Introduction to the predictnl function

Mark Clements
Karolinska Institutet

Abstract

The `predictnl` generic function supports variance estimation for non-linear estimators using the delta method with finite differences for the partial derivatives. The function loosely extends the `predict` generic function.

Keywords: delta method,variance estimation.

1. Introduction

The delta method provides a general approach to calculate the variance (or standard errors) for a number of likelihood-based estimators. If we have an estimated vector of parameters $\hat{\theta}$ with covariance matrix $\hat{\mathbf{\Sigma}}$ and a vector prediction function $g(\theta)$, then the variance can be calculated using the delta method:

$$\text{var}(g(\hat{\theta})) = \left(\frac{\partial g(\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}} \right)^T \hat{\mathbf{\Sigma}} \left(\frac{\partial g(\theta)}{\partial \theta}\bigg|_{\theta=\hat{\theta}} \right)$$

Classical statistics tends to focus on analytical calculations for the partial derivatives. As an example, let $g(\theta_i) = \exp(\theta_i)$ for the ith parameter; then $\partial g(\theta_i)/\partial(\theta_i) = \exp(\theta_i)$ and $\partial g(\theta_i)/\partial(\theta_j) = 0$ for $i \neq j$. We then have that $\text{var}(g(\hat{\theta}_i)) = \text{diag}\left(\exp(\hat{\theta}_i)\right)^T \hat{\mathbf{\Sigma}}\ \text{diag}\left(\exp(\hat{\theta})\right)$. The partials can be calculated symbolically:

```
> thetahat <- c(0.1,0.2)
> Sigma <- matrix(c(1,0.3,0.3,1),2,2)
> print(partial <- D(expression(exp(theta)),"theta"))
```

```
exp(theta)
```

```
> partial <- diag(eval(partial,list(theta=thetahat)))
> var <- t(partial) %*% Sigma %*% partial
> data.frame(fit=exp(thetahat), se.fit=sqrt(diag(var)))
```

```
       fit   se.fit
1 1.105171 1.105171
2 1.221403 1.221403
```

Note that we have calculated the standard errors by taking the square root of the diagonal of the matrix. We can also calculate the partials using finite differences, such that for a continuous function f

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}$$

In R:

```
> myD <- function(f,x,eps=1e-5) (f(x+eps)-f(x-eps))/(2*eps)
> partial <- diag(myD(exp,thetahat))
> var <- t(partial) %*% Sigma %*% partial
> data.frame(fit=exp(thetahat),se.fit=sqrt(diag(var)))

       fit   se.fit
1 1.105171 1.105171
2 1.221403 1.221403
```

This gives the same estimates to six decimal places. We could also calculate these values using the `predictnl` command, which we now introduce.

The `predictnl` generic function provides a generalisation of the `predict` generic function to calculate standard errors. `predictnl` uses finite differences (or a user-supplied matrix) to calculate the partial derivatives. The basic call is `predictnl(object, fun, newdata=NULL, ...)` where `fun` takes the `object` as the first argument; `newdata` is an optional argument to `fun`, and other arguments `...` are passed to `fun`. The `object` is required to have three methods: (i) `vcov.class` to extract the variance-covariance matrix of the estimated parameters; (ii) `coef.class` to extract the coefficients; and (iii) `coef<-.class` to set the coefficients. For the example above, we use a `test` class and define the three methods; note that the `coef` and `coef<-` methods are as per the default. We then define an object (named `fit`) of that class, define a prediction function `expcoef` which exponentiates the coefficients, and then call `predictnl`:

```
> library(rstpm2)
> vcov.test <- function(object) object$vcov
> coef.test <- function(object) object$coefficients # default
> "coef<-.test" <- function(object,value)
+   { object$coefficients <- value; object} # default
> fit <- structure(list(vcov=Sigma,coefficients=thetahat),class="test")
> expcoef <- function(object) exp(coef(object))
> predictnl(fit,expcoef)

       fit   se.fit
1 1.105171 1.105171
2 1.221403 1.221403
```

This gives the same output as before. Generic methods for `vcov` and `coef` are available for most regression models; a default implementations for `coef<-` is provided in the `rstpm2` package.

```
> rstpm2:::"coef<-.default"

function (x, value)
{
    x$coefficients <- value
    x
}
<bytecode: 0x5564cd06ccc8>
<environment: namespace:rstpm2>
```

For another simple example, consider a linear regression with a single covariate, such that $E(y) = \beta_0 + \beta_1 x_1$ for standard normal distributed x_1 with homoscedastic standard normal errors. For a given outcome y, we want to find the value of x that solves $y = \hat{\beta}_0 + \hat{\beta}_1 x$.

```
> set.seed(123456)
> x1 <- rnorm(1000)
> y <- rnorm(1000,x1)
> fit <- lm(y~x1)
> invert <- function(object,newdata) {
+     thetahat <- coef(object)
+     (newdata$y-thetahat[1])/thetahat[2]
+ }
> predictnl(fit, invert, newdata=data.frame(y=seq(0,2,by=0.5)))

          fit     se.fit
1 0.007973223 0.03208338
2 0.505825537 0.03585955
3 1.003677852 0.04540490
4 1.501530166 0.05793476
5 1.999382481 0.07190551
```

We can also calculate average values across the sample under counterfactual exposures. We extend the previous example to include a second Bernoulli-distributed covariate x_2 with $p = 0.5$, where $E(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. We may be interested in calculating the marginal estimators $E(y|x_2 = 0)$, $E(y|x_2 = 1)$ and $E(y|x_2 = 1) - E(y|x_2 = 0)$, where the expectations are across the full dataset[1]. Then:

```
> set.seed(123456)
> x1 <- rnorm(1000)
> x2 <- rbinom(1000,1,0.5)
> y <- rnorm(1000,x1+x2)
> fit <- lm(y~x1+x2)
> standardise <- function(object,newdata) mean(predict(object,newdata))
> predictnl(fit, standardise, newdata=data.frame(x1,x2=0))
```

[1]Under suitable causal assumptions, we could re-write the three estimators as $E(y|\mathrm{do}(x_2) = 0)$, $E(y|\mathrm{do}(x_2) = 1)$ and $E(y|\mathrm{do}(x_2) = 1) - E(y|\mathrm{do}(x_2) = 0)$.

```
        fit     se.fit
1 0.0752667 0.04430177

> predictnl(fit, standardise, newdata=data.frame(x1,x2=1))

        fit     se.fit
1 0.9659855 0.04547089

> standdiff <- function(object,newdata)
+     standardise(object,transform(newdata,x2=1))-
+         standardise(object,transform(newdata,x2=0))
> predictnl(fit, standdiff, newdata=data.frame(x1))

        fit     se.fit
1 0.8907188 0.06353538
```

As a final example, we consider modelling for additive interaction contrasts using `rstpm2`. Consider a time-to-event variable T with two binary exposures x_1 and x_2. Adapting the formulations from Rothman et al (Modern Epidemiology, third edition, 2008) to time-dependent outcomes, the interaction contrast $\mathrm{IC}(t)$ and relative excess risk for interaction $\mathrm{RERI}(t)$ can be calculated by

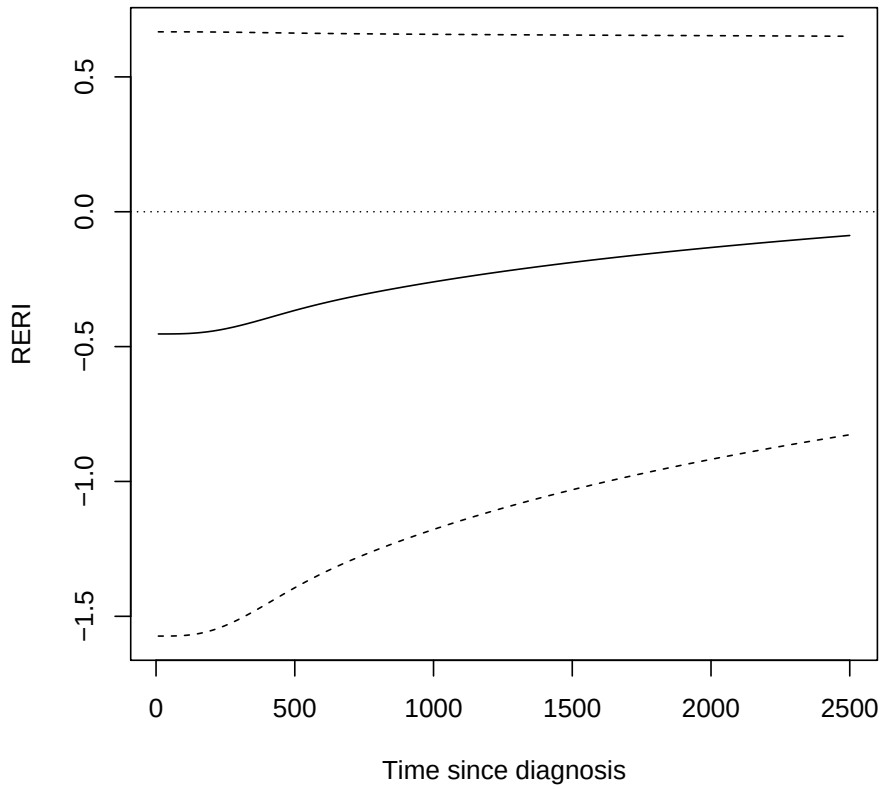
$$\begin{aligned}
\mathrm{IC}(t) &= \Pr(T \le t|x_1=1,x_2=1) - \Pr(T \le t|x_1=1,x_2=0) - \Pr(T \le t|x_1=0,x_2=1) + \Pr(T \le t|x_1=0,x_2=0) \\
&= -(S(t|x_1=1,x_2=1) - S(t|x_1=1,x_2=0) - S(t|x_1=0,x_2=1) + S(t|x_1=0,x_2=0)) \\
\mathrm{RERI}(t) &= \frac{\Pr(T \le t|x_1=1,x_2=1) - \Pr(T \le t|x_1=1,x_2=0) - \Pr(T \le t|x_1=0,x_2=1) + \Pr(T \le t|x_1=0,x_2=0)}{\Pr(T \le t|x_1=0,x_2=0)} \\
&= -\frac{S(t|x_1=1,x_2=1) - S(t|x_1=1,x_2=0) - S(t|x_1=0,x_2=1) + S(t|x_1=0,x_2=0)}{1 - S(t|x_1=0,x_2=0)}
\end{aligned}$$

where $S(t|x_1,x_2)$ is the survival function.

As an empirical example, we can use the `brcancer` dataset with the randomised hormonal therapy (variable `hormon`, encoded 0 and 1) and stage (1 vs 2–3; variable `x4.23` encoded `TRUE` and `FALSE`). The research question is whether there is a more than additive interaction between therapy and stage.

```
> brcancer2 <- transform(brcancer, x4.23=x4 %in% 2:3)
> fit1 <- stpm2(Surv(rectime,censrec==1)~hormon*x4.23,data=brcancer2,df=3)
> newd <- data.frame(hormon=0,x4.23=FALSE)
> RERI <- function(object, newdata,
+                  var1, val1=1,
+                  var2, val2=1) {
+     exp1 <- function(data) {data[[var1]] <- val1; data}
+     exp2 <- function(data) {data[[var2]] <- val2; data}
+     s11 <- predict(object, newdata=exp1(exp2(newdata)), type="surv")
+     s10 <- predict(object, newdata=exp1(newdata), type="surv")
+     s01 <- predict(object, newdata=exp2(newdata), type="surv")
+     s00 <- predict(object, newdata, type="surv")
+     -(s11-s10-s01+s00)/(1-s00)
+ }
```

```
> times <- seq(0,2500,length=301)[-1]
> reri <- predictnl(fit1,fun=RERI,newdata=transform(newd,rectime=times),
+                   var1="hormon",var2="x4.23",val2=TRUE)
> with(reri, matplot(times,fit+cbind(0,-1.96*se.fit,+1.96*se.fit),
+                    type="l",lty=c(1,2,2),col=1,
+                     xlab="Time since diagnosis", ylab="RERI"))
> abline(h=0,lty=3)
>
```



Affiliation:

Mark Clements
Department of Medical Epidemiology and Biostatistics
Karolinska Institutet
Email: mark.clements@ki.se