Average Predictive Comparisons for Stanfit objects

Harold Eyster

April 16, 2019

Situation:

Average predictive comparisons are a good way to interpret complex models

Compication:

It's hard to implement this method, and the difference between this method and average marginal effects is unclear.

Question:

Can implementing Average Predictive Comparisons be made accessible and intelligible relative to other methods?

Answer:

Let's implement it in R!

What are Average Predictive Comparisons?

Average Predictive Comparisons can increase interpretability of variables in complex (e.g., hierarchical and interactive) models. Gelman and Pardoe (2007) presented the theoretical argument for their efficacy, but there are no easily accessible implementations. There is an R package (predcomps) developed by David Chudzicki in 2014, but is only suited for a few model types (e.g., lm), doesn't allow categorical inputs, and does not appear to be under any further development.

Average Marginal Effects

Another R package, modmarg, reverse engineers Stata's margins function to create average marginal effects and std errors. See here for an example where this method increases model interpretability. This package only works for `glm` and `ivreg` objects in R. Are average marginal effects and average predictive comparisons the same? Not quite. It is calculated differently. To find the average marginal effect of, say, forest type on biodiversity, first a model is created using all of your data on the diversity found in forest and nonforest patches. Then, all the data is set to forest and the model is used to predict the biodiversity in these plots and the mean is taken; next all the data is set to 'nonforest' and again the model is used to predict biodiversity, and the mean is taken. The difference between these means is the average marginal effect. To estimate the associated standard error, this approach uses the delta method. Let's look at an example to see how this works.

Average marginal effects example

First, let's create some fake data that we'll pretend represent our observations of birds in forest and nonforest plots:

```
set.seed(538)
biodiv.forest<- round(rnorm(50,50,5), 0)
biodiv.nonforest <-round(rnorm(50,40,5), 0)
biodiv<-c(biodiv.forest,biodiv.nonforest)
forest<-c(rep(1,50),rep(0,50))
dat <- data.frame(cbind(biodiv,forest))
```

We can see that we now have observations from 100 plots – 50 forest and 50 nonforest.

```
head(dat)
```

```
##   biodiv forest
## 1     37      1
## 2     58      1
## 3     58      1
## 4     58      1
## 5     40      1
## 6     47      1
```

Now let's create a simple model:

```
mod<-glm(biodiv~forest, data=dat, family=gaussian)
```

Now let's run a average marginal effects analysis: First setting all the data to forest:

```
dat.forest<-data.frame(cbind(biodiv, 'forest'=rep(1,100)))
```

And predicting with the model:

```
pred_forest <- predict(mod, newdata = dat.forest, type = 'response')
```

And taking the average:

```
(avg.forest<-mean(pred_forest))
```

```
## [1] 50.88
```

Now doing the same for nonforest:

```
dat.nonforest<-data.frame(cbind(biodiv, 'forest'=rep(0,100)))
pred_nonforest <- predict(mod, newdata = dat.nonforest, type = 'response')
(avg.nonforest<-mean(pred_nonforest))
```

```
## [1] 40.76
```

We can see that the average marginal effect of forest on biodiversity is

```
avg.forest-avg.nonforest
```

```
## [1] 10.12
```

Now confirming that the `modmarg` package yields the same answer:

```
modmarg::marg(mod, 'forest', type = 'levels')
```

```
## [[1]]
##        Label Margin Standard.Error Test.Stat      P.Value Lower CI (95%)
## 1 forest = 0  40.76      0.7253993  56.18975 2.316974e-76       39.32047
## 2 forest = 1  50.88      0.7253993  70.14068 1.427899e-85       49.44047
##   Upper CI (95%)
## 1       42.19953
## 2       52.31953
```

Which is the same as the 'forest' coefficient from our linear model:

```
summary.glm(mod)$coefficients
```

```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)    40.76  0.7253993 56.189746 2.316974e-76
## forest         10.12  1.0258695  9.864803 2.378663e-16
```

This equivalence is because this is a simple model with no interactions, transformations, or mixed-effects – in more complex models, marginal effects become more useful. To see an example of marginal effects applied to a more complex model, and proof of their usefulness, see here.

Variance

We want to have some idea about how well-constrained these average marginal effects values are. We cannot just average the standard errors from `pred_nonforest` and `pred_forest`, since these do not represent the uncertainty encoded in `mod`. Instead, AME uses the delta method to estimate variance. What is the delta method? Simplistically: given a known sequence of random numbers, $\{\hat{\theta}_n\}$ (the original `dat` in our example), that converge asymptotically to a distribution with known mean θ_0 and known variance V (e.g., `mod`) such that:

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \rightarrow^d N(0, V) \tag{1}$$

Then applying a continuously differentiable function g (the marginal effect) acting on these random numbers, the delta method can be used to find the model parameters corresponding to the new random number sequence:

$$\sqrt{n}(g(\hat{\theta}_n) - g(\theta_0)) \rightarrow^d N\left(0, \left(\frac{dg(\theta_0)}{d\theta}\right)^2 V\right) \tag{2}$$

How do these average marginal effects compare to average predictive comparisons?

Average Predictive Comparisons

Average predictive comparisons are calculated differently. From eq'n (1) of Gelman and Pardoe, a predictive comparison of changing the input of interest, υ, from one value to another is defined as:

$$\delta_u(\upsilon^1 \rightarrow \upsilon^2, \nu, \theta) = \frac{E(y|\upsilon^2, \nu, \theta) - E(y|\upsilon^1, \nu, \theta)}{\upsilon^2 - \upsilon^1} \tag{3}$$

Where θ is the model, y is the response variable, and ν is a vector of all the other variables (i.e., all the variables except υ).

In our example, θ is `mod`, y is `biodiv`, υ is `forest`, and ν is empty, because our example did not have any other predictors. Average predictive comparisons are distinct from partial derivatives, since APCs do not collapse $\upsilon^2 - \upsilon^1$ to zero.

But this equation is just the predictive comparison – to get the *average* predictive comparison, we need to average (in this case, the root-mean-square) across values of θ and ν and changes in υ. For unordered categorical inputs (e.g., `forest` and `nonforest` in our example), the APC is defined as:

$$APC = \widehat{\Delta}\upsilon = \left(\frac{\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{s=1}^{S}[\sum_{j\epsilon(k)}\omega(\upsilon_i, \upsilon_j)]E(y|\upsilon^k, \nu_i, \theta^s) - E(y|\upsilon_i, \nu_i, \theta^s)}{S\sum_{i=1}^{n}\sum_{k=1}^{K}[\sum_{j\epsilon(k)}\omega(\upsilon_i, \upsilon_j)]}\right)^{\frac{1}{2}} \tag{4}$$

Where n is the number of observations in the data (n=50 in our example `dat`), K is the number of categories (K=2 in our example `dat`), S is the number of model draws, and $\omega(\upsilon_i, \upsilon_j)$ is the likelihood that υ transitions from υ_i to υ_j when $\nu = \nu_i$. For example, let's say that for every habitat patch (i) we not only had data on

bird diversity and forest cover, but also data on how many chainsaws were operating in each patch. Our dataset would not be balanced – there would be no chainsaws operating in the nonforest areas. Thus, when $\nu_i = nonforest$, the likelihood of a transition from $\upsilon_i = 5chainsaws$ to $\upsilon_i = 6chainsaws$ would be zero. This addition of a weighting element is one way that APC differs from AME. AME assumes that all variables are balanced and uncorrelated, whereas APC accounts for any correlation between input variables.

Variance

Variance is the second way that APC differs from AME. Instead of of using the Delta Method, APCs extract the uncertainty directly from the uncertainty in the model, θ. In a Bayesian context, this model uncertainty is obtained from draws, s, from the posterior distribution. In a Frequentist context, these could be obtained by e.g., bootstrapping. This makes APCs especially amenable to Bayesian models, and obviates the computation complexity from using the delta method. This difference in assessing uncertainty is the primary distinction between AME and APC. To see clearly how the APC standard error results directly from the uncertainty in the model, see this definition of APC standard error for unorderd categorical inputs:

$$S.E.(\widehat{\Delta_\upsilon}) = \frac{1}{2\widehat{\Delta_\upsilon}} \left(\frac{1}{S-1} \sum_{s=1}^{S} [(\widehat{\Delta_\upsilon^s})^2 - (\widehat{\Delta_\upsilon})^2]^2 \right)^{\frac{1}{2}} \tag{5}$$

Differences between APC and AME:

To recap, the two primary differences between AME and APC are: 1. One assumption that AMEs make is that they assume that all input variables independent and uncorrelated. APC, on the other hand, weights each set of variable values according to their frequency in the data. 2. They calculate uncertainty around the point estimates differently – AME uses the Delta Method, whereas APC uses the variability in the model draws. This makes APC well-suited for Bayesian models and more complex models.

A real example

Given the potential for APCs to make complex Bayesian models more interpretable, I will apply this method to a model that describes real data on growth time of invasive plants.

The model

The model includes three unordered categorical fixed effects: stratification length (`strat` – 2 levels), germination temperature (`temp` – 4 levels), seed origin (`origin` – 2 levels). These fixed effects have a full suite of interaction terms. The model also includes three nested random slopes and intercepts: seed family (`sfamily`) nested within sampling location (`loc`) nested within species (`sp`). The output variable is growth rate. The data come from a balanced experiment, where the treatment variables were uncorrelated. The model was fitted with $n = 1119$ observations using Stan and rstanarm.

Our goal will be to calculate the Average Predictive Comparison for $\upsilon =$`origin`. First, let's load the model:

```
load("C:/Users/Owner/Documents/Thesis/Stan/mod_gr.Rdata")
mod<-mod_gr
```

and the data:

```
load("C:/Users/Owner/Documents/github/germination_stan/datax.Rdata")
dat<-datax
```

Let's see what the data structure looks like:

```
head(dat)
```

```
##      N          y temp1 temp2 temp3 origin strat nsp sp loc sfamily
## 1 1119 0.08508549     1     0     0      0     0   7  5  10      68
## 2 1119 0.07896257     1     0     0      0     0   7  4  13      30
## 3 1119 0.14475508     1     0     0      0     0   7  4   9      37
## 4 1119 0.13855679     1     0     0      0     0   7  2   5       6
## 5 1119 0.10021950     1     0     0      0     0   7  6   5      72
## 6 1119 0.09450092     1     0     0      0     0   7  6   5      73
```

and add some libraries necessary to process stanfit objects:

```
require(rstan)
require(rstanarm)
```

Because the data come from a balanced experiment (i.e., every combination of input values is equally likely to co-occur), we can ignore the weighting element of the APC in equation (4). Additionally, because `origin` is a binary variable, we do not have to sum across multiple values of k.Thus, equation 4 becomes:

$$APC = \widehat{\Delta}\upsilon = \left(\frac{\sum_{i=1}^{n} \sum_{s=1}^{S} (E(y|\upsilon = 1, \nu_i, \theta^s) - E(y|\upsilon = 0, \nu_i, \theta^s))^2}{Sn} \right)^{\frac{1}{2}} \tag{6}$$

Our dataset has 1119 observations – we'll use this full set of ν values, since the vectorized code below is quite efficient. We'll also use a sample of 1000 posterior draws (S=1000). The code below uses the model to calculate the expected growth rate for each value of ν and θ, for given values of υ(`origin` = 1 and 0). Then, following equation 6, we take the squared difference and sum over ν and θ.

```
S<-1000
n<-nrow(dat) #1119
newdat<-dat[,c(3:7,9:11)] #subsetting just the observation data, nu and upsilon
newdat1<-newdat
newdat1$origin<-rep(1,n)
newdat0<-newdat
newdat0$origin<-rep(0,n)
E_u1<-posterior_predict(mod,newdata=newdat1,draws=S,seed=248) #each of these columns
#represents the expected value for a different value of nu; each row represents the
#expected value according to a different model draw; each of these is for origin=1
E_u0<-posterior_predict(mod,newdata=newdat0,draws=S,seed=248) # Now the same, but for origin=0
E_diff<-((E_u1-E_u0)^2) #the squared difference, as in equation 6.
sum_theta<-colSums(E_diff) #summing across model draws
sum_nu_theta<-sum(sum_theta) #summing across nu
num<-sum_nu_theta #this is the numerator in equation 6
(num)
```

```
## [1] 2128.031
```

`num` is the numerator of the fraction in eq'n (4). For the denominator:

```
denom<-S*n
(denom)
```

```
## [1] 1119000
```

Finally, the Average Predictive Comparison for `origin` is:

```
APC<-(num/denom)^(1/2)
(APC)
```

```
## [1] 0.04360878
```

Now to calculate the standard error, as in equation 5.

```
sum_nu<-rowSums(E_diff)/n
```

sum_nu represents $(\widehat{\Delta}_v^s)^2$ in equation 5, with each value representing a different draw of s.

```
apc_vec<-rep(APC^2,S) # a vector of apc
SE<-(1/(2*APC))*(sqrt(1/(S-1)*(sum  ((sum_nu-apc_vec)^2)  )))
(SE)
```

```
## [1] 0.006843574
```

Now we'll wrap the above code into a function, for future calculations:

```
apc<-function(mod,dat){
  S<-1000
  n<-nrow(dat) #1119
  newdat<-dat[,c(3:7,9:11)] #subsetting just the observation data, nu and upsilon
  newdat1<-newdat
  newdat1$origin<-rep(1,n)
  newdat0<-newdat
  newdat0$origin<-rep(0,n)
  E_u1<-posterior_predict(mod,newdata=newdat1,draws=S,seed=248) #each of these columns
    #represents the expected value for a different value of nu; each row represents the
    #expected value according to a different model draw; each of these is for origin=1
  E_u0<-posterior_predict(mod,newdata=newdat0,draws=S,seed=248) # Now the same, but for origin=0
  E_diff<-((E_u1-E_u0)^2) #the squared difference, as in equation 6.
  sum_theta<-colSums(E_diff) #summing across model draws
  sum_nu_theta<-sum(sum_theta) #summing across nu
  num<-sum_nu_theta #this is the numerator in equation 6
  denom<-S*n #this is the denominator in equation 6.
  APC<-(num/denom)^(1/2) #this is the average predictive comparison

  #now calculating the stadard error
  sum_nu<-rowSums(E_diff)/n
  apc_vec<-rep(APC^2,S) # a vector of apc
  SE<-(1/(2*APC))*(sqrt(1/(S-1)*(sum  ((sum_nu-apc_vec)^2)  ))) #The standard error
  return(paste('APC =',APC, ', SE = ',SE))
}
```

And testing it:

```
apc(mod_gr, datax)
```

```
## [1] "APC = 0.0436087796931172 , SE =  0.00684357362050058"
```

What is the APC telling us? It reports that, on average, the growth rate of a plant with a European origin will differ from that with an American origin by 0.04 cm/day. The average growth rate for all plants was 0.12 cm/day. Thus, this effect of origin is quite important.

Validation

How do the APC values compare to our model coefficients?

```
as.data.frame(summary(mod_gr,pars=c("origin")))
```

```
##               mean         mcse         sd       2.5%          25%
## origin 0.006406923 0.0002860658 0.01389071 -0.0212469 -0.002376574
##                 50%        75%      97.5% n_eff     Rhat
```

```
## origin 0.006575513 0.01541256 0.03295983  2358 1.000324
```

The model coefficient, 0.006575513, is much less than the APC. Why is this? It's because we are working with a model with interactions and random effects. The APC takes these all into a account, and thus reports a more holistic effect of `origin`. But how do we know that our calculation is correct and makes sense? There are no accepted ways for calculating apc for such a complex model, but we can validate our method on a simpler one.

For a model with no interactions and no random effects, we expect that the APC = coefficient. Thus, let's create and test this simpler model:

```
mod2<-stan_glm(y~origin+strat+temp1+temp2+temp3+sp+loc+sfamily,data=dat,
               algorithm = "sampling", prior=normal(), prior_intercept=normal(0,10),
               prior_aux=cauchy(0,5), chains=4, iter=1000,refresh=0)

apc(mod2,datax) #apc
```

```
## [1] "APC = 0.0735284220758959 , SE =  0.00999134461755107"
```

```
as.data.frame(summary(mod2,pars=c("origin"))) #model coefficient
```

```
##               mean         mcse          sd       2.5%        25%
## origin 0.07300443 0.0002673612 0.009960859 0.05222484 0.06670741
##                50%        75%      97.5% n_eff     Rhat
## origin 0.07307511 0.07978814 0.09132614  1388 1.003894
```

Here we see good alignment between our APC and the model coefficient, as predicted. Thus, our method seems to be valid.

We can also verify that the Average Marginal Effect is the same:

```
mod3<-glm(y~origin+strat+temp1+temp2+temp3+sp+loc+sfamily,data=dat) #the `marg' function doesn't work o
marg(mod3,'origin', type='effects')
```

```
## [[1]]
##        Label     Margin Standard.Error Test.Stat      P.Value
## 1 origin = 0 0.00000000     0.00000000       NaN          NaN
## 2 origin = 1 0.07345474     0.01012519  7.254653 7.544243e-13
##   Lower CI (95%) Upper CI (95%)
## 1     0.00000000     0.00000000
## 2     0.05358807     0.09332141
```

Conclusion

Average Predictive Comparisons offer a powerful method for interpreting complex models. They offer 2 benfits over Average Marginal Effects: 1) they account for the correlation of input variables and 2) offer a simpler standard error calculation. Due to their capacity to directly utilize model uncertainty, they are especiallly well-suited for Bayesian models. Here, I have demonstrated their applicability to a hierachical mixed-effect model, and shown the utility of the result. APCs should become more widelly used to interpret complex Bayesian models. I hope that this illustration can serve as a guide for future attempts.