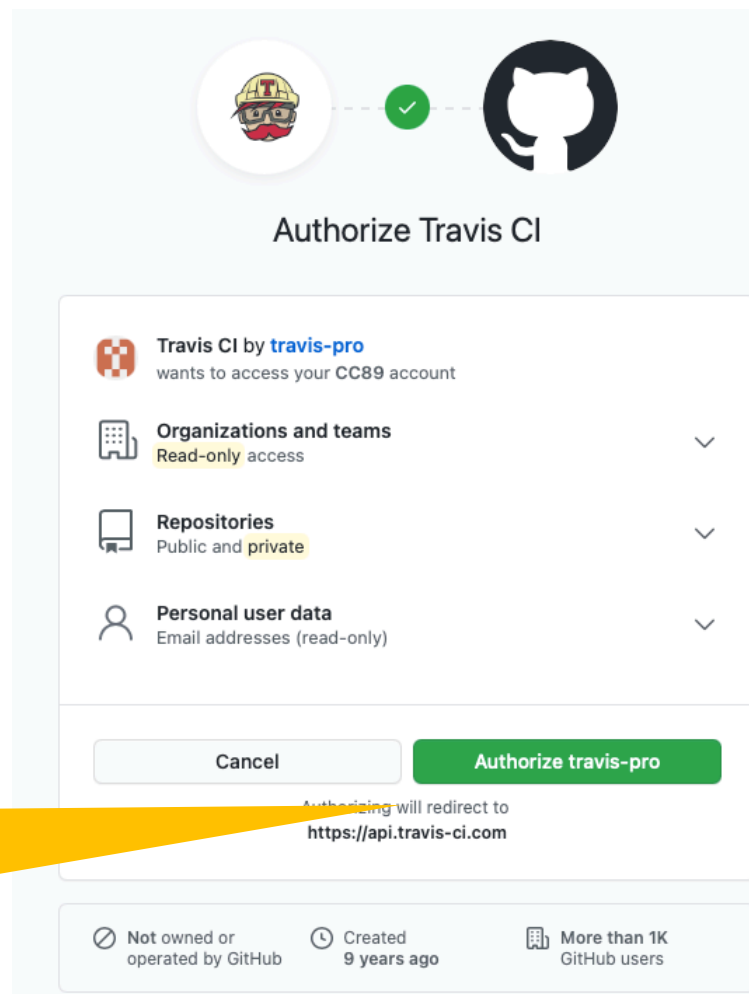
The screenshot shows the Travis CI website. The browser address bar displays [travis-ci.com](https://travis-ci.com/). A green circle with the number '1' highlights the 'Sign in' button in the top right corner of the navigation bar. Below the navigation bar, the main content area features the text 'The simplest way to test and deploy your projects.' and 'Easily sync your projects with Travis CI and you'll be testing your code in minutes.' A green 'Sign up' button is visible. Below it, the text 'Already have an account? [Sign in](#)' is shown. On the right side of the page, there is a 'Sign in' modal window. A green circle with the number '2' highlights the 'SIGN IN WITH GITHUB' button within this modal. Other buttons in the modal include 'SIGN IN WITH BITBUCKET', 'SIGN IN WITH GITLAB', and 'SIGN IN WITH ASSEMBLA', each with a 'Beta' label. At the bottom of the modal, it says 'Don't have an account? [Sign up](#)'.

<https://github.com/dwyl/learn-travis>

## Step 2: Create account for Travis CI

You will be redirected to GitHub  
where you need to click  
"Authorize travis-pro"

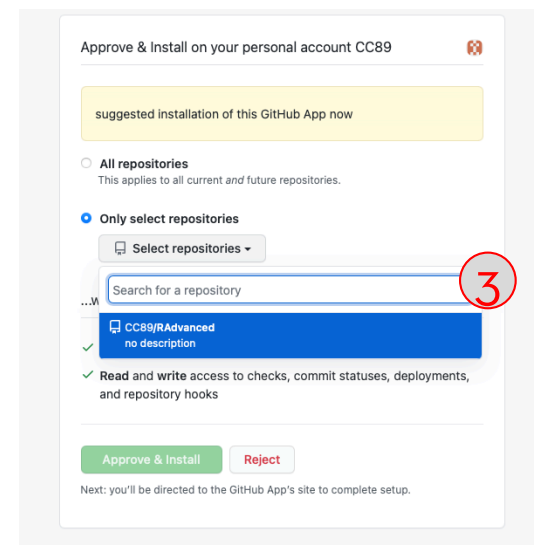
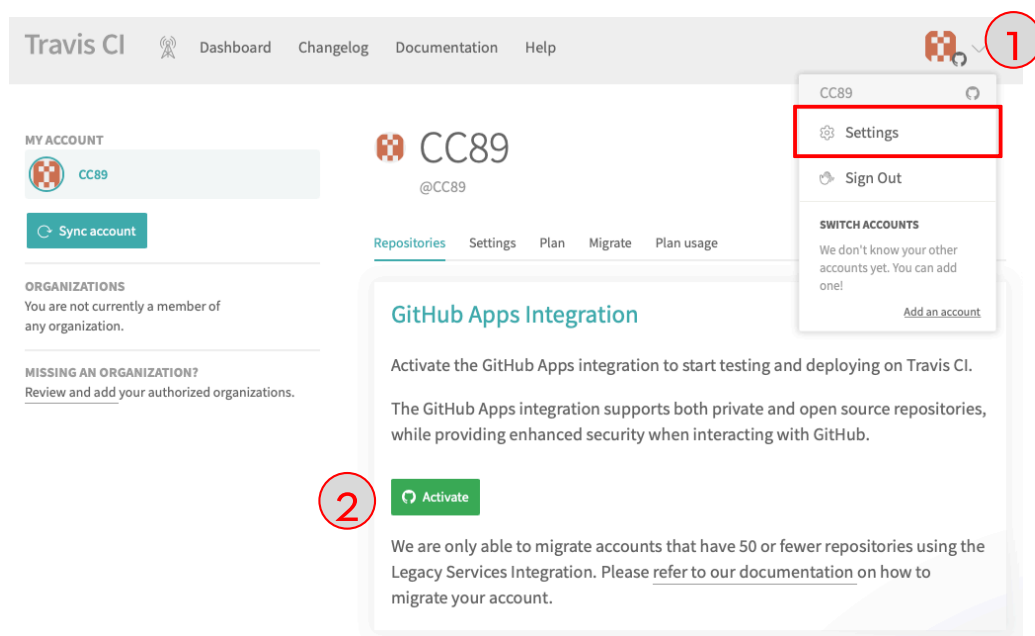
Note: If you ever want to stop Travis from accessing your GitHub account, simply visit:  
<https://github.com/settings/applications>  
and click on Revoke.



## Step 3: Activate Travis CI for the GitHub repository with the R package

Once you have allowed access you will be taken back to Travis where you will need to enable a specific Git Repository. You can also do this in your Travis Profile:

<https://travis-ci.com/profile>



<https://github.com/dwyl/learn-travis>

## Step 4: Add a travis.yml file to the root directory of your package

The screenshot shows the RStudio interface for a package named 'TestPackage'. The console at the bottom shows the command `file.create(".travis.yml")` being executed, with the output `[1] TRUE`. The editor window shows the content of `.travis.yml` as `language: r`. The file explorer on the right shows the root directory of the package, with `.travis.yml` listed among other files.

1 `> file.create(".travis.yml")`

2 `.travis.yml`

3 `language: r`

## Step 5: Commit this new file to your Git repository.

The screenshot shows the RStudio interface for a project named 'TestPackage'. The editor on the left displays the content of '.travis.yml', which is:

```
1 language: r
2
```

The top right pane shows the 'Commit' button (circled with a red 4) and the 'Push' button (circled with a red 5). The 'Staged' pane shows the following files:

Staged	Status	Path
<input type="checkbox"/>	M	README.Rmd
<input checked="" type="checkbox"/>	A	travis.yml
<input type="checkbox"/>	M	.Rproj.user/66C59C03/sources/prop/999A84D2
<input type="checkbox"/>	M	.Rproj.user/66C59C03/sources/prop/INDEX
<input type="checkbox"/>	D	.Rproj.user/66C59C03/sources/s-15DBE808/4DB7FCC3
<input type="checkbox"/>	M	.Rproj.user/shared/notebooks/paths
<input type="checkbox"/>	M	.Rproj.user/shared/notebooks/C5BFF084-README/1/s/chunks.json

The bottom right pane shows the file explorer for the 'TestPackage' directory. The files and their sizes and modification dates are:

Name	Size	Modified
..		
.gitignore	40 B	Feb 5, 2021, 9:13 AM
.Rbuildignore	42 B	Feb 5, 2021, 8:35 AM
data		
DESCRIPTION	425 B	Feb 5, 2021, 9:11 AM
man		
NAMESPACE	60 B	Feb 5, 2021, 9:14 AM
R		
README.md	2.5 KB	Feb 5, 2021, 9:22 AM
README.Rmd	1.3 KB	Feb 5, 2021, 9:22 AM
TestPackage.Rproj	396 B	Feb 5, 2021, 9:14 AM
.travis.yml	12 B	Feb 5, 2021, 10:14 AM

The console at the bottom shows the command `file.create(".travis.yml")` being executed, resulting in `[1] TRUE`.

# How to see if Travis CI did actually run the tests?

The screenshot displays the Travis CI web interface. At the top, the navigation bar includes 'Travis CI', 'Dashboard', 'Changelog', 'Documentation', and 'Help'. A search bar for repositories is on the left. The main content area shows the repository 'CC89 / RAdvanced' with a 'build passing' status. Below this, tabs for 'Current', 'Branches', 'Build History', and 'Pull Requests' are visible. The 'Current' tab shows a build for 'master data documentation' with a green checkmark and '#3 passed'. It includes details like 'Commit 53ed863', 'Compare b478a61..53ed863', and 'Branch master'. The build duration is '1 min 49 sec' and it was finished '2 days ago'. A 'Restart build' button is present. Below the build summary, there are links for 'Job log' and 'View config'. The 'Job log' section shows the output of the build, including a message: 'R for Travis-CI is not officially supported, but is community maintained.' and sections for 'Worker information' and 'Build system information'.

Travis CI

Dashboard Changelog Documentation Help

Search all repositories

My Repositories Running (0/0) +

✓ CC89/RAdvanced # 3

Duration: 1 min 49 sec  
Finished: 2 days ago

Current Branches Build History Pull Requests

More options

✓ master data documentation - #3 passed Restart build

Commit 53ed863  
Compare b478a61..53ed863  
Branch master

CC89

</> R  
AMD64

Job log View config

Remove log Raw log

R for Travis-CI is not officially supported, but is community maintained.

1 Worker information worker\_info 0.07s

6

7 Build system information system\_info 0.01s

142

143 2.54s

The package passed all test. In case of mistakes, Travis will produce an error or warning.

**Now it's your turn!**