The Default Effect Method

John Fox and Sanford Weisberg

March 28, 2018

The `effects` package in `R` is designed primarily to draw graphs that visualize a fitted response surface of a fitted model in problems with a linear predictor. Many modeling paradigms that can be fit with base `R` or contributed packages fit into this framework, including methods for linear, multivariate linear, and generalized linear models fit by the standard `lm` and `glm` functions and by the `svyglm` function in the `survey` package [Lumley, 2004]; linear models fit by generalized least squares using the `gls` function in the `nlme` package [Pinheiro et al., 2016]; multinomial regression models fit by `multinom` in the `nnet` package [Venables and Ripley, 2002]; ordinal regression models using `polr` from the `MASS` package [Venables and Ripley, 2002] and `clm` and `clm2` from the `ordinal` package [Christensen, 2015]; linear and generalized linear mixed models using the `lme` function in the `nlme` package [Pinheiro et al., 2016] and the `lmer` and `glmer` functions in the `lme4` package [Bates et al., 2015]; and latent class models fit by `poLCA` in the `poLCA` package [Linzer and Lewis, 2011]. This is hardly an exhaustive list of fitting methods that are based on a linear predictor, and we have been asked from time to time to write functions to use `effects` with this other fittig methods.

The default `Effect.default` may work with some modeling functions, as would objects of the class `gls` that we describe below in Section 1. This will will work if your function recognizes the defaults for the arguments in the `sources` list described in Section 1. If the defaults don't work, you will need to create your own `Effect` method or call `Effect.default` with your own value of `sources`.

The `effect` package has five functions that create the information needed for drawing effects plots, `Effect`, `allEffects`, `effect` and `predictorEffect` and `predictorEffects`. To add new modeling to the package only `Effect` needs to be written; the package will take care of all the other functions.

1 Generalized Least Squares

The `gls` function in the `nlme` package [Pinheiro et al., 2016] fits linear models via generalized least squares. A call to `gls` creates an object of class `gls`. The following function will allow usage of such objects with the `effects` package.

```
R> Effect.gls <- function(focal.predictors, mod, ...){
+      args <- list(
+        type = "glm",
+        call = mod$call,
+        formula = formula(mod),
+        family = family(mod),
+        link = NULL,
+        coefficients = coef(mod),
+        vcov = as.matrix(vcov(mod)))
+      Effect.default(focal.predictors, mod, ..., sources=args)
+    }
```

This function consists of a call the the `Effect.default` function. The argument `sources` to that function contains the information needed is a list of up to five arguments:

`type` The `effects` package has three basic modeling functions: `type = "glm"`, the default, is used for functions with a univariate response and a linear predictor and possibly a link function. This class includes linear models, generalized linear models, robust regression, generalized least squares fitting, linear and generalized linear mixed effects models and many others. The other types are type = "multinom" for multinomial log-linear models as fit by the multinom function in nnet and similar, and type = "polr" for ordinal regression models, as in the polr function in the MASS package. Examples of using these types are given later.

call Effects uses the call to harvest additional arguments that it needs such as data, subset and family. These are needed to compute fitted value, and to interpret the terms in the formula for the linear predictor. The default is mod$call for S3 objects and mod@call for S4 objects.

formula In most cases the formula for the linear predictor is returned by formula(mod), the default, but if this is not the case then the the value of this argument should be the value of the formula.

family and link GLM-like models include a family specifying both an error distrubtion and a link function. If the call family(mod) returns a family, you do not need to specify either of these values. The betareg function, on the other hand, fixes the error distribution as a Beta distribution, but it does include a link function using a link arguement. In this case, you must specify the link function; see the betareg example in Section 4 below. If you have a family, but it is not returned by family(mod), use the family argument to specify it.

coefficients In many cases the coefficient estimates are returned by coef(mod), the default, but if this is not the case then the the value of this argument should be the estimates of the coefficients in the linear predictor.

vcov In many cases the estimated covariance matrix of the coefficient estimates is returned by vcov(mod), the default, but if this is not the case then the the value of this argument should be the estimates of the estimated covariance matrix of the coefficient estimates in the linear predictor.

Since the values of all the arguments in sources are default values for the gls function, there is no need to have written the Default.gls method, as the default method would work.

2 Mixed Effects with the lmer from the lme4 package

The lme4 package [Bates et al., 2015] fits linear and generalized linear mixed effects models with the lmer and glmer functions, respectively. The same Effect function can be used for both types of models.

```
R> Effect.merMod <- function(focal.predictors, mod, ..., KR=FALSE){
+      if (KR && !requireNamespace("pbkrtest", quietly=TRUE)){
+        KR <- FALSE
+        warning("pbkrtest is not available, KR set to FALSE")}
+      fam <- family(mod)
+      args <- list(
+        call = mod@call,
+        coefficients = lme4::fixef(mod),
+        vcov = if (fam == "gaussian" && fam$link == "identity" && KR)
+          as.matrix(pbkrtest::vcovAdj(mod)) else as.matrix(vcov(mod)))
+      Effect.default(focal.predictors, mod, ..., sources=args)
+    }
```

This method is somewhat more complicated because it adds an additional argument to Effect to choose the method of estimating the sample covariance matrix. If KR=TRUE, the vovAdj function in the pbkrtest package is used to compute the sample covariance matrix. Because this function is (painfully) slow, the default is KR=FALSE. Because lmer is an S4 object (tested using the isS4 function), the default for call is mod@call, and this argument would have been set authomatically had we not included it in the above fucntion. The coefficient for an object created by a call to lmer or glimer are not returned by coef(mod), so the value of coefficients is the value returned by lme4::fixef(mod). The vcov estimate contains its estimated variance covariance matrix of the fixed effects.

The formula for a mixed-effects model in the lme4 package specifies both the linear predictor in the mean function and the linear predictor(s) in the variance functions in terms with parentheses and and vertical bars such as (1 + age | subject). The effects code will automatically remove any terms like these.

3 Robust Linear Mixed Models

The robustlmm package [Koller, 2016] fits linear mixed models with a robust estimation method. As iits rlmer function closely parallels the lmer package, an object created by rlmer is easily used with effects:

```
R> Effect.rlmerMod <- function(focal.predictors, mod, ...){
+      args <- list(
+        coefficients = lme4::fixef(mod))
+      Effect.default(focal.predictors, mod, ..., sources=args)
+    }
```

4 Beta Regression

The betareg function in the betareg package [Grün et al., 2012] fits regressions with a link function but with Beta distributed errors.

```
R> Effect.betareg <- function(focal.predictors, mod, ...){
+      coef <- mod$coefficients$mean
+      vco <- vcov(mod)[1:length(coef), 1:length(coef)]
+      args <- list(
+        call = mod$call,
+        formula = formula(gy_logit),
+        coefficients = coef,
+        link = mod$link$mean,
+        vcov = vco)
+      Effect.default(focal.predictors, mod, ..., sources=args)
+    }
```

The Beta distributions require estimation of the parameters of the Beta, but these are not used by effects. The relevant coefficients and covariance matrix are extracted by the first two lines of the function. This method has a link function specified by the link argument, but no family, so the link argument is added to sources.c

5 Ordinal Models

Proportional odds logit and probit regression models fit with the polr function in the MASS package [Venables and Ripley, 2002] are supported in the effects package. The ordinal package, [Christensen, 2015] contains three functions that are very similar to polr. The clm and clm2 functions allow more link functions and a number of other generalizations. The clmm function allows including random effects.

5.1 clm

```
R> Effect.clm <- function(focal.predictors, mod, ...){
+      if (requireNamespace("MASS", quietly=TRUE)){
+        polr <- MASS::polr}
+      if(mod$link != "logit") stop("Effects only supports the logit link")
+      if(mod$threshold != "flexible") stop("Effects only supports the flexible threshold")
+      if(is.null(mod$Hessian)){
+        message("\nRe-fitting to get Hessian\n")
+        mod <- update(mod, Hess=TRUE)}
+       numTheta <- length(mod$Theta)
+       numBeta <- length(mod$beta)
+       or <- c( (numTheta+1):(numTheta + numBeta), 1:(numTheta))
+       args <- list(
+         type = "polr",
+         coefficients = mod$beta,
+         vcov = as.matrix(vcov(mod)[or, or]))
+       Effect.default(focal.predictors, mod, ..., sources=args)
+     }
```

This function first checks that the `MASS` package is available. Since the `clm` function allows suppressing the computation of the Hessian, the function checks and computes it if needed to get the estimated covariance matix. The `clm` function orders the parameters in the order (threshold parameters, linear predictor parameters), so the next few lines identify the elements of `vcov` that are needed by `Effects`. Since the `polr` function does not allow thresholds other thab `flexible`, we don't allow them either. Simiarly, we have only implemented effects for the default `logit` link

5.2 `clm2`

```
R> Effect.clm2 <- function(focal.predictors, mod, ...){
+       if (requireNamespace("MASS", quietly=TRUE)){
+         polr <- MASS::polr}
+       if(is.null(mod$Hessian)){
+         message("\nRe-fitting to get Hessian\n")
+         mod <- update(mod, Hess=TRUE)}
+       if(mod$link != "logistic") stop("Effects only supports the logit link")
+       if(mod$threshold != "flexible") stop("Effects only supports the flexible threshold")
+       numTheta <- length(mod$Theta)
+       numBeta <- length(mod$beta)
+       or <- c( (numTheta+1):(numTheta + numBeta), 1:(numTheta))
+       args <- list(
+         type = "polr",
+         formula = mod$call$location,
+         coefficients = mod$beta,
+         vcov = as.matrix(vcov(mod)[or, or]))
+       Effect.default(focal.predictors, mod, ..., sources=args)
+     }
```

The syntax for `clm2` is not the same as `clm`, so a separate method is required

5.3 `clmm`

This function allows for random effects in an ordinal model.

```
R> Effect.clmm <- function(focal.predictors, mod, ...){
+       if (requireNamespace("MASS", quietly=TRUE)){
+         polr <- MASS::polr}
+       if(is.null(mod$Hessian)){
+         message("\nRe-fitting to get Hessian\n")
+         mod <- update(mod, Hess=TRUE)}
+       if(mod$link != "logit") stop("Only the logistic link is supported by Effects")
+       if(mod$threshold != "flexible") stop("Only threshold='flexible supported by Effects")
+       numTheta <- length(mod$Theta)
+       numBeta <- length(mod$beta)
+       or <- c( (numTheta+1):(numTheta + numBeta), 1:(numTheta))
+      skip <- length(unique(model.frame(mod)[,1])) - 1
+      vcov <- matrix(NA, nrow=numBeta + skip, ncol=numBeta + skip)
+      sel <- rownames(vcov(mod)) %in% names(mod$beta)
+      vcov[1:numBeta, 1:numBeta] <- vcov(mod)[sel, sel]
+      args <- list(
+        type = "polr",
+        formula = fixFormula(as.formula(mod$formula)),
+        coefficients = mod$beta,
+        vcov = as.matrix(vcov))
+      Effect.default(focal.predictors, mod, ..., sources=args)
+    }
```

Complications here come from getting the right elements of `vcov(mod)` corresponding to the fixed effects.

6 Multinomial Models

6.1 `multinom`

6.2 Latent Class Models

References

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

R. H. B. Christensen. `ordinal`—Regression Models for Ordinal Data, 2015. URL `http://www.cran.r-project.org/package=ordinal/`. `R` package version 2015.6-28.

B. Grün, I. Kosmidis, and A. Zeileis. Extended beta regression in R: Shaken, stirred, mixed, and partitioned. *Journal of Statistical Software*, 48(11):1–25, 2012. URL `http://www.jstatsoft.org/v48/i11/`.

M. Koller. robustlmm: An R package for robust estimation of linear mixed-effects models. *Journal of Statistical Software*, 75(6):1–24, 2016. doi: 10.18637/jss.v075.i06.

D. A. Linzer and J. B. Lewis. `poLCA`: An ° package for polytomous variable latent class analysis. *Journal of Statistical Software*, 42(10):1–29, 2011.

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. `R` package version 2.2.

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and `R` Core Team. `nlme`: Linear and Nonlinear Mixed Effects Models, 2016. URL `http://CRAN.R-project.org/package=nlme`. `R` package version 3.1-127.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with* `S`. Springer-Verlag, New York, 4th edition, 2002.