DATA501 Assignment 1: Unit Testing

Sharna Granwal, 300622554

2023-08-04

Overview

This document outlines the changes made to the mylm package, including the unit tests added using testthat library and the corresponding changes to the function "mylm()". The updated package with all additional and modified files can be found at: **https://github.com/sharnag/mylm/tree/main**

Unit Test File

The following tests were included in the test file "test_mylm.R", covering four areas.

1. Input validation tests for *valid* inputs.
   The "valid" inputs were determined based on documentation inside the function. i.e.

- **formula** an object of class "formula" (or one that can be coerced to that class) a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
- **data** an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
- **subset** an optional vector specifying a subset of observations to be used in the fitting process.

```
test_that("mylm works with valid formula, data and subset arguments", {
  #Create data objects and environment variables for model parameters
  mtcars <- mtcars
  mtcars_list <- as.list(mtcars)
  mpg <- mtcars[,"mpg"]
  wt <- mtcars[,"wt"]
  disp <- mtcars[,"disp"]
  hp <- mtcars[,"hp"]

  # dataframe
  expect_silent(mylm(mtcars, data=mtcars))
  expect_silent(mylm(formula=mpg ~ wt + disp + hp, data=mtcars))
  expect_silent(mylm(formula="mpg ~ wt + disp + hp", data=mtcars))

  # list
  expect_silent(mylm(formula=mpg ~ wt + disp + hp, data=mtcars_list))

  # data is NULL but environment vars exist.
```

```
  expect_silent(mylm(formula=mpg ~ wt + disp + hp))

  # valid subsets
  expect_silent(mylm(formula=mpg ~ wt + disp + hp, data=mtcars, subset=c(1:25)))
  expect_silent(mylm(formula=mpg ~ wt + disp + hp, data=NULL, subset=c(1:25)))
  expect_silent(mylm(formula=mpg ~ wt + disp + hp, data=mtcars,
                     subset=c(rep(TRUE,25),rep(FALSE,7))))
})
```

2. Input validation tests for *invalid* inputs.
   A range of invalid inputs were tested, with helpful errors generated.

```
test_that("mylm gives helpful errors for invalid formula argument", {
  expect_error(mylm(), "argument \"formula\" is missing, with no default")
  expect_error(mylm("mpg = wt + disp + hp", data=mtcars),
               "The argument \"formula\" is not a valid formula") # = is deprecated
  expect_error(mylm("abc", data=mtcars),
               "The argument \"formula\" is not a valid formula")
  expect_error(mylm(123, data=mtcars),
               "The argument \"formula\" is not a valid formula")
})

test_that("mylm gives helpful errors for invalid data argument", {
  expect_error(mylm(mpg ~ wt + disp + hp, data="mtcars"),
               "The argument \"data\" is not a dataframe or list object")
  # check if model parameters are missing from the dataframe
  expect_error(mylm(formula=random ~ wt + disp + hp, data=mtcars),
               "The response variable is undefined")
  expect_error(mylm(mpg ~ wt + disp + hp + random, data=mtcars),
               "object 'random' not found")
  # data is NULL. Check environment variables exist and do not generate an error
  # when creating the model matrix.
  expect_error(mylm(random ~ wt + var1),
  "There was an error generating the dataframe from the environment variables.")
  expect_error(mylm(mpg ~ wt + var1),
  "There was an error generating the dataframe from the environment variables.")
  invalid_param <- c(1:10)
  expect_error(mylm(formula=mpg ~ wt + disp + hp + invalid_param),
  "There was an error generating the dataframe from the environment variables.")
})

test_that("mylm gives helpful errors for invalid subset argument", {
  expect_error(mylm(mpg ~ wt + disp + hp, data=mtcars, subset=mtcars),
               "The argument \"subset\" is not a vector")
  expect_error(mylm(mpg ~ wt + disp + hp, data=mtcars, subset="abc"),
               "The argument \"subset\" is not a numerical or logical vector")
  expect_error(mylm(mpg ~ wt + disp + hp, data=mtcars, subset=c(-2:0)),
               "The subset is out of range")
  expect_error(mylm(formula=mpg ~ wt + disp + hp, data=mtcars, subset=c(1:25, 55)),
               "The subset is out of range")

})
```

3. Functionality tests to test the output.
   We test that the output class is of the correct class (i.e. "mylm") and that the regression results are correct. The values are compared against the output of the "lm()" function for the same input.

```
test_that("mylm produces valid output", {
  # Create mylm and lm objects from the same data
  result <- mylm(formula=mpg ~ wt + disp + hp, data=mtcars)
  result_lm <- lm(formula=mpg ~ wt + disp + hp, data=mtcars)

  # Check object of expected class is created
  expect_s3_class(result, "mylm")
  expect_type(result$coef, "double")
  expect_type(result$call, "language")

  # Check/spot check coefficients, std dev, residuals and fitted values are
  # the same as those fitted by lm()
  expect_equal(result$coef,result_lm$coef)
  expect_equal(result$sigma, summary(result_lm)$sigma)
  expect_equal(result$residuals[[2]], result_lm$residuals[[2]])
  expect_equal(result$fitted.values[[3]], result_lm$fitted.values[[3]])

})
```

4. Additional tests to test related functions.
   The "expect_snapshot" function was used to ensure the expected output of these related functions remains the same when run against the "mylm" object.

```
test_that("mylm output can be used with additional functions",  {
  # Create mylm and lm objects from the same data
  result <- mylm(formula=mpg ~ wt + disp + hp, data=mtcars)
  result_lm <- lm(formula=mpg ~ wt + disp + hp, data=mtcars)

  expect_snapshot(summary(result))
  expect_snapshot(residuals(result))
  expect_snapshot(fitted(result))
  expect_snapshot(vcov(result))
  expect_snapshot(confint(result))
  expect_snapshot(print(result))
  expect_snapshot(print(summary(result)))
  expect_silent(plot(result)) # snapshot does not support plots
})
```

Code Changes

A number of code changes were made to the mylm() function in order to pass the unit tests.

1. Validation of the "formula" input.

```
# Check if "formula" argument is null
  if(is.null(formula)){
  } # if null, system error thrown before continuing
```

```
  # Try to coerce the "formula" argument to an object of type formula
  # This should succeed for formula objects and strings of the correct format
  tryCatch(
    expr = {formula <-as.formula(formula)},
    error = function(e){stop('The argument \"formula\" is not a valid formula')},
    warning = function(w){stop('The argument \"formula\" is not a valid formula')}
  )
```

2. Validation of the "data" input.

```
# Check if "data" argument is null
  if(is.null(data)){
    # Create data frame using environment variables based on formula by
    # re-using functionality in model matrix, then convert to a new dataframe
    tryCatch(
      expr = {data <- data.frame(model.matrix(formula,data=environment(formula)))},
      error = function(e){stop('There was an error generating the
                               dataframe from the environment variables.')}
    )
    # Add the response variable to the dataframe
    data$y <- eval(parse(text = formula[[2]]), envir=parent.frame())
    colnames(data)[which(colnames(data) == 'y')] <- format(formula[[2]])
  }

  # Check if "data" argument is a list or dataframe object
  if(!is.data.frame(data) & !is.list(data)){
    stop('The argument \"data\" is not a dataframe or list object')
  }

  # Convert list object to a dataframe
  if(is.list(data)){
    data <- as.data.frame(data)
  }
```

To allow for an optional data frame, the function arguments were changed to allow NULL to be passed as an argument for "data" (in addition to the changes mentioned prior):

```
mylm <- function(formula, data=NULL, subset=NULL){
  ...
}
```

3. Validation of the "subset" input.

```
# Check the "subset" argument is a numerical or logical vector
  if(!is.null(subset)){
    if(!is.vector(subset)){
      stop('The argument \"subset\" is not a vector')
    }
    if(is.vector(subset) & !is.numeric(subset) & !is.logical(subset)){
      stop('The argument \"subset\" is not a numerical or logical vector')
    }
```

```
    # Check "subset" argument is in valid range
    if(is.numeric(subset)){
      if(min(subset)<0 ||max(subset) > nrow(data)){
        stop('The subset is out of range')
      }
    }
    data <- data[subset,]
  }
```

4. Additional validation was added to check the response variable is defined.

```
tryCatch(
    expr={yvec <- data[,yname]},
    error = function(e){stop('The response variable is undefined')}
  )
```

Running the Tests

After the function was updated and the test file created, the tests were run using the following code:

```
library(mylm)
library(testthat)
#devtools::clean_dll() # required to clear an error when re-running
devtools::test()
```

The resulting output indicates are tests were passed successfully.

```
ℹ Testing mylm
✔ | F W S  OK | Context
✔ |         37 | mylm [0.2s]

══ Results ═══════════════════════════════
Duration: 0.2 s

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 37 ]

Woot!
```