Defining Effect Methods for Other Models

John Fox and Sanford Weisberg

June 11, 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 fitting methods. The mechanism for this is fairly simple. This vignette assumes you are familiar with R's S3 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, but as illustrated in later sections you may need to modify some of the arguments that are sent to `Effect.default`. .

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 a new `Effect` needs to be written; the package will take care of all the other functions.

1 Using `effects` with Other Modeling Methods, with Generalized Least Squares in the `nlme` package as an Example

The `gls` function in the `nlme` package (Pinheiro et al., 2018) fits linear models via generalized least squares. A call to `gls` creates an object of class `gls`. The

following function for `gls` objects is included in the `effects` package.

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

This function sets an argument `sources` that is then passed to the function `Effect.default`. The arguments `focal.predictors` and `mod` match the first two arguments of `Effect.default`, and the `...` matches all other arguments. The value of `sources`, a list of up to six named vaues, is set in the body of the function:

`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 `type = "polr"` is used for ordinal regression models, as in the `polr` function in the `MASS` package, and similar methods described below in Section 6. The The `type = "multinom"` for multinomial log-linear models as fit by the `multinom` function in `nnet`, and to polytomous latent class models created with the `poLCA` function in the `poLCA` package.

`call` The `Effect.default` method uses the call to harvest additional arguments that it needs. For `type="glm"`, these arguments are `formula`, `data`, `contrasts`, `subset`, `family`, and `offset`, although only the `formula` argument is required. 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 the value of this argument should be the value of the formula for fixed effects.

`family` This argument is **required** for GLM-like models that include a `family` that specifies both an error distribution and a link function. The specification `family=family(mod)` is usually appropriate. See the `betareg` example in Section 5 below for an example that includes a user-selected link function, but a fixed error distribution.

`coefficients` In many cases the (fixed-effect) coefficient estimates are returned by `coef(mod)`, the default, but if this is not the case then the value of this

argument should be the estimates of the coefficients in the linear predictor. The functions in the `effects` package do not use estimates of random effects.

`vcov` In many cases the estimated covariance matrix of the (fixed-effect) coefficient estimates is returned by `vcov(mod)`, the default, but if this is not the case then the value of this argument should be the estimated covariance matrix of the (fixed-effect) 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 `Effect.gls` method, as the default method would work.

```
 library(effects)
```

```
Loading required package: carData
```

```
lattice theme set by effectsTheme()
See ?effectsTheme for details.
```

```
 g <- nlme::ngls(Employed ~ GNP + Population,
             correlation=corAR1(form= ~ Year), data=longley)
```

```
Error: 'ngls' is not an exported object from 'namespace:nlme'
```

```
 plot(predictorEffects(g))
```

```
Error in predictorEffects(g): object 'g' not found
```

2 Mixed Effects with `lme` (`nlme` package)

The `lme` function in the `nlme` package (Pinheiro et al., 2018) fits linear mixed models. The required function for fitted objects from this function to be used with `effects` functions is

```
Effects.lme <- function(focal.predictors, mod, ...){
     args <- list(
       formula = mod$call$fixed,
       coefficients = mod$coefficients$fixed,
       vcov = mod$varFix)
     Effect.default(focal.predictors, mod, ..., sources=args)
   }
```

The `formula`, `coefficients` and `vcov` arguments are set to non-default values. The other arguments are automatically set to default values.

```
data(Orthodont, package="nlme")
m1 <- nlme::lme(distance ~ age + Sex, data=Orthodont,
                    random= ~ 1 | Subject)
as.data.frame(Effect("age", m1))
```

```
   age      fit        se    lower    upper
1  8.0 22.04259 0.3657805 21.31732 22.76787
2  9.5 23.03287 0.2632240 22.51095 23.55479
3 11.0 24.02315 0.2185957 23.58971 24.45658
4 12.0 24.68333 0.2394595 24.20853 25.15814
5 14.0 26.00370 0.3657805 25.27843 26.72898
```

3 Mixed Effects with the `lmer` (`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 `lmer` and `glmer` models.

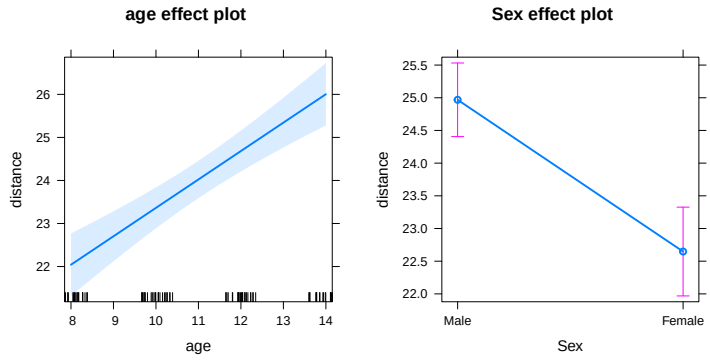
The following method is a little more complicated because it contains an additional argument `KR` to determine if the Kenward-Roger coefficient covariance matrix is to be used to compute effect standard errors. The default is `FALSE` because the computation is very slow. If `KR = TRUE`, the function also checks if the `pbkrtest` package is present.

```
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),
       family=fam,
       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)
   }
```

Because `lmer` is an S4 object, the default for `call` is `mod@call`, and this argument would have been set automatically 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 linear predictors for both the mean function and the variance functions, specified by, for example `(1 + age | Subject)`. The `effects` code will automatically remove any terms like these in any formula, as the effects package only displays the mean function.

```
fm2 <- lme4::lmer(distance ~ age + Sex + (1 |Subject), data
                      = Orthodont)
plot(allEffects(fm2))
```




```
data(cbpp, package="lme4")
gm1 <- lme4::glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
                   data = cbpp, family = binomial)
as.data.frame(predictorEffect("period", gm1))
```

```
  period        fit        se      lower      upper
1      1 0.19807921 0.1535960 0.15454555 0.25024608
2      2 0.08391784 0.2291554 0.05523162 0.12552359
3      3 0.07401714 0.2593423 0.04587560 0.11729928
4      4 0.04842565 0.3031773 0.02732349 0.08441086
```

4 Robust Linear Mixed Models (`robustlmm` package)

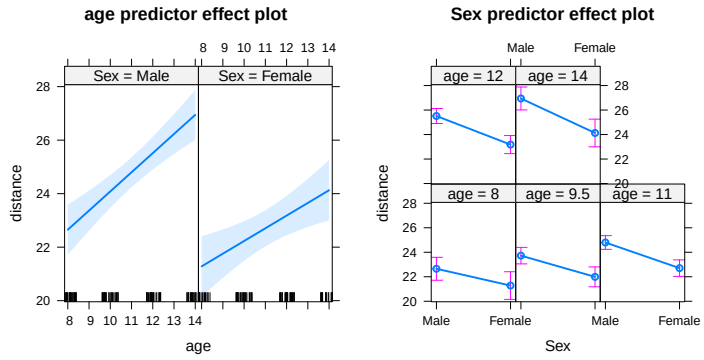
The `rlmer` function in the `robustlmm` package (Koller, 2016) fits linear mixed models with a robust estimation method. As `rlmer` closely parallels the `lmer` function, an object created by `rlmer` is easily used with `effects`:

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

require(lme4)
```

```
Loading required package: lme4
```

```
Loading required package: Matrix
```

```
fm3 <- robustlmm::rlmer(distance ~ age * Sex + (1 |Subject),
                        data = Orthodont)
plot(predictorEffects(fm3))
```



5 Beta Regression

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

```
Effect.betareg <- function(focal.predictors, mod, ...){
    coef <- mod$coefficients$mean
    vco <- vcov(mod)[1:length(coef), 1:length(coef)]
  # betareg uses beta errors with mean link given in mod$link$mean.
  # Construct a family based on the binomial() family
    fam <- binomial(link=mod$link$mean)
  # adjust the varince function to account for beta variance
    fam$variance <- function(mu){
      f0 <- function(mu, eta) (1-mu)*mu/(1+eta)
      do.call("f0", list(mu, mod$coefficient$precision))}
  # adjust initialize
    fam$initialize <- expression({mustart <- y})
    args <- list(
      call = mod$call,
      formula = formula(mod),
      family=fam,
      coefficients = coef,
      vcov = vco)
    Effect.default(focal.predictors, mod, ..., sources=args)
  }
```

Beta regression has a response $y \in [0, 1]$, with the connection between the mean μ of the Beta and a set for predictors $\mathbf{x}$ through a link function $\mathbf{x}'\boldsymbol{\beta} = g(\mu)$. The variance function for the beta is $\text{var}(y) = \mu(1-\mu)/(1+\phi)$, for a precision parameter ϕ estimated by `betareg`.

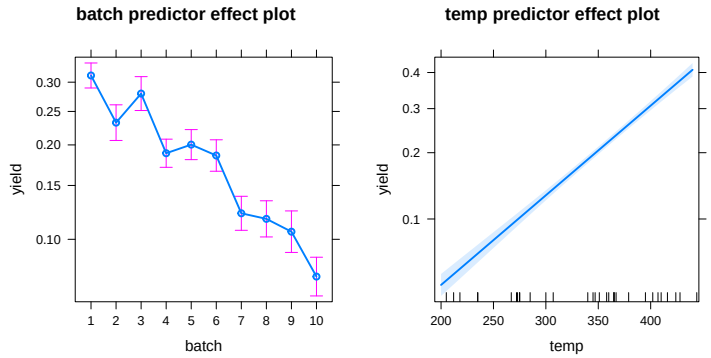
The call to `betareg` does not have a family argument, although it does have a link stored in `mod$link$mean`. For use with `Effect.default`, the function above creates a family from the binomial family generator. It then adjusts

this family by changing from binomial variance to the variance for the beta distribtuion. Since the `glm` function expects a variance that is a function of only one parameter, we fix the value of the precision ϕ at its estimator from the `betareg` fit, as shown in the funciton. We need to replace the `initialize` function to one appropriate for $y \in [0, 1]$. Finally, although the `aic` function is not used for computing effects, it is accessed by the call to `glm`. The `aic` function for the binomial depends on named parameters not present in the beta regression, and so we substitute a dummy function for binomial version.

```
library(betareg)
require(lme4)
data("GasolineYield", package = "betareg")
gy_logit <- betareg(yield ~ batch + temp, data = GasolineYield)
summary(gy_logit)
```

```
Call:
betareg(formula = yield ~ batch + temp, data = GasolineYield)

Standardized weighted residuals 2:
    Min      1Q  Median      3Q     Max
-2.8750 -0.8149  0.1601  0.8384  2.0483

Coefficients (mean model with logit link):
             Estimate Std. Error z value    Pr(>|z|)
(Intercept) -6.1595710  0.1823247 -33.784     < 2e-16
batch1       1.7277289  0.1012294  17.067     < 2e-16
batch2       1.3225969  0.1179020  11.218     < 2e-16
batch3       1.5723099  0.1161045  13.542     < 2e-16
batch4       1.0597141  0.1023598  10.353     < 2e-16
batch5       1.1337518  0.1035232  10.952     < 2e-16
batch6       1.0401618  0.1060365   9.809     < 2e-16
batch7       0.5436922  0.1091275   4.982 0.000000629
batch8       0.4959007  0.1089257   4.553 0.000005297
batch9       0.3857930  0.1185933   3.253     0.00114
temp         0.0109669  0.0004126  26.577     < 2e-16

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value  Pr(>|z|)
(phi)    440.3      110.0   4.002 0.0000629

Type of estimator: ML (maximum likelihood)
Log-likelihood:  84.8 on 12 Df
Pseudo R-squared: 0.9617
Number of iterations: 51 (BFGS) + 3 (Fisher scoring)
```

```
plot(predictorEffects(gy_logit))
```



6 Ordinal Models (`ordinal` package)

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.

6.1 `clm`

```
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 than `flexible`, we don't allow them either. Simiarly, we have only implemented effects for the default `logit` link.

```
 require(ordinal)
```

```
Loading required package: ordinal
```

```


Attaching package: 'ordinal'
```

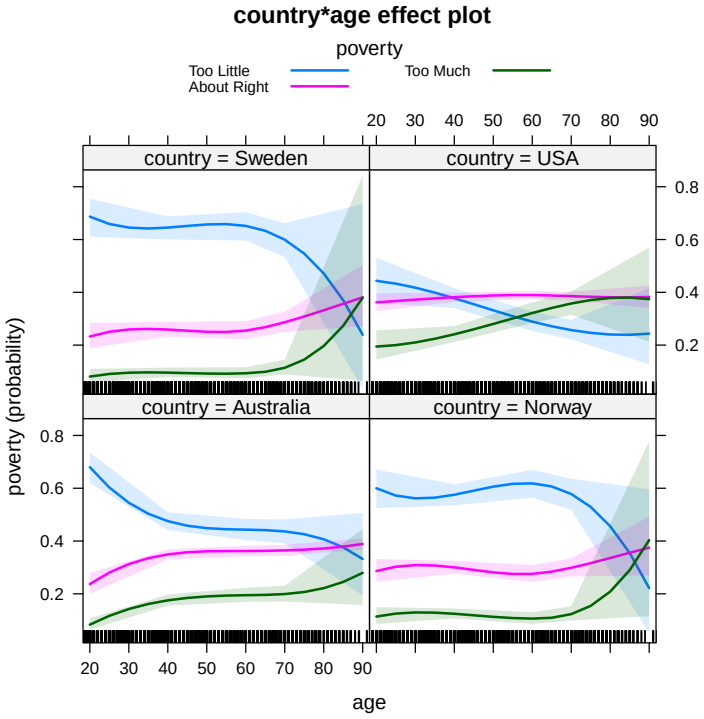
```
The following objects are masked from 'package:lme4':

    ranef, VarCorr
```

```
 require(MASS)
```

```
Loading required package: MASS
```

```
 mod.wvs1 <- clm(poverty ~ gender + religion + degree + country*poly(age,3),
        data=WVS)
 plot(Effect(c("country", "age"), mod.wvs1), lines=list(multiline=TRUE),
        layout=c(2, 2))
```

```


Re-fitting to get Hessian
```



6.2 clm2

```
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.

```
require(ordinal)
require(MASS)
v2 <- clm2(poverty ~ gender + religion + degree + country*poly(age,3),data=WVS)
as.data.frame(emod2 <- Effect(c("country", "age"), v2))
```

```
Re-fitting to get Hessian
```

	country	age	prob.Too.Little	prob.About.Right	prob.Too.Much
1	Australia	20	0.6797882	0.2370610	0.08315076
2	Norway	20	0.6000281	0.2862294	0.11374242
3	Sweden	20	0.6867429	0.2325238	0.08073335
4	USA	20	0.4436803	0.3618540	0.19446567
5	Australia	40	0.4753481	0.3493924	0.17525947
6	Norway	40	0.5753972	0.3002015	0.12440129
7	Sweden	40	0.6452012	0.2590605	0.09573831
8	USA	40	0.3775611	0.3815067	0.24093214
9	Australia	60	0.4431620	0.3620431	0.19479490
10	Norway	60	0.6183860	0.2754172	0.10619675
11	Sweden	60	0.6517064	0.2549975	0.09329615
12	USA	60	0.2889516	0.3895747	0.32147365
13	Australia	70	0.4364803	0.3644358	0.19908392
14	Norway	70	0.5781701	0.2986607	0.12316922
15	Sweden	70	0.5993890	0.2865999	0.11401108
16	USA	70	0.2569656	0.3854097	0.35762470
17	Australia	90	0.3317561	0.3888028	0.27944107
18	Norway	90	0.2214379	0.3748887	0.40367338
19	Sweden	90	0.2391184	0.3809809	0.37990066
20	USA	90	0.2435327	0.3822299	0.37423743

	logit.Too.Little	logit.About.Right	logit.Too.Much	se.prob.Too.Little
1	0.75279879	-1.1688606	-2.4002877	0.03000831
2	0.40558233	-0.9137678	-2.0530712	0.03754335
3	0.78493557	-1.1941150	-2.4324244	0.03685908
4	-0.22623867	-0.5673263	-1.4212502	0.04403803
5	-0.09868762	-0.6217109	-1.5488013	0.01729005
6	0.30390645	-0.8463385	-1.9513953	0.02050097
7	0.59801138	-1.0508575	-2.2455003	0.02243889
8	-0.49991301	-0.4831577	-1.1475759	0.01941098
9	-0.22833907	-0.5665074	-1.4191498	0.02068837
10	0.48270331	-0.9673088	-2.1301922	0.02697660
11	0.62654816	-1.0721339	-2.2740370	0.02756881
12	-0.90048116	-0.4491002	-0.7470077	0.01900352
13	-0.25545917	-0.5561627	-1.3920297	0.02300907
14	0.31526596	-0.8536837	-1.9627548	0.03031121
15	0.40291990	-0.9119551	-2.0504088	0.03301763

```
16      -1.06180028        -0.4666488     -0.5856886         0.01900136
17      -0.70025311        -0.4523473     -0.9472358         0.08216666
18      -1.25730642        -0.5113005     -0.3901825         0.14475615
19      -1.15751898        -0.4853867     -0.4899699         0.20194848
20      -1.13340831        -0.4800941     -0.5140806         0.07473028
   se.prob.About.Right se.prob.Too.Much se.logit.Too.Little
1          0.019789279      0.010823709          0.13785755
2          0.022240558      0.016092827          0.15643429
3          0.024364813      0.012965957          0.17133631
4          0.017491911      0.028210516          0.17841578
5          0.009683414      0.010742983          0.06932874
6          0.012373982      0.009744940          0.08391194
7          0.014475054      0.008961325          0.09802212
8          0.008763096      0.015507751          0.08259686
9          0.010191233      0.013707812          0.08383682
10         0.016677219      0.011279555          0.11431503
11         0.017700323      0.010654437          0.12145641
12         0.008106919      0.020151684          0.09249319
13         0.010677048      0.015393875          0.09354602
14         0.017580053      0.013831406          0.12428260
15         0.019661210      0.014255316          0.13750366
16         0.008737091      0.022653530          0.09951783
17         0.010942087      0.074710963          0.37063098
18         0.057891452      0.202130189          0.83963741
19         0.060083290      0.261516998          1.10996816
20         0.021675725      0.094925384          0.40564789
   se.logit.About.Right se.logit.Too.Much L.prob.Too.Little
1            0.10941580        0.14197502        0.61836234
2            0.10886111        0.15964302        0.52472401
3            0.13653085        0.17470691        0.61043060
4            0.07575021        0.18008769        0.35987029
5            0.04259866        0.07432348        0.44162506
6            0.05890113        0.08946418        0.53480414
7            0.07541127        0.10351239        0.60010234
8            0.03713816        0.08479563        0.34033541
9            0.04412402        0.08739449        0.40307803
10           0.08356887        0.11883348        0.56430440
11           0.09317245        0.12595091        0.59592106
12           0.03409043        0.09238453        0.25317225
13           0.04609679        0.09654388        0.39202551
14           0.08392937        0.12807027        0.51791147
15           0.09616146        0.14112420        0.53330503
16           0.03688574        0.09860968        0.22151634
17           0.04604574        0.37104330        0.19361706
18           0.24703287        0.83968593        0.05200716
19           0.25476891        1.11011662        0.03445573
```

```
20           0.09179565        0.40534569        0.12692074
   L.prob.About.Right L.prob.Too.Much U.prob.Too.Little U.prob.About.Right
1           0.2004777      0.06425078         0.7355555          0.2779989
2           0.2446908      0.08580507         0.6708844          0.3317227
3           0.1882058      0.05869881         0.7541275          0.2836314
4           0.3283198      0.14501886         0.5308233          0.3967899
5           0.3306597      0.15518855         0.5092975          0.3686020
6           0.2765214      0.10652469         0.6149981          0.3249985
7           0.2317135      0.07955704         0.6878595          0.2884233
8           0.3644857      0.21185630         0.4162890          0.3988238
9           0.3423163      0.16932202         0.4840000          0.3822460
10          0.2439580      0.08603001         0.6696841          0.3092737
11          0.2218792      0.07440619         0.7036205          0.2912091
12          0.3738076      0.28331480         0.3275696          0.4055761
13          0.3437763      0.17061821         0.4819797          0.3856073
14          0.2653816      0.09852092         0.6361877          0.3342144
15          0.2496597      0.08891081         0.6620453          0.3266267
16          0.3684334      0.31454407         0.2959312          0.4026696
17          0.3675841      0.15782948         0.5065422          0.4104515
18          0.2698318      0.11548114         0.5958861          0.4932192
19          0.2719553      0.06502436         0.7345787          0.5034877
20          0.3407373      0.21273067         0.4162096          0.4255140
   U.prob.Too.Much L.logit.Too.Little L.logit.About.Right L.logit.Too.Much
1        0.1069748         0.48260295          -1.3833116       -2.6785536
2        0.1492905         0.09897676          -1.1271317       -2.3659658
3        0.1100721         0.44912257          -1.4617105       -2.7748437
4        0.2557294        -0.57592718          -0.7157940       -1.7742156
5        0.1973199        -0.23456944          -0.7052028       -1.6944726
6        0.1447917         0.13944207          -0.9617826       -2.1267419
7        0.1148002         0.40589156          -1.1986608       -2.4483808
8        0.2726181        -0.66179989          -0.5559472       -1.3137722
9        0.2230708        -0.39265622          -0.6529888       -1.5904399
10       0.1304163         0.25864998          -1.1311008       -2.3631015
11       0.1163788         0.38849797          -1.2547485       -2.5208963
12       0.3621748        -1.08176448          -0.5159162       -0.9280781
13       0.2309762        -0.43880600          -0.6465107       -1.5812522
14       0.1529379         0.07167654          -1.0181823       -2.2137680
15       0.1450691         0.13341767          -1.1004281       -2.3270071
16       0.4031356        -1.25685165          -0.5389435       -0.7789600
17       0.4452197        -1.42667648          -0.5425953       -1.6744673
18       0.7782629        -2.90296550          -0.9954761       -2.0359366
19       0.8436735        -3.33301659          -0.9847246       -2.6657585
20       0.5696400        -1.92846356          -0.6600102       -1.3085435
   U.logit.Too.Little U.logit.About.Right U.logit.Too.Much
1          1.02299462         -0.95440955       -2.1220217
2          0.71218790         -0.70040396       -1.7401766
```

```
3          1.12074857        -0.92651945       -2.0900052
4          0.12344983        -0.41885867       -1.0682848
5          0.03719421        -0.53821909       -1.4031299
6          0.46837084        -0.73089441       -1.7760488
7          0.79013121        -0.90305408       -2.0426197
8         -0.33802613        -0.41036829       -0.9813795
9         -0.06402192        -0.48002586       -1.2478598
10         0.70675664        -0.80351686       -1.8972828
11         0.86459835        -0.88951921       -2.0271778
12        -0.71919785        -0.38228420       -0.5659374
13        -0.07211235        -0.46581462       -1.2028072
14         0.55885538        -0.68918520       -1.7117417
15         0.67242212        -0.72348215       -1.7738104
16        -0.86674891        -0.39435409       -0.3924172
17         0.02617026        -0.36209933       -0.2200043
18         0.38835266        -0.02712502        1.2555717
19         1.01797863         0.01395121        1.6858187
20        -0.33835305        -0.30017792        0.2803824
```

6.3 clmm

This function allows for random effects in an ordinal model.

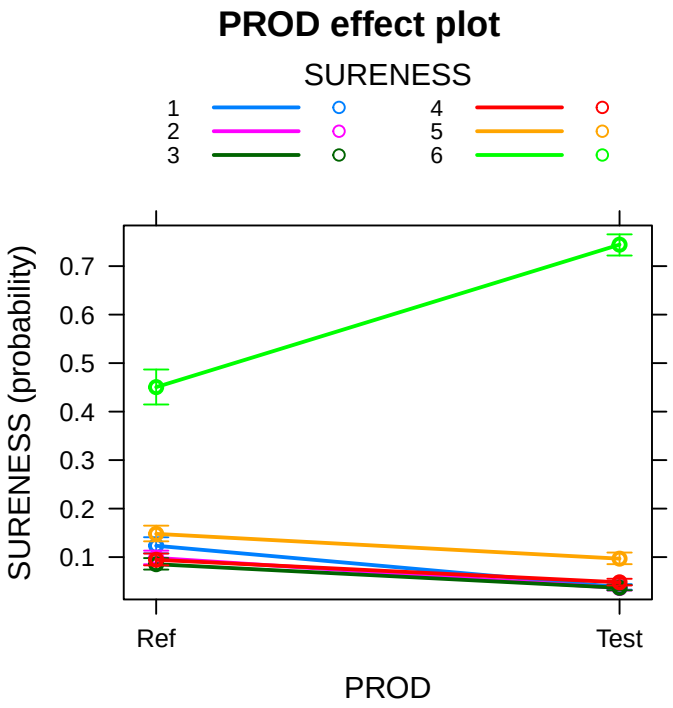
```
Effect.clmm <- function(focal.predictors, mod, ...){
     if (requireNamespace("MASS", quietly=TRUE)){
       polr <- MASS::polr}
     if(is.null(mod$Hessian)){
       message("\nRe-fitting to get Hessian\n")
       mod <- update(mod, Hess=TRUE)}
     if(mod$link != "logit")
       stop("Only the 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.

```
require(ordinal)
require(MASS)
mm1 <- clmm(SURENESS ~ PROD + (1|RESP) + (1|RESP:PROD), data = soup,
            link = "logit", threshold = "flexible")
plot(Effect("PROD", mm1),multiline=TRUE)
```

```
Re-fitting to get Hessian
```



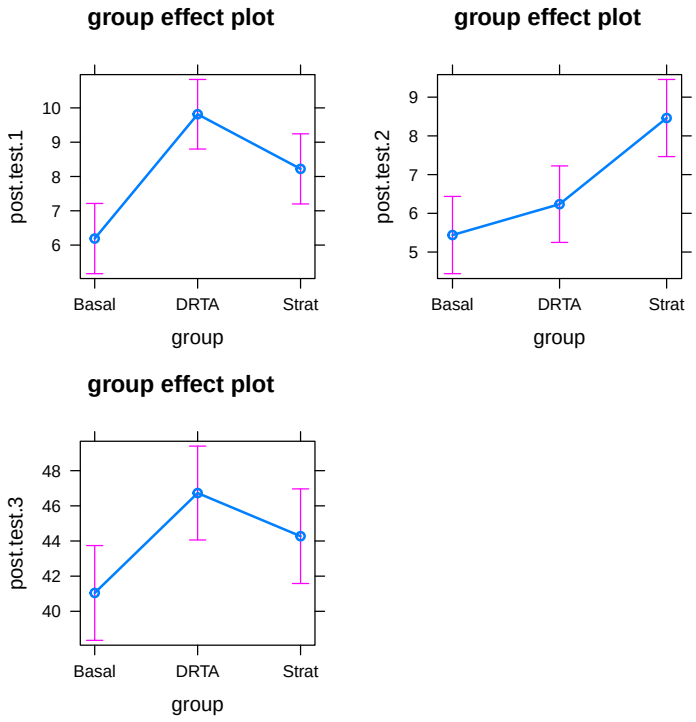
6.4 Others

The `poLCA` function in the `poLCA` package (Linzer and Lewis, 2011) fits polytomous variable latent class models, which uses the multinomial effects plots.

The `svyglm` function in the `survey` package (Lumley, 2004, 2016) fits generalized linear models using survey weights.

The `lm` function can also be used to create a multivariate linear model. The `Effect.mlm` function, with slightly different syntax, will drow effects plots for these models, with separate plots of each response.

```
data(Baumann, package="carData")
b1 <- lm(cbind(post.test.1, post.test.2, post.test.3) ~ group +
       pretest.1 + pretest.2, data = Baumann)
plot(Effect("group", b1))
```



References

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

Christensen, R. H. B. (2015). `ordinal`—Regression Models for Ordinal Data. R package version 2015.6-28.

Grün, B., I. Kosmidis, and A. Zeileis (2012). Extended beta regression in R: Shaken, stirred, mixed, and partitioned. *Journal of Statistical Software 48*(11), 1–25.

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

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

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

Lumley, T. (2016). survey: analysis of complex survey samples. R package version 3.32.

Pinheiro, J., D. Bates, S. DebRoy, D. Sarkar, and R Core Team (2016). `nlme`: Linear and Nonlinear Mixed Effects Models. R package version 3.1-127.

Pinheiro, J., D. Bates, S. DebRoy, D. Sarkar, and R Core Team (2018). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-137.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with* S (4th ed.). New York: Springer-Verlag.