Example of R Package and Unit Test

Create a package that contains two functions

1. Function 1: Raise the numerical variable x to twice of another numerical variable y. $(x^2)^y$
2. Function2:
   gets two numerical vector of the same size (x and y) and create a data frame to create a new vetors which is x*y.

Install and load packages:

- {usethis}, {roxygen2}, {testthat}, {knitr}, {covr} and {devtools}

Process:

Step 1:

- Create a folder called "powpack"

Step 2:

- Set your working Directory to this folder

Step 3:

- Type `usethis::create_package(path = ".", rstudio = FALSE, open = FALSE)` to create package

Step 4:

- Type `usethis::use_r(name = "power1")` to create an R script in R folder called "power1.R"

Step 5:

- Need to write the functions in the blank R script "Power1.R" that we just created. First start with Function 1

FUNCTION ONE:

```
 pwr1 <- function(x, y){
  if (!is.numeric(x) | !is.numeric(y)){
    return(cat("x =", x, "and y=", y,
               ". Both must be numeric. Please enter numeric # values for both"))
  }
 return((x^2)^y)
}
```

Step 6:

- `Check()` if the function works as we expected. Type `devtools::check()`
- If you have only one warning, the warning is to tell you that you haven't set a license.
- If you want to add a license you can edit your "DISCRIPTION" file by typing `usethis::use_mit_license()`
- Now mit license has been added to your DISCRIPTION
- If you run `devtools::check()` you will see there is no warning.

Step 7:

- Make documentation. Put the curser inside of the function and go to

"Code > Insert Roxygen Skeleton"

Step 8:

- Run `devtools::check()`

Step 9:

- Type `devtools::document()` to create the `.rd` file in your `man` folder

Step 10:

- Load the package into memory by `devtools::load_all()` or install it by `devtools::install()`

Now we can work on the function two

```
pwr2 <- function(x, y){
 if (class(x) != "numeric" | class(y) != "numeric") {
   stop("Check the class of your vectors both must be numeric")
 }
 else if (sum(is.na(x) | is.na(y)) !=0){
   stop("There is at least one missing value in your vectors")
 }

 else if (length(x) %% length(y) !=0 | length(y) %% length(x) !=0 )
 { warning(" The # lengths are not equal but one is multiple of the other.
            Do you want R to perform recycle?")}

 else if (length(x) == length(y) & sum(is.na(x) | is.na(y)) ==0 ){
   df <- tibble::tibble(x, y)
   return(dplyr::mutate(df, product = x *y))
 }  else { stop("Check your vectors!")

 }
}
```

Make documentation:

- Put the curser inside of the function and go to "Code > Insert Roxygen Skeleton"

- Run the `devtools::check()`. You may get one error and one warning. The error is because you have used two packages {tibble} and {dplyr} and you have not loaded them into function. You can not use `library()`. You need to import thsese two function to DESRIPTION file. The best way is to use

`usethis::use_package("tibble")` and `usethis::use_package("dplyr")`

- Now these two packages are in your DESCRIPTION file.
- Now add a license for this function

Type `devtools::document()` to create the `.rd` file in your `man` folder

- Load the package into memory by `devtools::load_all()` or install it by `devtools::install()`

Unit Test

- Now we want to test our functions formally.

Step 1:

- Type `usethis::use_testthat()` to create the `tests` folder on your package.

Step 2:

- `Type usethis::use_test("power1.R")` to open th R Script `test-power1.R`

By default you have an example of multiplication. You may run this and it will return pass.

Inside of `test-power1.R` type the following

```
test_that("The Power function meets the expectation", {
expect_equal(pwr1(-2, 0.5), 2)
expect_equal(pwr1(2.3, 1.2), (2.3)^(2*1.2))
expect_equal(pwr1(10, -2), 10^(-4))

})


test_that("unknown object should return erroror missing argument", {
 expect_error(pwr1(-2, a))
 expect_error(pwr1(-2, a), "object 'a' not found")
 expect_error(pwr1(b, 2))
```

```
  expect_error(pwr1(b, 2), "object 'b' not found")
  expect_error(pwr1(2,))
  expect_error(pwr1(2,), "argument \"y\" is missing, with no default")
})
```

SECOND FUNCTION

```
test_that("The Product function meets the expectation", {
  expect_error(pwr2(c(NA, 1), c(1,2)))
  expect_error(pwr2(c(NA, 1),
                    c(1,2)), "There is at least one missing value in your vectors")
 expect_error(pwr2(c(2,1),c("a",1)))
 expect_error(pwr2(c(2,1),c("a",1)),
              "Check the class of your vectors both must be numeric")
  expect_warning(pwr2(c(2, 1), c(1,2, 3, 4)))
  expect_warning(pwr2(c(2, 1), c(1,2, 3, 4)),
                 "The lengths are not equal but one is multiple of the other.
                 Do you want R to perform recycle?")

})
```