Travis CI is a common tool to build R packages. It is in my opinion the best platform to use R in continuous integration. Some of the most downloaded R packages built at this platform. These are for example *testthat*, *magick* or *covr*. I also built my package [*RTest*](https://medium.com/@zappingseb/rtest-pretty-testing-of-r-packages-50f50b135650) at this platform. During the setup I ran into some trouble. The knowledge I gained I’m sharing with you in this guide.

**Table of contents**

* [Basics from “Building an R project”](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#a056)
* [Modifying R CMD build](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#031a)
* [Multiple operating systems](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#9571)
* [Run scripts with user interfaces](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#45c4)
* [Code coverage](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#5da6)
* [Build and deploy a pkgdown page to github pages](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#22c9)
* [ImageMagick and Travis CI](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#7ef5)
* [Further reading](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#b418)

**Basics from “Building an R project”**

The article “[Building an R Project](https://docs.travis-ci.com/user/languages/r/)” from Travis CI tells you about the basics. It allows setting up a build for an R-package or R project. The main take away comes with this .travis.yml file.

# Use R language  
language: r

#Define multiple R-versions, one from bioconductor  
r:  
 - oldrel  
 - release  
 - devel  
 - bioc-devel

# Set one of you dependencies from github  
r\_github\_packages: r-lib/testthat

# Set one of your dependencies from CRAN  
r\_packages: RTest

# set a Linux system dependency  
apt\_packages:  
 - libxml2-dev

The tutorial explains to you that you should setup your type language as **R.** You can use different **R-versions.** Those R-Versions are:

[oldrel, release, devel, bioc-devel, bioc-release]

Additionally you can load any package from github by r\_github\_packages . Or you can get any package from [CRAN](https://cran.r-project.org/) by r\_packages . A list of multiple packages can be created using the standard yml format:

r\_packages:  
 - RTest  
 - testthat

In case you have a Linux dependency, it needs to be mentioned. The RTest package uses XML test cases. The XML Linux library needed is libxml2 . It can be added by:

apt\_packages:  
 - libxml2-dev

You are done with the basics. In case you have this .travis.yml file inside your repository, it will use R CMD build and R CMD check to check your project.

**Modifying R CMD commands**

To build my project I wanted to build it like on CRAN. Therefore I needed to change the script of the package check. Therefore I added:

script:  
 - R CMD build . --compact-vignettes=gs+qpdf  
 - R CMD check \*tar.gz --as-cran

Inside this script you can changeR CMD build orR CMD check arguments. For a list of arguments to R CMD see this [tutorial from RStudio](https://support.rstudio.com/hc/en-us/articles/200486518-Customizing-Package-Build-Options).

To run vignette compression get gs+qpdf by:

addons:  
 apt:  
 update: true  
 packages:  
 - libgs-dev  
 - qpdf  
 - ghostscript

**Multiple operating systems**

Travis CI offers two different operating systems right now (Jan 2019). Those are macOS and Linux. The standard way of testing is Linux. For my project [RTest](https://medium.com/@zappingseb/rtest-pretty-testing-of-r-packages-50f50b135650) I needed to test in macOS, too. To test in two operating systems use the matrix parameter of Travis CI.

The matrix parameter allows adjusting certain parameters for certain builds. To have the exact same build in Linux and macOS I used the following structure:

matrix:  
 include:  
 - r: release  
 script:  
 - R CMD build . --compact-vignettes=gs+qpdf  
 - R CMD check \*tar.gz --as-cran   
 - r: release  
 os: osx  
 osx\_image: xcode7.3  
 before\_install:  
 - sudo tlmgr install framed titling  
 script:  
 - R CMD build . --compact-vignettes=gs+qpdf  
 - R CMD check \*tar.gz --as-cran

The matrix function splits the build into different operating systems. For macOS I used the image xcode7.3 as it is proposed by [rOpenSCI](https://ropensci.org/blog/2016/07/12/travis-osx/). An extra point for this version is that it is close to the current CRAN macOS version. As you can see you should install the Latex packages framed and titling to create vignettes.

**Run scripts with User interfaces**

My package [RTest](https://medium.com/@zappingseb/rtest-pretty-testing-of-r-packages-50f50b135650) uses [Tcl/Tk](https://www.rdocumentation.org/packages/tcltk/versions/3.5.2?tap_a=5644-dce66f&tap_s=10907-287229) user interfaces. To test such user interfaces you need enable user interfaces in Linux and macOS separately. Travis CI provides the xvfb package for Linux. For macOS you need to reinstall xquartz and tcl-tk with homebrew .

**User interfaces for Linux**

To enable user interfaces in Linux install xvfb .

addons:  
 apt:  
 update: true  
 packages:  
 - x11proto-xf86vidmode-dev  
 - xvfb  
 - libxxf86vm-dev

You can run all R scripts with a user interface using the xvfb-run command in front of the R command.

script:  
 - R CMD build . --compact-vignettes=gs+qpdf  
 - xvfb-run R CMD check \*tar.gz --as-cran

**User interfaces for macOS**

For macOS the installation of a user interface is more difficult. You need to add xquart and tcl-tk to the image provided in xcode7.3 .

before\_install:  
 - brew update  
 - brew cask reinstall xquartz  
 - brew install tcl-tk --with-tk  
 - brew link --overwrite --force tcl-tk; brew unlink tcl-tk

To use xquartz there is no xvfb-run command under macOS. In a [github issue](https://github.com/travis-ci/travis-ci/issues/7313) I found a solution that still makes user interfaces work with xquartz .

before\_script:  
 - "export DISPLAY=:99.0"  
 - if [ "${TRAVIS\_OS\_NAME}" = "osx" ]; then ( sudo Xvfb :99 -ac -screen 0 1024x768x8; echo ok ) & fi

You create a display before running any R script that can be used by the R session. It is important to export the DISPLAY variable. This variable is read by the tcktk R package.

In macOS you do not need to change the script

script:  
 - R CMD build . --compact-vignettes=gs+qpdf  
 - R CMD check \*tar.gz --as-cran

**Addon**

For more information on user interfaces you can read these two github issues:

* [RSelenium with travis and shiny](https://www.google.com/search?q=rselenium+shiny+trvais&rlz=1C1GCEA_enDE800DE801&oq=rselenium+shiny+trvais&aqs=chrome..69i57.2913j0j4&sourceid=chrome&ie=UTF-8)
* [RSelenium with sauce](https://github.com/ropensci/RSelenium/issues/145)

**Code coverage**

For code coverage I would suggest to use one specific version of your builds. I decided for Linux + r-release to test the code coverage. First of all I added the *covr* package to my build script:

r\_github\_packages:  
 - r-lib/covr

Secondly I wanted to test my package using *covr*. This can be done in Travis using the after\_success step. To use *covr* inside this step you need to define how your package tarball will be named. You can write this directly into your script. A better way to do it is to write it into the env part of you .travis.yml file. The name of your tarball will always be *PackageName + “\_” + PackageVersion + “.tar.gz”.* Inside your DESCRIPTION file you defined *PackageName* and *PackageVersion.* I used CODECOV to store the results of my coverage tests.

env:  
 - PKG\_TARBALL=RTest\_1.2.3.1000.tar.gz

after\_success:  
 - tar -C .. -xf $PKG\_TARBALL  
 - xvfb-run Rscript -e 'covr::codecov(type=c("tests", "vignettes", "examples"))'

The setup I’m using for my package includes code coverage for all my examples, vignettes and tests. To deploy the results of the code coverage you must define the global variable CODECOV\_TOKEN . The token can be found under https://codecov.io/gh///settings . You can insert it secretly into your Travis CI build. Add tokens inside https://travis-ci.org///settings . The section environment variables stores variables secretly for you.

To use COVERALLS instead of CODECOV use the covr::coveralls function and define a COVERALLS\_TOKEN inside your environment.

**Build and deploy a pkgdown page to github pages**

Building a *pkgdown* page can be really useful to document your code. On my github repository I also host the *pkgdown* page of my package RTest. You can find the page here: <https://zappingseb.github.io/RTest/index.html>

To allow deployment to github pages I activated this feature at: https://github.com///settings. You have to use the *gh-pages* branch. If you do not have such a [branch you need to create it](https://help.github.com/articles/creating-and-deleting-branches-within-your-repository/).

Inside the .travis.yml you start by installing *pkgdown.*

r\_github\_packages:  
 - r-lib/pkgdown

You will have to build the page from your package tarball. The name of the package tarball has to be defined. Please see the section [code coverage](https://towardsdatascience.com/travis-ci-for-r-advanced-guide-719cb2d9e0e5?source=rss-dbc9f652035a------2#5da6) for how this is done. After unpacking the tarball you should delete any leftovers from checking the package by rm -rf .Rcheck .

after\_success:  
 - tar -C .. -xf $PKG\_TARBALL  
 - rm -rf RTest.Rcheck  
 - Rscript -e 'pkgdown::build\_site()'

The Rscript will produce the website inside adocs folder. This folder must be deployed on github pages.

First go to to <https://github.com/settings/tokens> when you’re logged into github. There you have to create a token with public\_repo or repo scope. Now store this token inside your Travis CI build. Therefore go to https://travis-ci.org///settings and store it as a global variable named GITHUB\_TOKEN . The website will now be deployed on every successful build using this script:

deploy:  
 - provider: pages  
 skip-cleanup: true  
 github-token: $GITHUB\_TOKEN  
 keep-history: false  
 local-dir: docs  
 on:  
 branch: master

for more info on deploying pages you can check the [Travis CI guide on pages](https://docs.travis-ci.com/user/deployment/pages/).

**ImageMagick and Travis CI**

Inside the [travis-ci-community](https://travis-ci.community/t/error-configuration-failed-for-package-magick/1674) there was a question on how to install the magick package on Travis-CI. The answer is simple. You need to have all system dependencies of ImageMagick. Install these for Linux by:

addons:  
 apt:  
 update: true  
 sources:  
 - sourceline: 'ppa:opencpu/imagemagick'  
 - sourceline: 'ppa:ubuntugis/ppa'  
 packages:  
 - libmagick++-dev  
 - librsvg2-dev  
 - libwebp-dev  
 - libpoppler-cpp-dev  
 - libtesseract-dev  
 - libleptonica-dev  
 - tesseract-ocr-eng  
 - r-cran-rgdal  
 - libfftw3-dev  
 - cargo

This also worked for macOS for me.