TRANSFORMED REGRESSION PARAMETERS IN R USING THE DELTA METHOD? | R FAQ

The purpose of this page is to introduce estimation of standard errors using the delta method. Examples include manual calculation of standard errors via the delta method and then confirmation using the function **`deltamethod`** so that the reader may understand the calculations and know how to use **`deltamethod`**.

This page uses the following packages Make sure that you can load them before trying to run the examples on this page. We will need the **`msm`** package to use the **`deltamethod`**function. If you do not have a package installed, run: **`install.packages("packagename")`**, or if you see the version is out of date, run: **`update.packages()`**.

```
library(msm)
```

`Version info: Code for this page was tested in R version 3.1.1 (2014-07-10) On: 2014-08-01 With: pequod 0.0-3; msm 1.4; phia 0.1-5; effects 3.0-0; colorspace 1.2-4; RColorBrewer 1.0-5; xtable 1.7-3; car 2.0-20; foreign 0.8-61; Hmisc 3.14-4; Formula 1.1-2; survival 2.37-7; lattice 0.20-29; mgcv 1.8-1; nlme 3.1-117; png 0.1-7; gridExtra 0.9.1; reshape2 1.4; ggplot2 1.0.0; vcd 1.3-1; rjson 0.2.14; RSQLite 0.11.4; DBI 0.2-7; knitr 1.6`

Background to delta method

Background to delta method

Often in addition to reporting parameters fit by a model, we need to report some transformation of these parameters. The transformation can generate the point estimates of our desired values, but the standard errors of these point estimates are not so easily calculated. They can, however, be well approximated using the delta method. The delta method approximates the standard errors of transformations of random variable using a first-order Taylor approximation. Regression coefficients are themselves random variables, so we can use the delta method to approximate the standard errors of their transformations. Although the delta method is often appropriate to use with large samples, this page is by no means an endorsement of the use of the delta method over other methods to estimate standard errors, such as bootstrapping.

Essentially, the delta method involves calculating the variance of the Taylor series approximation of a function. We, thus, first get the Taylor series approximation of the function using the first two terms of the Taylor expansion of the transformation function about the mean of of the random variable. Let G be the transformation function and μ_X be the mean vector of random variables (X=(x1,x2,...)). The first two terms of the Taylor expansion are then an approximation for $G(X)$,

$$G(X) \approx G(\mu_X) + \nabla G(\mu_X)^T (X - \mu_X)$$

where $\nabla G(\mu_X)$ is the gradient of $G(X)$ at $X = \mu_X$, or a vector of partial derivatives of $G(X)$ at point μ_X. We can then take the variance of this approximation to estimate the variance of $G(X)$ and thus the standard error of a transformed parameter. For a random variable X with known variance-covariance matrix $Cov(X)$, the variance of the transformation of X, $G(X)$ is approximated by:

$$Var(G(X)) \approx \nabla G(\mu_X)^T Cov(X) \nabla G(\mu_X)$$

.

Example 1: Adjusted prediction

Adjusted predictions, or adjusted means, are predicted values of the response calculated at a set of covariate values. For example, we can get the predicted value of an "average" respondent by calculating the predicted value at the mean of all covariates. Adjusted predictions are functions of the regression coefficients, so we can use the delta method to approximate their standard errors.

We would like to calculate the standard error of the adjusted prediction of **y** at the mean of **x, 5.5,** from the linear regression of **y** on **x**:

```
x <- 1:10
mean(x)
```

```
## [1] 5.5
```

```
y <- c(1,3,3,4,5,7,7,8,9,10)
m1 <- lm(y~x)
summary(m1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3636 -0.2455 -0.1273 -0.0455  0.8182
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4000     0.2949    1.36     0.21
## x             0.9636     0.0475   20.27  3.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.432 on 8 degrees of freedom
## Multiple R-squared:  0.981,  Adjusted R-squared:  0.979
## F-statistic:  411 on 1 and 8 DF,  p-value: 3.66e-08
```

The predicted value of **y** at **x** = 5.5 is simply: $y = b_0 + 5.5b_1$. We can think of **y** as a function of the regression coefficients, or $G(B)$:

$$G(B) = b_0 + 5.5b_1$$

We thus need to get the vector of partial derivatives of **`G(B)`** and the covariance matrix of **B**. The partial derivatives in this case are very easy to compute by hand: $\frac{dG}{db_0} = 1$ and $\frac{dG}{db_1} = 5.5$.

```
grad <- c(1, 5.5)
```

We can easily get the covariance matrix of **B** using **`vcov`** on the model object.

```
vb <- vcov(m1)
```

```
vb
```

```
##             (Intercept)        x
## (Intercept)      0.0870 -0.01242
## x               -0.0124  0.00226
```

Finally, we can approximate the standard error using the formula above.

```
vG <- t(grad) %*% vb %*% grad
sqrt(vG)
```

```
##       [,1]
## [1,] 0.137
```

It turns out the **`predict`** function with **`se.fit=T`** calculates delta method standard errors, so we can check our calculations against those from **`predict`**.

```
predict(m1, newdata=data.frame(x=5.5), se.fit=T)
```

```
## $fit
##   1
## 5.7
##
```

```
## $se.fit
## [1] 0.137
##
## $df
## [1] 8
##
## $residual.scale
## [1] 0.432
```

Looks like our manual calculations are good!

Now that we understand how to manually calculate delta method standard errors, we are ready to use the **`deltamethod`** function in the **`msm`** package. The **`deltamethod`** function expects at least 3 arguments. The first argument is a formula representing the function, in which all variables must be labeled as x1, x2, etc. The second argument are the means of the variables. Recall that $G(B)$ is a function of the regression coefficients, whose means are the coefficients themselves. $G(B)$ is not a function of the predictors directly. The third argument is the covariance matrix of the coefficients. By default, **`deltamethod`** will return standard errors of $G(B)$, although one can request the covariance of $G(B)$ instead through the fourth argument.

```
deltamethod(~ x1 + 5.5*x2, coef(m1), vcov(m1))
```

```
## [1] 0.137
```

Success!

Notice that in this example since $G(B)$ is a linear function, delta method is not an exact method and not an approximations.

Example 2: Odds ratio

Example 1 was somewhat trivial given that the **`predict`** function calculates delta method standard errors for adjusted predictions. Indeed, if you only need standard errors for adjusted predictions on either the linear predictor scale or the response variable scale, you can use **`predict`** and skip the manual calculations. However, other transformations of regrssion coefficients that **`predict`** cannot readily handle are often useful to report. One such tranformation is expressing logistic regression coefficients as odds ratios. As odds ratios are simple non-linear transformations of the regression coefficients, we can use the delta method to obtain their standard errors.

In the following example, we model the probability of being enrolled in an honors program (not enrolled vs enrolled) predicted by gender, math score and reading score. Here we read in the data and use **`factor`** to declare the levels of the **`honors`** such that the probability of "enrolled" will be modeled (R will model the probability of the latter factor level). We will run our logistic regression using **`glm`** with **`family=binomial`**.

```
d <- read.csv("https://stats.idre.ucla.edu/stat/data/hsbdemo.csv"
d$honors <- factor(d$honors, levels=c("not enrolled", "enrolled")
m3 <- glm(honors ~ female + math + read, data=d, family=binomial)
summary(m3)
```

```
##
## Call:
## glm(formula = honors ~ female + math + read, family = binomial
##     data = d)
```

```
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.006  -0.606  -0.273   0.484   2.395
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.9727     1.7387   -6.89  5.7e-12 ***
## femalemale   -1.1548     0.4341   -2.66   0.0078 **
## math          0.1317     0.0325    4.06  5.0e-05 ***
## read          0.0752     0.0276    2.73   0.0064 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 231.29  on 199  degrees of freedom
## Residual deviance: 150.42  on 196  degrees of freedom
## AIC: 158.4
##
## Number of Fisher Scoring iterations: 5
```

In the output above the regression coeffients are expressed on the log odds (linear predictor) scale. To express them as odds ratios, we simply exponentiate the coefficients. Let’s take a look at the math coefficient expressed as an odds ratio:

```
b2 <- coef(m3)[3]
exp(b2)
```

```
## math
## 1.14
```

So for each unit increase in math, we expect a 14% increase in the odds of being enrolled in the honors program.

We can use the same procedure as before to calculate the delta method standard error. First we define the transformation function, here a simple exponentiation of the coefficient for math:

$$G(B) = \exp(b_2)$$

The gradient is again very easy to obtain manually — the derivative of $\exp(X)$ is $\exp(X)$ — so $\nabla G(B) = \exp(b_2)$:

```
grad <- exp(b2)
grad
```

```
## math
## 1.14
```

Because our transformation only involves the coefficient for math, we do not want the entire covariance matrix of all regression parameters. We only want the variance of the math coefficient:

```
#do not want this
vcov(m3)
```

```
##             (Intercept) femalemale      math      read
## (Intercept)      3.0230    0.10703 -0.035147 -0.018085
## femalemale       0.1070    0.18843 -0.001892 -0.001287
## math            -0.0351   -0.00189  0.001054 -0.000427
## read            -0.0181   -0.00129 -0.000427  0.000761
```

```
#want this instead
vb2 <- vcov(m3)[3,3]
vb2
```

```
## [1] 0.00105
```

Finally, we are ready to calculate our delta method standard error

```
vG <- grad %*% vb2 %*% grad
sqrt(vG)
```

```
##       [,1]
## [1,] 0.037
```

Let's confirm our findings with **deltamethod**:

```
deltamethod(~ exp(x1), b2, vb2)
```

```
## [1] 0.037
```

Example 3: Relative risk

In the previous 2 examples, the gradient was easy to calculate manually because the partial derivatives of the transformation function were easy to determine. Many times, however, the gradient is laborious to calculate manually, and in these cases the **deltamethod** function can really save us some time. As before, we will calculate the delta method standard errors manually and then show how to use **deltamethod** to obtain the same standard errors much more easily.

In this example we would like to get the standard error of a relative risk estimated from a logistic regression. Relative risk is a ratio of probabilities. We will work with a very simple model to ease manual calculations. In this model, we are predicting the probability of being enrolled in the honors program by reading score. We would like to know the relative risk of being in the honors program when reading score is 50 compared to when reading score is 40.

```
d <- read.csv("https://stats.idre.ucla.edu/stat/data/hsbdemo.csv"
d$honors <- factor(d$honors, levels=c("not enrolled", "enrolled"))
m4 <- glm(honors ~ read, data=d, family=binomial)
summary(m4)
```

```
##
## Call:
## glm(formula = honors ~ read, family = binomial, data = d)
##
```

```
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.966  -0.662  -0.404   0.671   2.092
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.3002     1.2461   -6.66  2.7e-11 ***
## read          0.1326     0.0217    6.12  9.5e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 231.29  on 199  degrees of freedom
## Residual deviance: 179.06  on 198  degrees of freedom
## AIC: 183.1
##
## Number of Fisher Scoring iterations: 5
```

Let’s get the predicted probabilites of honors when read is 50 and when read is 40. Then we will get the ratio of these, the relative risk. The argument **`type="response"`** will return the predicted value on the response variable scale, here the probability scale.

```
p50 <- predict(m4, newdata=data.frame(read=50), type="response")
p50
```

```
##     1
## 0.158
## 0.158
```

```
p40 <- predict(m4, newdata=data.frame(read=40), type="response")
p40
```

```
##      1
## 0.0475
```

```
rel_risk <- p50/p40
rel_risk
```

```
##    1
## 3.33
```

Students with reading score 50 are 3.33 times as likely to be in enrolled in honors as those with reading score 40.

Now we want the standard error of this relative risk. As always, to begin we need the define the relative risk transformation as a function of the regression coefficients. First, we should define the conditional probability in terms of the regression coefficients. In our model, given a reading score X, the probability the student is enrolled in the honors program is:

$$Pr(Y=1|X) = \frac{1}{1+\exp(-\beta \cdot X)}$$

In our simple example, $\beta = (b0, b1)$. Therefore, the probabality of being enrolled in honors when reading = 50 is $Pr(Y=1|X=50) = \frac{1}{1+\exp(-b0 - b1 \cdot 50)}$, and when reading = 40 the probability of being enrolled in honors is $Pr(Y=1|X=40) = \frac{1}{1+\exp(-b0 - b1 \cdot 40)}$. The relative risk is just the ratio of these probabilities. So, the equation for the relative transformation function, $G(X)$, is (using generic X_1 and X_2 instead of 50 and 40, respectively):

$$G(X) = \frac{\frac{1}{1+\exp(-b_0 - b_1 \cdot X_1)}}{\frac{1}{1+\exp(-b_0 - b_1 \cdot X_2)}},$$

which simplifies to:

$$G(X) = \frac{1 + \exp(-b_0 - b_1 \cdot X_2)}{1 + \exp(-b_0 - b_1 \cdot X_1)}.$$

We now need the gradient, $\nabla G(X)$. Using, product rule and chain rule, we obtain the following partial derivatives:

$$\frac{dG}{db_0} = -\exp(-b_0 - b_1 \cdot X_2) \cdot p1 + (1 + \exp(-b_0 - b_1 \cdot X_2)) \cdot p1 * (1 - p1)$$

and

$$\frac{dG}{db_1} = -\exp(-b_0 - b_1 \cdot X_2) \cdot X_2 \cdot p1 + (1 + \exp(-b_0 - b_1 \cdot X_2)) \cdot X_1 \cdot p1 * (1-$$

where $p1 = Pr(Y = 1|X1) = \frac{1}{1+\exp(-b0 - b1 \cdot X_1)}$, so in our case, the probability of enrolled when read = 50. Let's calculate our gradient:

```
x1 <- 50
x2 <- 40
b0 <- coef(m4)[1]
b1 <- coef(m4)[2]
```

```
e1 <- exp(-b0 - 50*b1)
e2 <- exp(-b0 - 40*b1)
p1 <- 1/(1+e1)
p2 <- 1/(1+e2)
dgdb0 <- -e2*p1 + (1+e2)*p1*(1-p1)
dgdb1 <- -x2*e2*p1 + (1+e2)*x1*p1*(1-p1)
grad <- c(dgdb0, dgdb1)
grad
```

```
## (Intercept) (Intercept)
##      -0.368      13.280
```

Now we can calculate our standard error as before.

```
vG <- t(grad) %*% vcov(m4) %*% (grad)
sqrt(vG)
```

```
##       [,1]
## [1,] 0.745
```

With a more complicated gradient to calculate, **`deltamethod`** can really save us some time. Fortunately, $G(X)$ is not too bad to specify.

```
deltamethod( ~ (1 + exp(-x1 - 40*x2))/(1 + exp(-x1 - 50*x2)), c(b
```

```
## [1] 0.745
```

Much easier!

In sum, R provides a convenient function to approximate standard errors of transformations of regression coefficients with the function **`deltamethod`**. All that is needed is an expression of the transformation and the covariance of that is needed is an expression of the transformation and the covariance of the regression parameters.

Click here to report an error on this page or leave a comment

How to cite this page (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-cite-web-pages-and-programs-from-the-ucla-statistical-consulting-group/)

© 2021 UC REGENTS (http://www.ucla.edu/terms-of-use/) HOME (/) CONTACT (/contact)