R Package Development: From Idea to CRAN

Complete tutorial for creating your first R package

Table of contents

## 1 Learning Objectives

By the end of this tutorial, you will: - Set up a proper R package development environment - Create package structure and documentation - Write and test package functions - Prepare for CRAN submission

## 2 Prerequisites

* Basic R programming knowledge
* RStudio installed
* Git familiarity (helpful but not required)

## 3 Step 1: Development Environment Setup

First, install the essential packages for R development:

install.packages(c("devtools", "usethis", "roxygen2", "testthat"))

Configure your development environment:

library(usethis)  
use\_git\_config(user.name = "Your Name", user.email = "your.email@example.com")

## 4 Step 2: Create Package Structure

Create a new package:

create\_package("~/path/to/mypackage")

This creates the standard package directory structure: - R/ - Your R functions - man/ - Documentation files (auto-generated) - DESCRIPTION - Package metadata - NAMESPACE - Exported functions (auto-generated)

## 5 Step 3: Write Your First Function

Create a new R file in the R/ directory:

#' Add two numbers together  
#'  
#' This function takes two numeric inputs and returns their sum.  
#'  
#' @param x A numeric value  
#' @param y A numeric value  
#' @return The sum of x and y  
#' @export  
#' @examples  
#' add\_numbers(2, 3)  
#' add\_numbers(10, -5)  
add\_numbers <- function(x, y) {  
 if (!is.numeric(x) || !is.numeric(y)) {  
 stop("Both inputs must be numeric")  
 }  
 x + y  
}

## 6 Step 4: Generate Documentation

Use roxygen2 to generate documentation:

devtools::document()

This creates help files in the man/ directory and updates your NAMESPACE.

## 7 Step 5: Testing

Create unit tests to ensure your functions work correctly:

usethis::use\_testthat()  
usethis::use\_test("add\_numbers")

Write tests in tests/testthat/test-add\_numbers.R:

test\_that("add\_numbers works correctly", {  
 expect\_equal(add\_numbers(2, 3), 5)  
 expect\_equal(add\_numbers(-1, 1), 0)  
 expect\_error(add\_numbers("a", 1))  
})

Run tests:

devtools::test()

## 8 Step 6: Package Checks

Before submitting to CRAN, run comprehensive checks:

devtools::check()

This runs R CMD check and identifies potential issues.

## 9 Step 7: Preparing for CRAN

Update your DESCRIPTION file with proper metadata:

Package: mypackage  
Title: What the Package Does (One Line, Title Case)  
Version: 0.1.0  
Authors@R:   
 person("First", "Last", , "first.last@example.com", role = c("aut", "cre"))  
Description: What the package does (one paragraph).  
License: MIT + file LICENSE  
Encoding: UTF-8  
Roxygen: list(markdown = TRUE)  
RoxygenNote: 7.2.3  
Suggests:   
 testthat (>= 3.0.0)  
Config/testthat/edition: 3

## 10 Next Steps

* Add more functions and documentation
* Create vignettes for complex workflows
* Set up continuous integration
* Submit to CRAN when ready

## 11 Resources

* [R Packages book](https://r-pkgs.org/) by Hadley Wickham
* [Writing R Extensions](https://cran.r-project.org/doc/manuals/r-release/R-exts.html) manual
* [CRAN Policy](https://cran.r-project.org/web/packages/policies.html)