Marginal predictions

January 15, 2021

1 Introduction

In this task, we describe various R machineries for displaying predictions in both simple and complex (involving interaction terms) generalized linear models. We first consider existing approaches for *conditioning* predictions and then describe our proposed approach for *marginalized* predictions. These provide a unified and intuitive way of describing relationships from a fitted model, especially complex models involving interaction terms or some kind transformations on the dependent variables whose estimates are usually, but not always, a subject to less clarity of interpretation.
We can derive various quantities from a fitted regression model. The first and most obvious, is the model coefficient estimates. Others include predicted values of the outcome variable—1) predictions at a particular; and 2) mean or median value of the predictors. The first case simply involves evaluation of the fitted model function, say $\hat{f}(X = x)$, at some particular value of x. The second case, chooses the values of the predictor based on its distributional properties.
Most importantly, from these predicted values, we can also generate second class quantities of interest – *conditional* or *marginal* predictions. These quantities describes the change in the predicted value of the dependent variable after changing one independent variable – either a discrete change in the categorical variable(s) or an instantaneous change in continuous variables, while all other variable are held at specified values.
Suppose we are interested in the *conditional* predictions of a particular predictor (hence forth referred as *focal* predictor otherwise *non-focal*), x_f, in the set of predictors. To keep it simple, assume that the model has no interaction terms. Then the idea is to fix the values of *non-focal* predictor(s) at

some typical values – typically determined by averaging in some meaningful way, for example, arithmetic mean and average over the levels of the factors of *non-focal* continuous and categorical predictors, respectively. This is achieved by averaging the columns of *model matrix*, $\mathbf{X}$, except for the column of focal predictor.

Consider a simple linear model with linear predictor $\eta = \mathbf{X}\beta$ and let $g(\mu) = \eta$ be an identity link function (in the case of simple linear model), where μ is the predicted (expected) value of outcome variable y. Let $\hat{\beta}$ be the estimate of β, together with the estimated covariance matrix $V(\hat{\beta})$ of $\hat{\beta}$. Let the entries of $\mathbf{X}^*$ include all conditioned (*non-focal*) and *focal* predictors. The model matrix, $\mathbf{X}^*$, inherits most of its key properties, for example transformation (e.g., scaling) on the predictors and interactions from the model matrix, $\mathbf{X}$. Then the predicted values $\hat{\eta}^* = \mathbf{X}^*\hat{\beta}$ represents the *conditional* predictions of the focal predictor. Alternatively, we can transform these predictions to response scale using $g^{-1}(\hat{\eta}^*)$.

Further, we can compute the standard errors (SEs) of the *conditional* predictions, $\hat{\eta}^*$, for constructing confidence intervals, as the $\text{sqrt}(\text{diag}(\mathbf{X}^*V(\hat{\beta})\mathbf{X}^{*T}))$. When computing *marginal* predictions, the *uncertainty* as a result of *non-focal* predictors are removed. This can be achieved in two ways:

- using variance-covariance matrix, $V(\hat{\beta})$
- using centered model matrix, $\mathbf{X}^*$

1.1 Variance-covariance

The computation of $\hat{\eta}^*$ remains the same as described above. However, to compute SEs $V(\hat{\beta})$ is modified by *zeroing-out* (the variance-covariance of all non-focal predictors are assigned zero) entries of *non-focal* terms in $V(\hat{\beta})$. This approach requires *centering* of the predictors in the model matrix, X. In other words, the fitted model should have *centered* predictors.

1.2 Centered model matrix

Suppose the *non-focal* entries in X^* are computed by some kind of averaging. Consider centered X^*, $X_c^* = (X^* - \bar{X}^*)$. It follows that the *non-focal* entries in X_c^* are all zero. As a result, the SEs are computed as $\text{sqrt}(\text{diag}(\mathbf{X}_c^*V(\hat{\beta})\mathbf{X}_c^{*T}))$, which actually *zeros-out*. More generally, *centering*, $\mathbf{X}_c^* = \mathbf{X}^* - k$ (for example $k = E(\mathbf{X}^*)$) impacts on the estimated value of the intercept and its associated variance. However, the slopes are not affect by this. The implication of this that since *non-focal* terms in $\mathbf{X}_c^*$ are zero, it doesn't matter what their corresponding values are in the variance-covariance matrix. Hence, we can compute *marginal* predictions from non-centered predictors (in other words, fitted models with predictors in their natural scales).

2 Available packages

The following R packages **effects**, **emmeans** and **margins** implement various schemes for constructing *conditional* predictions. However, currently, their ability to compute *marginal* prediction is limited to the use of variance-covariance approach which requires the fitted model to be centered. We propose **varpred** to overcome this limitation.

3 TODOs

1. Models with interactions
2. GLMs
3. LMEs

4 Illustrations

4.1 Simulation

Consider a simple no-interaction terms simulation:

- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ s.t $\epsilon \sim N(0, 1)$ and $\{\beta_0 = -5, \beta_1 = 0.1, \beta_2 = -0.05\}$
  - $x_1 \sim unif(1, 9)$
  - x_2 – two level categorical variable

```
> set.seed(101)
> N <- 100
> x1_min <- 1
> x1_max <- 9
> b0 <- 0.3
> b1 <- 0.1
> b2 <- -0.6
> x2_levels <- factor(c("A", "B"))
> df <- expand.grid(x1u = runif(n=N, min=x1_min, max=x1_max)
+         , x2u = x2_levels
+ )
> X <- model.matrix(~x1u + x2u, df)
> betas <- c(b0, b1, b2)
> df$y <- rnorm(nrow(df), mean= X %*% betas, sd=1)
> df2 <- df
> df <- transform(df
+         , x1c = drop(scale(x1u, scale=FALSE))
+         , x1s = drop(scale(x1u, center=FALSE))
+         , x1sd = drop(scale(x1u))
+ )
> head(df)
```

```
       x1u x2u          y       x1c       x1s       x1sd
1 3.977587   A -0.4524966 -1.222599 0.6940038 -0.5134124
2 1.350599   A  0.1605887 -3.849588 0.2356505 -1.6165773
3 6.677472   A  1.5456482  1.477286 1.1650760  0.6203644
4 6.261523   A -0.4707503  1.061337 1.0925018  0.4456928
5 2.998846   A  1.3489423 -2.201340 0.5232344 -0.9244202
6 3.400439   A -0.4111428 -1.799747 0.5933038 -0.7557773
```

4.2 Model fitting

No scaling or centering covariates

```
> df_temp <- drop(df)         # margins doesn't work with transform
>                            # but we need df to extract attributes :(
> lm_u <- lm(y ~ x1u + x2u, data = df_temp)
```

Centered covariates $(x - \bar{x})$ model

```
> lm_c <- lm(y ~ x1c + x2u, data = df_temp)
```

Scaled covariates (x/sdx) model

```
> lm_s <- lm(y ~ x1s + x2u, data = df_temp)
```

Both scaled and centered covariates

```
> lm_sd <- lm(y ~ x1sd + x2u, data = df_temp)
```

Coefficients estimates

```
> coef_est <- modsummary(fun=coef)
> print(coef_est)
```

```
              lm_c   lm_s  lm_sd   lm_u
(Intercept)  0.808  0.229  0.808  0.229
x1c          0.111  0.638  0.265  0.111
x2uB        -0.535 -0.535 -0.535 -0.535
```

Log likelihoods

```
> ll_est <- modsummary(fun=logLik)
> print(ll_est)
```

```
        lm_c    lm_s   lm_sd    lm_u
[1,] -272.27 -272.27 -272.27 -272.27
```

Variance covariance matrix

```
> vcov_est <- modsummary(fun=function(x)vcov(x), simplify=TRUE
+         , combine="rbind", match_colnames = names(coef(lm_u)))
> print(vcov_est)
```

```
                  (Intercept)    x1u   x2uB model
lm_c.(Intercept)        0.009  0.000 -0.009  lm_c
lm_c.x1c                0.000  0.001  0.000  lm_c
lm_c.x2uB              -0.009  0.000  0.018  lm_c
lm_s.(Intercept)        0.031 -0.024 -0.009  lm_s
lm_s.x1s               -0.024  0.026  0.000  lm_s
lm_s.x2uB              -0.009  0.000  0.018  lm_s
lm_sd.(Intercept)       0.009  0.000 -0.009 lm_sd
lm_sd.x1sd              0.000  0.005  0.000 lm_sd
lm_sd.x2uB             -0.009  0.000  0.018 lm_sd
lm_u.(Intercept)        0.031 -0.004 -0.009  lm_u
lm_u.x1u               -0.004  0.001  0.000  lm_u
lm_u.x2uB              -0.009  0.000  0.018  lm_u
```

4.3 Conditional predictions

The function will be used downstream to *tidy* conditional predictions from each of the methods highlighted above.

```
> condsummary <- function(mod_list, fun, resp = "y", simplify = TRUE
+         , combine = c("cbind", "rbind"), scale_param = list(), fmethod = NULL){
+         focal <- names(mod_list)
+         combine <- match.arg(combine)
+         out <- sapply(focal, function(f){
+                 mod <- mod_list[[f]]
+                 out <- fun(mod, f)
+                 if (inherits(out, c("emmeans", "emmGrid"))) {
+                         out <- as.data.frame(out)
+                         oldn <- c(f, "emmean"
+                                 , grep("\\.CL", colnames(out), value=TRUE)
+                         )
+                         newn <- c("xvar", "fit", "lwr", "upr")
+                         colnames(out)[colnames(out) %in% oldn] <-  newn
+                 } else if (inherits(out, "eff")) {
+                         out <- as.data.frame(out)
+                         oldn <- c(f, "lower", "upper")
+                         newn <- c("xvar", "lwr", "upr")
+                         colnames(out)[colnames(out) %in% oldn] <-  newn
+                 } else if (any(colnames(out) %in% c("xvals", "yvals"))) {
+                         ## Not proper way to handle margins object.
+                         ## Maybe extend cplot to return inheritable object
+                         oldn <- c("xvals", "yvals", "lower", "upper")
+                         newn <- c("xvar", "fit", "lwr", "upr")
+                         colnames(out)[colnames(out) %in% oldn] <-  newn
+                 } else {
+                         out <- data.frame(out)
+                         colnames(out)[colnames(out)%in%f] <- "xvar"
+                 }
+                 if (combine=="rbind"){
+                         out <- cbind.data.frame(out, model = f)
+                         out <- out[, c("xvar", "fit", "lwr", "upr", "model")]
+                 }
+                 pp <- names(scale_param)
+                 if (!is.null(pp)){
+                         if(grepl("s$",f)){
+                                 vv <- grep("s$",f, value=TRUE)
+                                 xsd <- scale_param[[vv]]
+                                 out[,"xvar"] <- out[,"xvar"]*xsd
+                         }
+                         if(grepl("c$",f)){
+                                 vv <- grep("c$",f, value=TRUE)
+                                 xmu <- scale_param[[vv]]
+                                 out[,"xvar"] <- out[,"xvar"] + xmu
+                         }
+                         if(grepl("sd$",f)){
+                                 vv <- grep("sd$",pp,value=TRUE)
+                                 xmu <- scale_param[[vv]][1]
+                                 xsd <- scale_param[[vv]][2]
+                                 out[,"xvar"] <- out[,"xvar"]*xsd + xmu
+
+                         }
+                 }
+                 if (!is.null(fmethod)){
+                         out[,"method"] <- fmethod
+                 }
+                 return(out)
+         }, simplify = FALSE)
+         f <- attr(out, "focal")
+         if (simplify){
+                 out <- do.call(combine, out)
+                 if (combine=="cbind") {
+                         colnames(out) <- mod
+                 } else {
+                         rownames(out) <- NULL
+                 }
+         }
+         attr(out, "response") <- resp
+         return(out)
+ }
```

We start by computing the conditional predictions using each of the methods (**varpred, emmeans, effects, margins**). In the first case, the predictions are displayed on the variable level scale-specific values (original (u), mean centered (c), divided by standard deviation (s), and both mean centered and scaled (sd)). In the second case, the predictions are transformed back to the original scale.

```
> simple_models <- list(x1u = lm_u, x1c = lm_c, x1s = lm_s, x1sd = lm_sd)
```

4.3.1 Continuous predictor

```
> ## varpred
> simple_vpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- varpred(x, f)
+ }, simplify = TRUE, combine = "rbind", fmethod = "varpred")
> ## emmeans
> simple_empred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         spec <- as.formula(paste0("~", f))
+         dd <- emmeans(x, spec=spec, cov.keep=f)
+ }, simplify = TRUE, combine = "rbind", fmethod = "emmeans")
> ## effects
> simple_efpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- Effect(f, x, xlevels=100)
+ }, simplify = TRUE, combine = "rbind", fmethod = "effects")
> ## margins
> simple_margpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- cplot(x, f, what="prediction", n=100, draw=FALSE)
+ }, simplify = TRUE, combine = "rbind", fmethod = "margins")
> ## Combine all the estimates
> simple_pred_all <- do.call("rbind"
+         , list(simple_vpred_all, simple_empred_all, simple_efpred_all, simple_
+ )
```

```
> head(simple_pred_all)
```

```
      xvar        fit        lwr       upr model  method
1 1.081509 0.08197337 -0.1835256 0.3474723   x1u varpred
2 1.160167 0.09072611 -0.1709771 0.3524294   x1u varpred
3 1.238825 0.09947884 -0.1584477 0.3574054   x1u varpred
4 1.317482 0.10823158 -0.1459380 0.3624011   x1u varpred
```

```
5 1.396140 0.11698432 -0.1334490 0.3674176   x1u varpred
6 1.474798 0.12573705 -0.1209816 0.3724557   x1u varpred

> class(simple_pred_all) <- c("jdeffects", "data.frame") # plot.effects
> simple_pred_all_plot <- (plot(simple_pred_all)
+          + facet_wrap(~method)
+          + theme(legend.position="bottom")
+ )
```

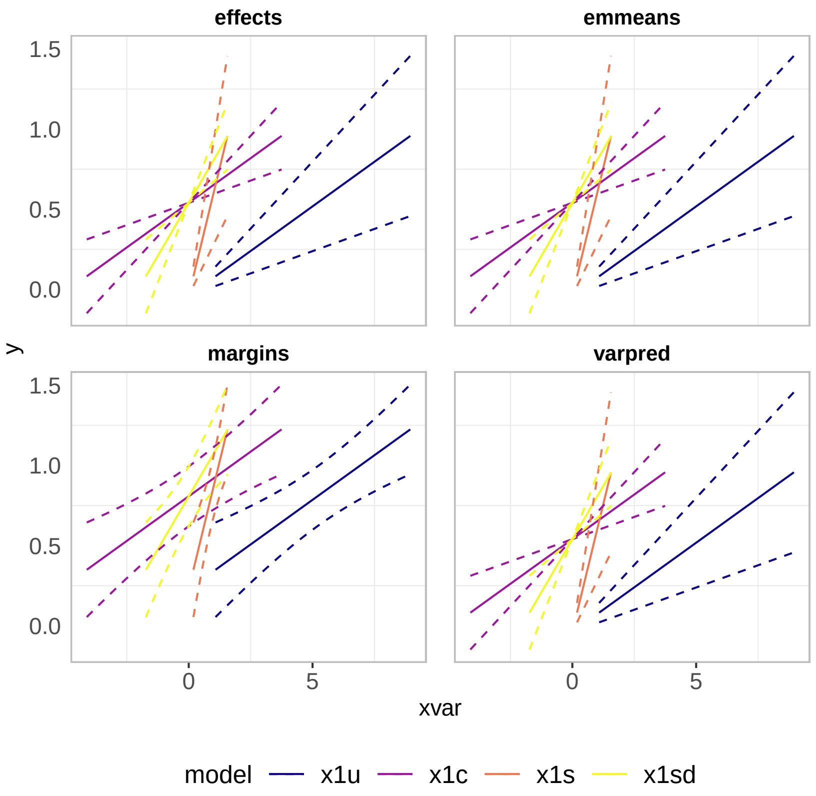


Figure 1: Conditional predictions on the variable-specific values.

Compute predictions and then back transform each predictor to the original scale.

```
> ## Extract scaling parameters
> scale_param <- list(x1s = unlist(attributes(df$x1s))
+         , x1c = unlist(attributes(df$x1c))
+         , x1sd = unlist(attributes(df$x1sd))
+ )
> ## varpred
> simple_vpred_spec <- condsummary(mod_list=simple_models, fun=function(x, f){
+                 dd <- varpred(x, f)
+         }, simplify = TRUE, combine = "rbind"
+         , scale_param = scale_param, fmethod = "varpred"
+ )
> ## emmeans
> simple_empred_spec <- condsummary(mod_list=simple_models, fun=function(x, f){
+                 spec <- as.formula(paste0("~", f))
+                 dd <- emmeans(x, spec=spec, cov.keep=f)
+         }, simplify = TRUE, combine = "rbind"
+         , scale_param = scale_param, fmethod = "emmeans"
+ )
> ## effects
> simple_efpred_spec <- condsummary(mod_list=simple_models, fun=function(x, f){
+                 dd <- Effect(f, x, xlevels=100)
+         }, simplify = TRUE, combine = "rbind"
+         , scale_param = scale_param, fmethod = "effects"
+ )
> ## margins
> simple_margpred_spec <- condsummary(mod_list=simple_models, fun=function(x, f)
+                 dd <- cplot(x, f, what="prediction", n=100, draw=FALSE)
+         }, simplify = TRUE, combine = "rbind"
+         , scale_param = scale_param, fmethod = "margins"
+ )
> ## Combine all the estimates
> simple_pred_spec <- do.call("rbind"
+         , list(simple_vpred_spec, simple_empred_spec, simple_efpred_spec, simp
+ )
```

```
> head(simple_pred_spec)
```

```
      xvar        fit        lwr       upr model  method
1 1.081509 0.08197337 -0.1835256 0.3474723   x1u varpred
```

```
2 1.160167 0.09072611 -0.1709771 0.3524294   x1u varpred
3 1.238825 0.09947884 -0.1584477 0.3574054   x1u varpred
4 1.317482 0.10823158 -0.1459380 0.3624011   x1u varpred
5 1.396140 0.11698432 -0.1334490 0.3674176   x1u varpred
6 1.474798 0.12573705 -0.1209816 0.3724557   x1u varpred

> class(simple_pred_spec) <- c("jdeffects", "data.frame") # plot.effects
> simple_pred_spec_plot <- (plot(simple_pred_spec)
+          + facet_wrap(~method)
+          + theme(legend.position="bottom")
+ )
```

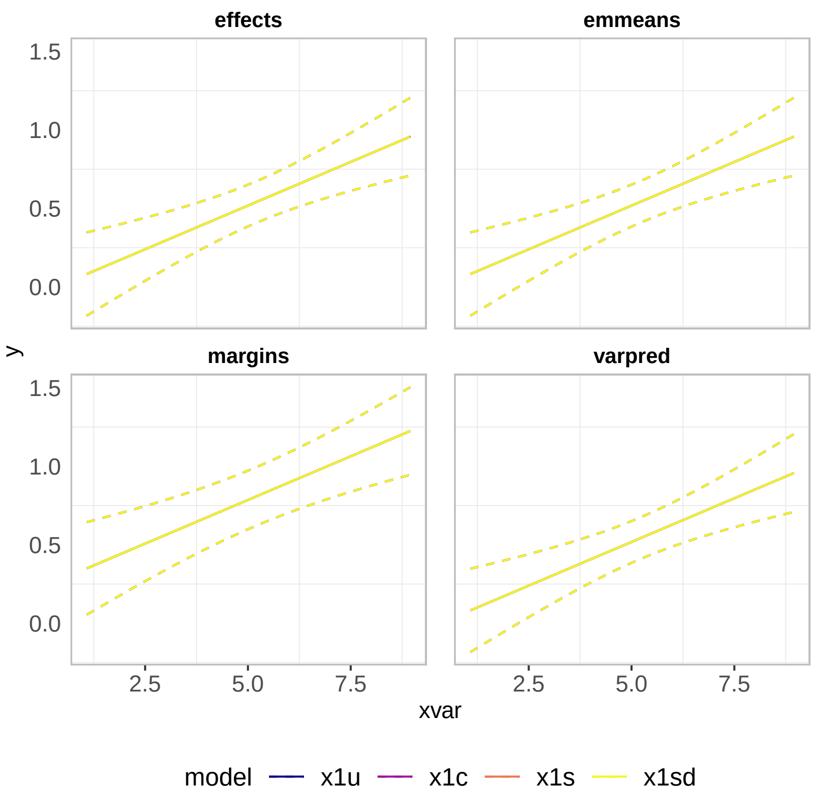


Figure 2: Back-transformed conditional predictions.

4.3.2 Categorical predictors

Here, we are considering case where we are only interested estimating the conditional predictions. In this case, it doesn't matter which of the four models we choose since they will give the same predictions. Later, we'll show how we can estimate marginalized and/or centered (CI) for categorical predictors.

```
> ## varpred
> simple_models_cat <- list(x2u = lm_u)
> simple_vpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f)
+         dd <- varpred(x, f)
+ }, simplify = TRUE, combine = "rbind", fmethod = "varpred")
> ## emmeans
> simple_empred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f
+         spec <- as.formula(paste0("~", f))
+         dd <- emmeans(x, spec=spec, cov.keep=f)
+ }, simplify = TRUE, combine = "rbind", fmethod = "emmeans")
> ## effects
> simple_efpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f
+         dd <- Effect(f, x, xlevels=100)
+ }, simplify = TRUE, combine = "rbind", fmethod = "effects")
> ## margins
> simple_margpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x,
+         dd <- cplot(x, f, what="prediction", n=100, draw=FALSE)
+ }, simplify = TRUE, combine = "rbind", fmethod = "margins")
> ## Combine all the estimates
> simple_pred_cat <- do.call("rbind"
+         , list(simple_vpred_cat, simple_empred_cat, simple_efpred_cat, simple_
+ )
```

```
> head(simple_pred_cat)
```

```
  xvar       fit       lwr       upr model  method
1    A 0.8077896 0.6202052 0.9953740   x2u varpred
2    B 0.2727794 0.0851950 0.4603639   x2u varpred
3    A 0.8077896 0.6202052 0.9953740   x2u emmeans
4    B 0.2727794 0.0851950 0.4603639   x2u emmeans
5    A 0.8077896 0.6202052 0.9953740   x2u effects
6    B 0.2727794 0.0851950 0.4603639   x2u effects
```

```
> class(simple_pred_cat) <- c("jdeffects", "data.frame") # plot.effects
> simple_pred_cat_plot <- (plot(simple_pred_cat)
+         + facet_wrap(~method)
+         + theme(legend.position="bottom")
+ )
```

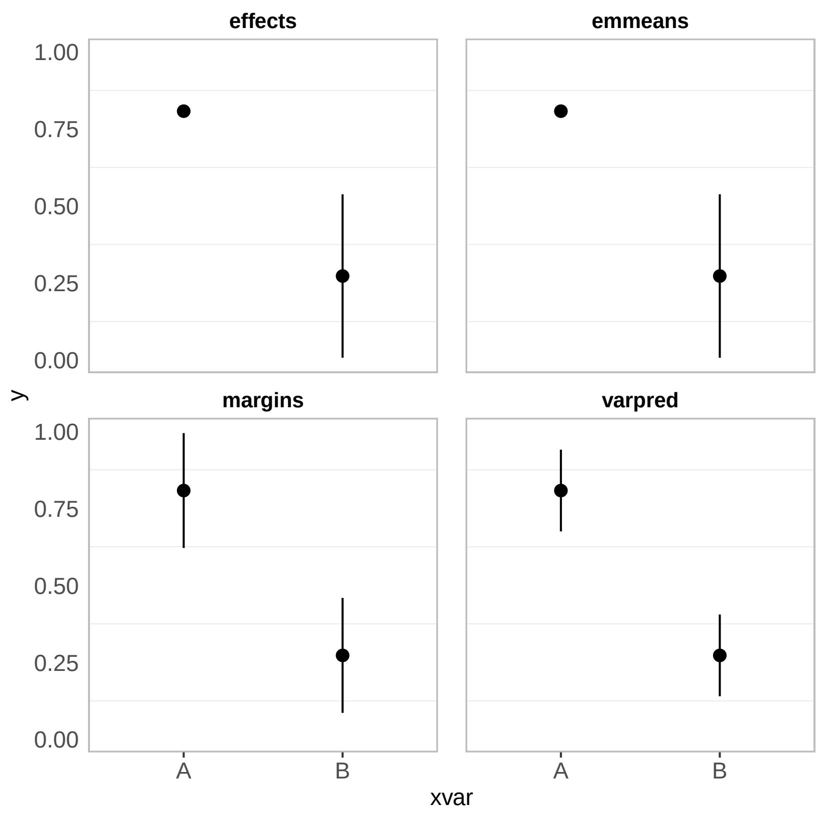


Figure 3: Conditional predictions.

4.4 Marginal predictions

We can implement marginalization by *zeroing-out* the variance of non-focal predictors (possible in all four methods) but this requires that the predictors are centered beforehand. On the other hand, in **varpred** marginal predictions can be obtained by centering the SE estimates.

4.5 zeroed-out covariance matrix

We **zero_vcov** function in **varpred** to transform the variances of non-focal predictors to zero. For example:

```
> lm_u_vcov <- vcov(lm_u)
> print(lm_u_vcov)

             (Intercept)           x1u         x2uB
(Intercept)  0.030729586 -4.169417e-03 -9.047842e-03
x1u         -0.004169417  8.017823e-04 -3.588329e-19
x2uB        -0.009047842 -3.588329e-19  1.809568e-02

> ## x1u is the focal variable
> lm_u_vcov_zero <- zero_vcov(lm_u, focal_vars = "x1u")
> print(lm_u_vcov_zero)

            (Intercept)          x1u x2uB
(Intercept)           0 0.0000000000    0
x1u                   0 0.0008017823    0
x2uB                  0 0.0000000000    0
```

We'll repeat the same procedure in conditional predictions section but modify the functions to incorporate *zeroed-out* covariance matrix.

4.5.1 Continuous predictor

Notice that the modification in each of the method function call to pass either covariance matrix or function. In particular, in **varpred**, it is passed as **vcov.**, **emmeans** and **effects** as **vcov.** and **margins** as **vcov**. However, this seems not to work currently in **cplot** which computes the predictions in **margins**.

```
> ## varpred
> simple_vpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- varpred(x, f, vcov. = zero_vcov(x, f))
+ }, simplify = TRUE, combine = "rbind", fmethod = "varpred")
> ## emmeans
> simple_empred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
```

```
+         spec <- as.formula(paste0("~", f))
+         dd <- emmeans(x, spec=spec, cov.keep=f, vcov. = zero_vcov(x, f))
+ }, simplify = TRUE, combine = "rbind", fmethod = "emmeans")
> ## effects
> simple_efpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- Effect(f, x, xlevels=100, vcov. = function(x, complete=FALSE)zer
+ }, simplify = TRUE, combine = "rbind", fmethod = "effects")
> ## margins
> simple_margpred_all <- condsummary(mod_list=simple_models, fun=function(x, f){
+         dd <- cplot(x, f, what="prediction", n=100, draw=FALSE, vcov=function(
+ }, simplify = TRUE, combine = "rbind", fmethod = "margins")
> ## Combine all the estimates
> simple_pred_all <- do.call("rbind"
+         , list(simple_vpred_all, simple_empred_all, simple_efpred_all, simple_
+ )
```

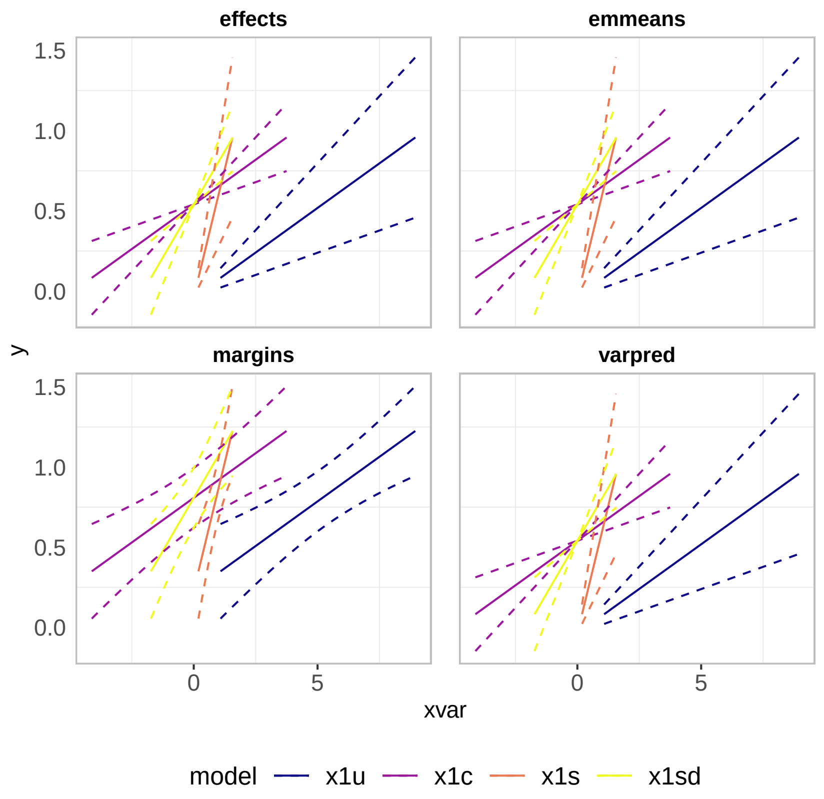
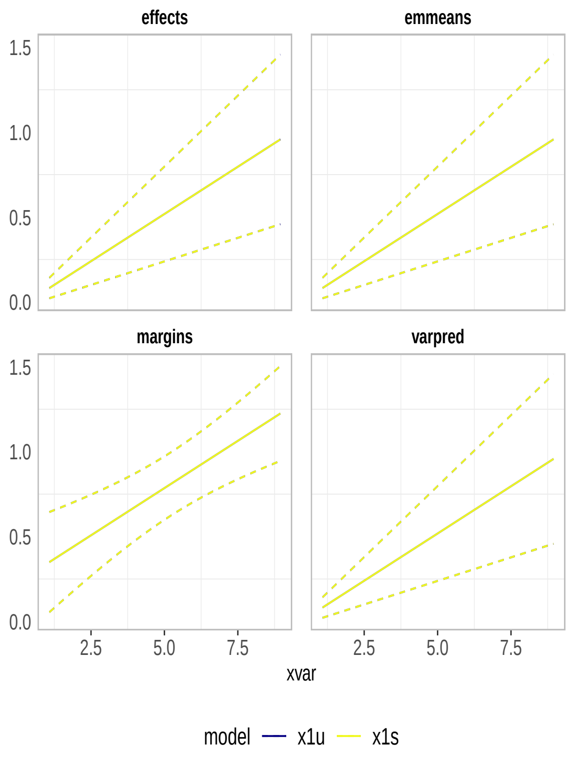


Figure 4: Marginal predictions with the zeroed-out covariances on the variable-specific scale.

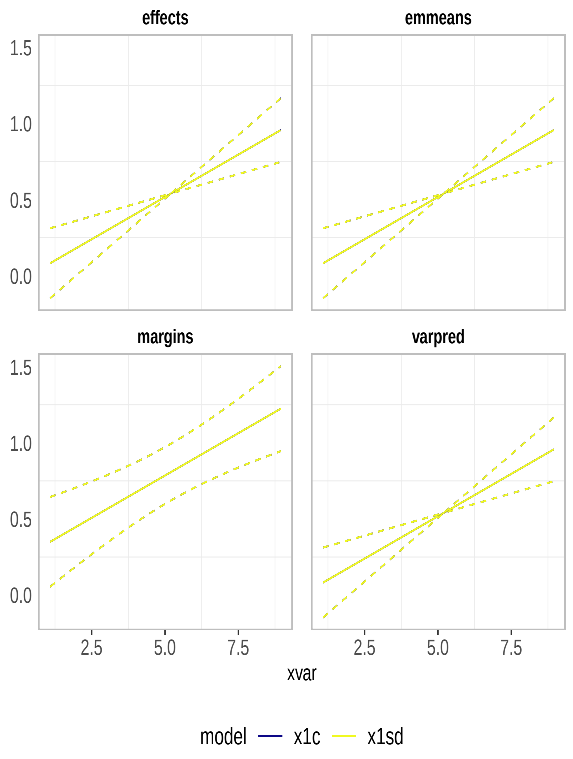
Also, we repeat above implementation but back-transform the predictions to original scale (unscale – uncenter and/or unscale) by modifying the previous code to include scaling parameters. To clearly distinguish between what happens when apply various scaling schemes, we separately do the plots (see Figure 5).

4.6 Centered model matrix

Marginal predictions in **effects** and **emmeans** require models fitted with transformed predictors. However, in **varpred** we can use model fitted using predictors on their original scale and estimate marginal predictions. In our example, we use the unscaled model `lm_u` and simply set `isolate=TRUE`. The predictions (see Figure 6) are similar to Figure 5 b (except for the **margins**).



(a) Either unscaled or devided by the *sd*.



(b) Centered or centered and/or devided by *sd*.

Figure 5: Back-transformed marginal predictions.

```
> ## Centered predictions from unscaled model
> vpred_c <- varpred(lm_u, focal = "x1u", isolate = TRUE)
> plot(vpred_c)
```

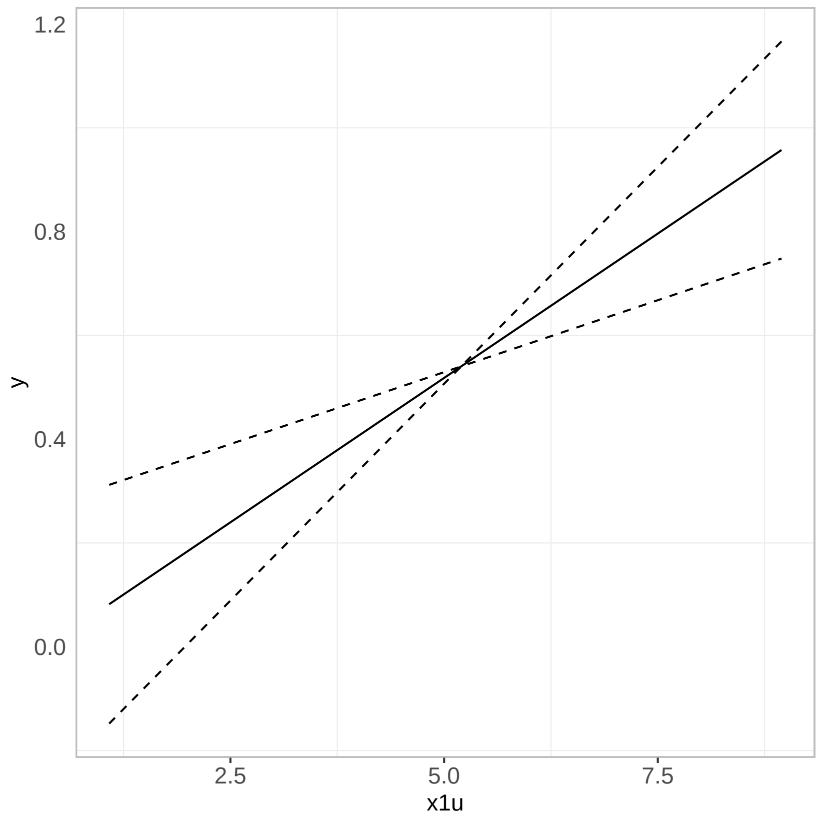


Figure 6: Using **varpred** to obtain marginal predictions from unscaled model.

4.6.1 Categorical predictors

Back to categorical predictor, `x2u`. The reference (base) category has not variance and has such we can not use the *zeroed-out* covariance approach to estimate CI for the marginalized predictions. However, if we can figure out how to center model matrix in the computation of CIs, we can actually use model with unscaled categorical predictors and estimate marginal predictions for centered categorical predictors. Currently, only **varpred** provide this functionality.

```
> ## varpred
> simple_models_cat <- list(x2u = lm_u)
> simple_vpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f)
+         dd <- varpred(x, f, isolate=TRUE)
+ }, simplify = TRUE, combine = "rbind", fmethod = "varpred")
> ## emmeans
> simple_empred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f
+         spec <- as.formula(paste0("~", f))
+         dd <- emmeans(x, spec=spec, cov.keep=f, vcov. = zero_vcov(x, f))
+ }, simplify = TRUE, combine = "rbind", fmethod = "emmeans")
> ## effects
> simple_efpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x, f
+         dd <- Effect(f, x, xlevels=100, vcov. = function(x, complete=FALSE)zer
+ }, simplify = TRUE, combine = "rbind", fmethod = "effects")
> ## margins
> simple_margpred_cat <- condsummary(mod_list=simple_models_cat, fun=function(x,
+         dd <- cplot(x, f, what="prediction", n=100, draw=FALSE, vcov=function(
+ }, simplify = TRUE, combine = "rbind", fmethod = "margins")
> ## Combine all the estimates
> simple_pred_cat <- do.call("rbind"
+         , list(simple_vpred_cat, simple_empred_cat, simple_efpred_cat, simple_
+ )
```

```
> head(simple_pred_cat)
```

```
  xvar       fit         lwr       upr model  method
1    A 0.8077896 0.675147365 0.9404318   x2u varpred
2    B 0.2727794 0.140137215 0.4054217   x2u varpred
3    A 0.8077896 0.807789595 0.8077896   x2u emmeans
4    B 0.2727794 0.007494985 0.5380639   x2u emmeans
5    A 0.8077896 0.807789595 0.8077896   x2u effects
6    B 0.2727794 0.007494985 0.5380639   x2u effects
```

```
> class(simple_pred_cat) <- c("jdeffects", "data.frame")
> simple_pred_cat_plot <- (plot(simple_pred_cat)
+         + facet_wrap(~method)
+         + theme(legend.position="bottom")
+ )
```

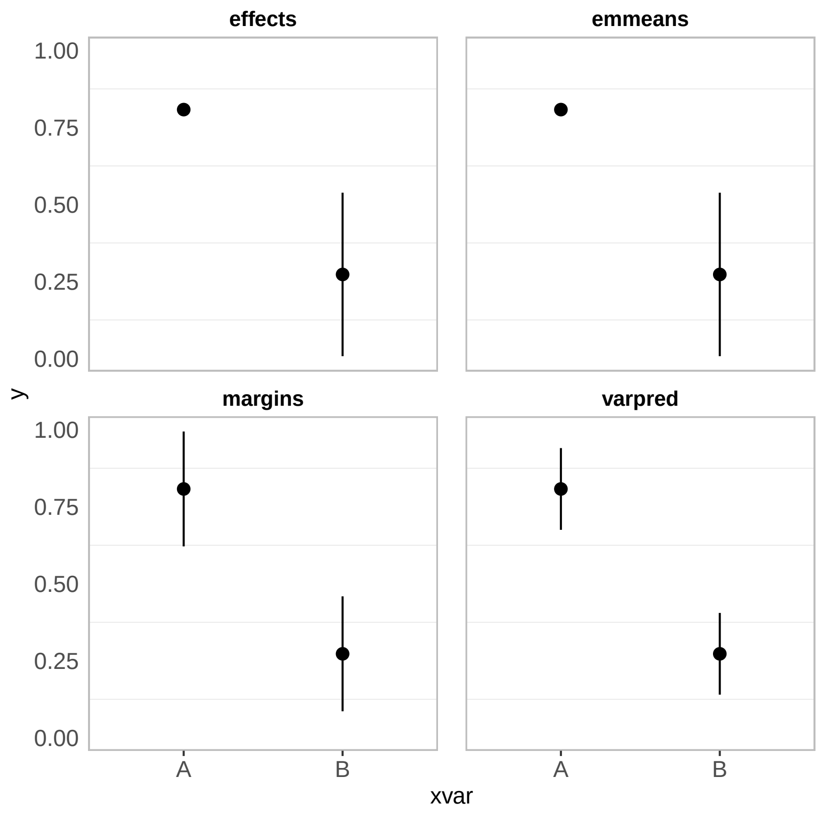


Figure 7: Conditional predictions.