1) Console, install several packages to help create own package

install.packages(c(“devtools”, “roxygen2”, “testthat”, knitr”, “usethis”)

2) Check R Studio API Package & have the latest version

install.packages(“rstudioapi”)
rstudioapi::isAvailable(“1.4.1103”)

3) Access the newest devtools functions

devtools::install_github(“r-lib/devtools”)

4) Macbook (Xcode or Command line tools for X code)

5) Standard File Structure
    - R/ directory with R code
    - Basic DESCRIPTION file with package metadata
    - Basic NAMESPACE file

6) Pick a name for package
   Name only consists of letters, numbers, periods
   Must start with letter, cannot end with period
   Do not use hyphens, underscores
   Unique
   Avoid both upper & lower case letters

   -> Check if a name exists on CRAN

- http://cran.r-project.org/web/packages/[mypackagename].

OR check it using the {available} package to check if the name is used elsewhere

library(available)

available(“ggplot2”)

7) Create a NEW “Bare-Bones” Package

   Pick a folder location for new package that is NOT in an existing RStudio Project or Git Repository

   Create packages using the console

   {usethis} package has functions designed to ease the process

create_package(path) to create a “bare-bones” -> invoke a new R studio session

#create_package("../../../../R_packages/testpackage")

Test the Initial Bare-Bones Package

-> test the bare-bones package to see if it will build/compile and install properly

8) Choose a License for the Package

MIT License

GNU GPLv3

#use_mit_license("copyright holder name")

#use_gpl3_license()

#use_apl2_license()

#use_cc0_license()

9) Write Code and Document Functions

Writing code for a package

10) Restart R session

11) Build and Reload

12) Test Package
13) Check Package
14) Document Package