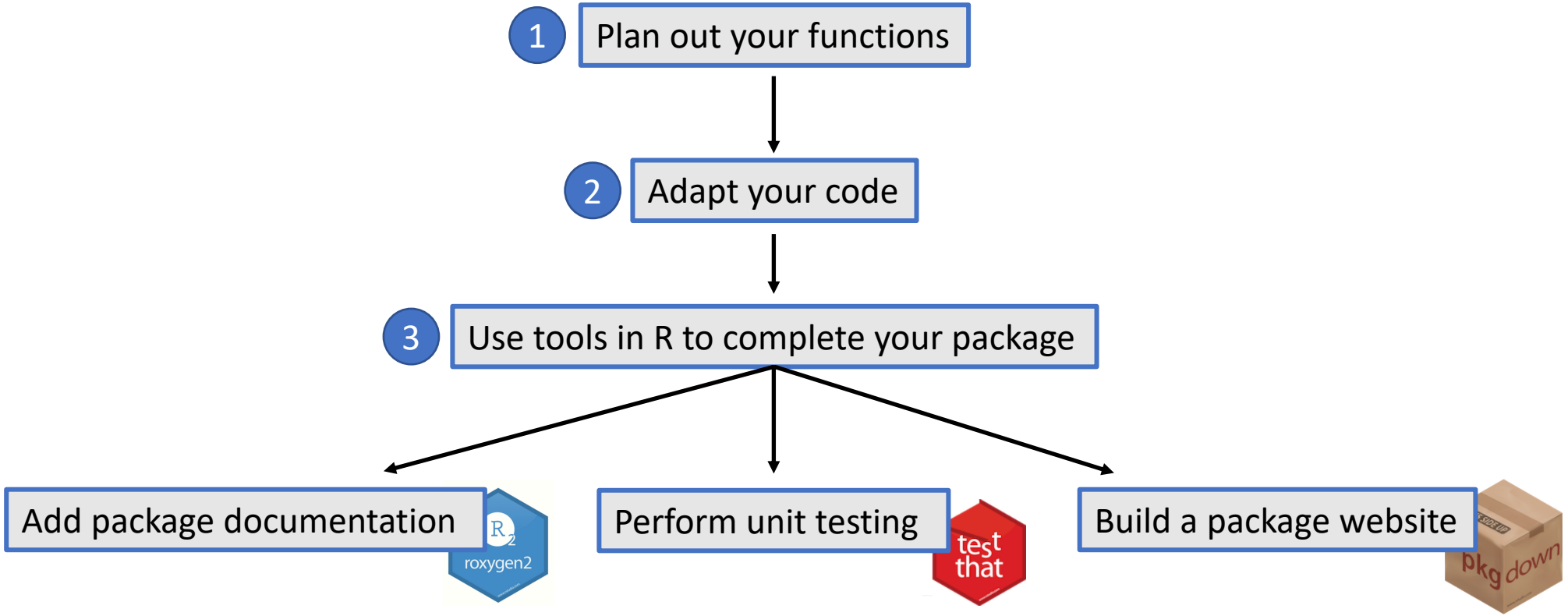
If you can write code, you can build a package!

Don’t be intimidated by the thought of turning your code into a package! Take time to do some thoughtful planning up front, and let R’s package tools do the rest.



Hadley Wickham’s book on R Packages is great tool to walk you through the process of package creation: http://r-pkgs.had.co.nz/

Planning out how to structure your package will simplify the process later on!

What do you want your package to accomplish?

- What is the scope?
- Are there additional functions that would be helpful to include?
- How much flexibility do you want to build into your functions?

What structure will fit best with your goal?

- What do you want your function inputs and outputs to look like? Do you want your functions to modify values or data tables?
- How can you build your functions in a way that makes the package feel cohesive?
- What naming conventions do you want to adopt?

Talk to potential users of your package to understand how you can adapt your functions to better serve their needs

Try to align the structure of your functions, it will help users learn how to use all your functions

Add object documentation with roxygen2

Why should you take time to add detailed object documentation?

- Crucial to making your package usable by others
- All you have to do is write it out and roxygen2 will handle the rest

```
#' Add 3 to a given value
#'
#' \code{add_three} returns the inputted value + 3
#'
#' @param x A numeric.
#'
#' @return The output is a numeric.
#'
#' @family helper functions
#' @export
add_three <- function(x){x+3}
```

devtools::document()

Add 3 to a given value

Description

`add_three` returns the inputted value + 3

Usage

```
add_three(x)
```

Arguments

`x`	A numeric.

Value

The output is a numeric.

Perform unit testing on your functions using testthat()

Why should you perform formal automated testing for your package?

- Documentation that your functions work properly
- Ability to quickly re-test code if you make updates.

How do you create documented unit testing?

- Within the testthat() function, use expect_equal() to compare the output of a function to what you would expect the function to produce if it is working correctly
- Repeat for all of your relevant functions

```
test_that("add_three correctly adds three", {

  expect_equal(add_three(5), 8)
  expect_equal(add_three(10.5), 13.5)
  expect_equal(add_three(0), 3)
  expect_equal(add_three(-0.7), 2.3)

})
```

Create a website to help share your package with others using pkgdown

Why should you create a website for your package?

- Share your package with others in an easy to navigate way
- Consolidate all of your background documents in one place

- Spend time on writing your tutorials and demos—the more detailed these are the easier it will be for others to understand how to implement your package
- pkgdown::build_site() will put all the elements of your package into a pretty website!