Using \texttt{car} and \texttt{effects} Functions in Other Functions

John Fox* & Sanford Weisberg†

August 10, 2020

Abstract

The \texttt{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 \texttt{car} for a particular purpose. Because of the scoping rules used in \texttt{R}, several functions in \texttt{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.

1 ncvTest

The function \texttt{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,

\begin{verbatim}
  m2 <- lm(prestige ~ education, Prestige)
  ncvTest(m2, ~ income)
\end{verbatim}

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

This fit \texttt{prestige} as a linear function of \texttt{education}, and tests for nonconstant variance as a function of \texttt{income}, another regressor in the data set \texttt{Prestige}. Embedding this in a function fails:

\begin{verbatim}
f3 <- function(meanmod, dta, varmod) {
  m3 <- lm(meanmod, dta)
  ncvTest(m3, varmod)
}
f3(prestige ~ education, Prestige, ~ income)
\end{verbatim}

\footnote{Department of Sociology, McMaster University}

\footnote{School of Statistics, University of Minnesota}
Error in is.data.frame(data) : 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.

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

This same method can be used with functions in the `effects` package. Suppose, for example, you want to write a function that will fit a model, provide printed summaries and also draw a effects plot. The following function will fail:

```r
library(effects)
f5 <- function(dta, formula, terms) {
  print(m1 <- lm(formula, .dta))
  Effect(terms, m1)
}
form <- prestige ~ income*type + education
terms <- c("income", "type")
f5(Duncan, form, terms)
```

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

As with `ncvTest`, `dta` will not be in the correct environment when `Effect` is evaluated. The solution is to copy `dta` to the global environment:
library(effects)
fc.working <- function(dta, formula, terms) {
  assign(".dta", dta, env=.GlobalEnv)
  print(m1 <- lm(formula, .dta))
  e1 <- Effect(terms, m1)
  remove(".dta", envir=.GlobalEnv)
  e1
}
fc.working(Duncan, form, terms)

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

2 Bootstrap

The Boot function (Fox and Weisberg, 2019, Sec. 5.1.3) in car provides a conve-
nience front-end for the function boot in the boot package (Canty and Ripley,
2013). With no arguments beyond the name of a regression object and the
number of replications R, Boot creates the proper arguments for boot for case
resampling bootstraps, and returns the coefficient vector for each sample:

\[
m1 <- lm(time \sim t1 + t2, Transact)
\]
\[
b1 <- Boot(m1, R=999)
\]
\[
summary(b1)
\]

Number of bootstrap replications R = 999

<table>
<thead>
<tr>
<th></th>
<th>original</th>
<th>bootBias</th>
<th>bootSE</th>
<th>bootMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>144.37</td>
<td>4.89815</td>
<td>198.357</td>
<td>148.46</td>
</tr>
<tr>
<td>t1</td>
<td>5.46</td>
<td>0.02905</td>
<td>0.690</td>
<td>5.51</td>
</tr>
<tr>
<td>t2</td>
<td>2.03</td>
<td>-0.00406</td>
<td>0.157</td>
<td>2.03</td>
</tr>
</tbody>
</table>

The returned object b1 is of class "boot", as are objects created directly from
the boot function, so helper functions in the boot package and in car can be
used on these objects, e.g.,

\[
confint(b1)
\]

Bootstrap bca confidence intervals

<table>
<thead>
<tr>
<th></th>
<th>2.5 %</th>
<th>97.5 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-271.612</td>
<td>518.427</td>
</tr>
<tr>
<td>t1</td>
<td>3.865</td>
<td>6.625</td>
</tr>
<tr>
<td>t2</td>
<td>1.772</td>
<td>2.405</td>
</tr>
</tbody>
</table>

The Boot function would have scoping problems even without the user em-
bedding it in a function because the boot function called by Boot tries to evaluate
the model defined in the global environment in a local environment. In car
we define an environment
.carEnv <- new.env(parent=emptyenv())

and then evaluate the model in the environment .carEnv. This environment is not exported, so to see that it exists you would need to enter car:::.carEnv. We use this same trick in the Boot.default function so that .carEnv is globally visible. Here is a copy of Boot.default to show how this works.

Boot.default <- function(object, f=coef, labels=names(coef(object)),
R=999, method=c("case", "residual")){
  if(!(require(boot))) stop("The 'boot' package is missing")
  f0 <- f(object)
  if(length(labels) != length(f0)) labels <- paste("V", seq(length(f0)), sep="")
  method <- match.arg(method)
  if(method=="case") {
    boot.f <- function(data, indices, .fn) {
      assign(".boot.indices", indices, envir=car:::.carEnv)
      mod <- update(object, subset=get(".boot.indices", envir=car:::.carEnv))
      if(mod$qr$rank != object$qr$rank){
        out <- .fn(object)
        out <- rep(NA, length(out)) } else {out <- .fn(mod)}
      out
    }
  } else {
    boot.f <- function(data, indices, .fn) {
      first <- all(indices == seq(length(indices)))
      res <- if(first) object$residuals else
        residuals(object, type="pearson")/sqrt(1 - hatvalues(object))
      res <- if(!first) (res - mean(res)) else res
      val <- fitted(object) + res[indices]
      if (!is.null(object$na.action)){
        pad <- object$na.action
        attr(pad, "class") <- "exclude"
        val <- naresid(pad, val)
      }
      assign(".y.boot", val, envir=car:::.carEnv)
      mod <- update(object, get(".y.boot", envir=car:::.carEnv) ~ .)
      if(mod$qr$rank != object$qr$rank){
        out <- .fn(object)
        out <- rep(NA, length(out)) } else {out <- .fn(mod)}
      out
    }
  }
  b <- boot(data.frame(update(object, model=TRUE)$model), boot.f, R, .fn=f)
  colnames(b$b$t) <- labels
  if(exists(".y.boot", envir=car:::.carEnv))
    remove(".y.boot", envir=car:::.carEnv)
  if(exists(".boot.indices", envir=car:::.carEnv))
}
remove(".boot.indices", envir=car:::carEnv)

References

Angelo Canty and Brian Ripley. boot: Bootstrap R (S-Plus) functions. R package version 1.3-9, 2013.