Using `car` and `effects` Functions in Other Functions

John Fox*& Sanford Weisberg†

June 26, 2021

Abstract

The `car` package (Fox and Weisberg, 2019) provides many functions that are applied to a fitted regression model, perform additional calculations on the model or possibly compute a different model, and then return values and graphs. In some cases, users may wish to write functions that call functions in `car` for a particular purpose. Because of the scoping rules used in R, several functions in `car` that work when called from the command prompt may fail when called inside another function. We discuss how users can modify their programs to avoid this problem.

Some users of the `car` and `effects` package have found it convenient to write their own functions that call the functions in `car` or `effects`. While this will generally occur as expected, in some instances calls to `car` or `effects` functions will fail because the results of an input fitted model may not be available inside a user-written function. This brief note describes how this problem can be solved.

For an illustration of the problem, the function `car::ncvTest` (Fox and Weisberg, 2019, Sec. 8.5.1) computes tests for non-constant variance in linear models as a function of the mean, the default, or any other linear function of regressors, even for regressors not part of the mean function. For example,

```
m2 <- lm(prestige ~ education, data=carData::Prestige)
car::ncvTest(m2, ~ income)
```

```
Non-constant Variance Score Test
Variance formula: ~ income
Chisquare = 1.521, Df = 1, p = 0.22
```

This fits `prestige` as a linear function of `education`, and tests for nonconstant variance as a function of `income`, another regressor in the data set `Prestige`. Embedding this in a function fails:

*Department of Sociology, McMaster University
†School of Statistics, University of Minnesota

```
f3 <- function(meanmod, dta, varmod) {
  m3 <- lm(meanmod, dta)
  car::ncvTest(m3, varmod)
  }
f3(meanmod=prestige ~ education, dta=carData::Prestige, varmod ~ income)
```

```
Error in eval(data, envir = environment(formula(model))) :
 object 'dta' not found
```

The arguments `dta` and `meanmod` are defined in the environment of the function, but the call to `lm` looks for them in the global environment, and they are therefore invisible when `lm` is called. A solution is to copy `dta` to the global environment.

```
f4 <- function(meanmod, dta, varmod) {
   assign(".dta", dta, envir=.GlobalEnv)
   assign(".meanmod", meanmod, envir=.GlobalEnv)
   m1 <- lm(.meanmod, .dta)
   ans <- car::ncvTest(m1, varmod)
   remove(".dta", envir=.GlobalEnv)
   remove(".meanmod", envir=.GlobalEnv)
   ans
   }
f4(prestige ~ education, carData::Prestige, ~income)
```

```
Non-constant Variance Score Test
Variance formula: ~ income
Chisquare = 1.521, Df = 1, p = 0.22
```

The `assign` function copies the `dta` and `meanmod` arguments to the global environment where `ncvTest` will be evaluated, and the `remove` function removes them before exiting the function. This is an inherently problematic strategy, because an object assigned in the global environment will replace an existing object of the same name. Consequently we renamed the `dta` argument `.dta`, with an initial period, but this is not a *guarantee* that there was no preexisting object with this name.

The functions `effects::Effect` and `effects::predictorEffect` may fail similarly when embedded in user-written functions because of scoping. Assigning arguments to the global environment as illustrated with the `car::ncvTest` function can again be applied.

The following function will fail:

```
fc <- function(dta, formula, terms) {
 if (!require("effects")) stop("effects package unavailable")
 print(m1 <- lm(formula, dta))
 Effect(terms, m1)
 }
```

```
 form <- prestige ~ income*type + education
 terms <- c("income", "type")
 fc(carData::Duncan, form, terms)
```

```
Error in is.data.frame(data) : object 'dta' not found
```

Assigning `.dta` to the global environment solves the problem:

```
 fc.working <- function(dta, formula, terms) {
  if (!require("effects")) stop("effects package unavailable")
  assign(".dta", dta, env=.GlobalEnv)
  print(m1 <- lm(formula, .dta))
  e1 <- Effect(terms, m1)
  remove(".dta", envir=.GlobalEnv)
  e1
 }
 form <- prestige ~ income*type + education
 terms <- c("income", "type")
 fc.working(carData::Duncan, form, terms)
```

```
Call:
lm(formula = formula, data = .dta)

Coefficients:
    (Intercept)           income         typeprof           typewc
         -4.732            0.776           31.712           -1.637
      education  income:typeprof    income:typewc
          0.357           -0.370           -0.360


 income*type effect
      type
income    bc  prof    wc
    7  19.48 48.60 15.32
    30 37.33 57.95 24.88
    40 45.09 62.01 29.04
    60 60.61 70.14 37.35
    80 76.14 78.27 45.67
```

Assigning `formula` to the global environment is not necessary here because it is used by `lm` but not by `Effect`.

References

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, 3rd edition, 2019. URL `http://z.umn.edu/carbook`.