Example of Wilks Confidence Interval

Mai Zhou

This example is about computing the Wilks confidence interval/confidence region for the logistic regression model. Similar procedure can also be used for Cox proportional hazards regression, censored data life regression or even the normal linear model, (but less often).

The theoretical base for the procedure is the (Wilks) likelihood ratio test: when the reduced model is valid, we have

$$-2[\max_{\beta} \log Lik(RM) - \max_{\beta} \log Lik(FM)] \sim \chi^2_{df}$$

where the degrees of freedom is the difference of number of free parameters between the FM and RM. [where FM stands for 'Full Model' and RM stand for 'Reduced Model']

When the RM is not valid then the distribution will be a non-central chi square distribution. We reject the RM for larger values of the test statistics.

Compare to the Wald confidence intervals:
(1) Wald confidence intervals are always symmetric about the MLE, Wilks do not. (2) Wilks confidence interval is "invariant" under transformation of parameters, Wald does not. (3) no need to compute variance when doing Wilks.

Example: This concerns an example with heart disease data from
http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/cleve.mod

We copy and paste it into a wordpad file, cut the front stuff and saved as a text file called cleveland.txt I am too lazy to type in the column names, but you can find out what they mean at the web page. (R is calling them V1, V2, ... V15 by default)

```
> cleveland <- read.table(file="C:/Users/mai/Desktop/cleveland.txt", header=FALSE)
> names(cleveland)
> dim(cleveland)
```

You should see 303 rows and 15 columns of data. The next to last column, with values buff or sick, is our response variable Y.

We want to try a model like (buff/sick $\sim$ age + sex + blood pressure + cholesterol)

```
> mylogitfit <- glm(V14~V1+V2+V4+V5, family=binomial(link=logit), data=cleveland)
> summary(mylogitfit)
```

You certainly should try to experimenting with "converting V14 into 0/1", and/or "converting V2 (male/female) into 0/1" and see what happens.

The summary should look like this

```
Call:
glm(formula = V14 ~ V1 + V2 + V4 + V5, family = binomial(link = logit),
    data = cleveland)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8472  -1.0202  -0.4327   1.0137   1.8974

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.367293   1.381622  -5.332 9.69e-08 ***
V1           0.057048   0.015507   3.679 0.000234 ***
V2male       1.707355   0.310873   5.492 3.97e-08 ***
V4           0.013164   0.007600   1.732 0.083282 .
V5           0.004566   0.002536   1.801 0.071758 .
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 417.64  on 302  degrees of freedom
Residual deviance: 363.90  on 298  degrees of freedom
AIC: 373.90

Number of Fisher Scoring iterations: 3
```

From the output we can easily get a 95% Wald confidence interval, for example, for β_1: it is [0.02665428, 0.08744172]. This is sometimes called the "plain vanilla" confidence interval, meaning no transformation applied to β_1.

```
> 0.057048 + 1.96*0.015507
[1] 0.08744172
> 0.057048 - 1.96*0.015507
[1] 0.02665428
```

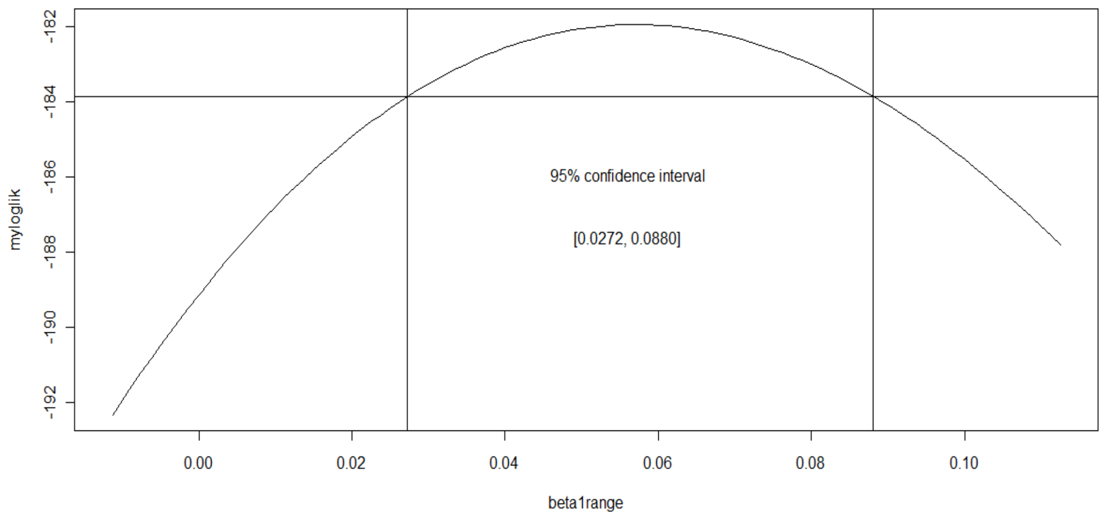
Now let us try to find the Wilks confidence interval for β_1. We need to find many log likelihood values when β_1 is fixed at values near the MLE (=0.057048).

```
> beta1range <- (1:100)/1000
> myloglik <- rep(NA, 100)
> for(i in 1:100) myloglik[i] <- logLik(glm(V14~offset(V1*beta1range[i])
                 +V2+V4+V5,family=binomial(link=logit), data=cleveland))
> plot(beta1range, myloglik, type="l")
```

Now shave $3.84/2 = 1.92$ off from the top, on the plot.

```
> abline( h= max(myloglik) - 1.92)
```

You "see" the confidence interval now. The two confidence intervals are very similar, as sample size n=303 is large here. For smaller n they will be somewhat different. To calculate the numeric value of the upper/lower confidence bound, you may try the function `findUL( )` in the package `emplik`.



Notice the max value: $\max(myloglik) = -181.9481$, and $-2 * \max(myloglik) = 363.8962$ is the "Residual Deviance" reported by the `glm` output. Also the "Null Deviance" reported there is the $-2 * \max(loglik(intercept\ only))$. (So, the difference of the two can be used to test a hypothesis.......)

Next, let us try a confidence region for β_1 (slope for `V1`) and β_4 (slope for `V4`) jointly. We already have a `beta1range`, we need `beta4range` as trial values. From the summary output, we know the β_4 value is near 0.013164 (we do not want to wonder too far away from the peak (the MLE).

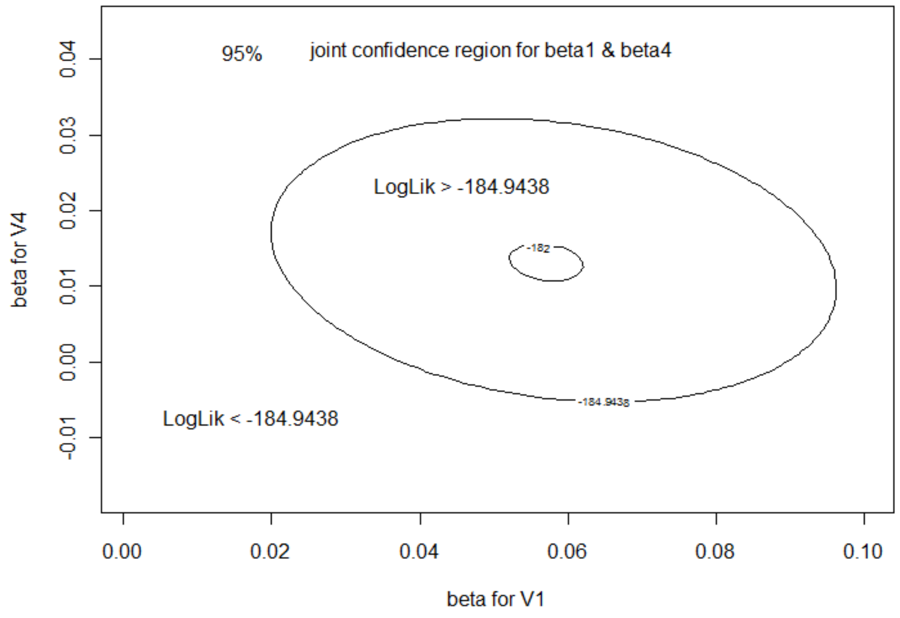
```
> beta4range <- 1:100/1600 - 0.018
> myloglik2 <- matrix(NA, nrow=100, ncol=100)
```

We are ready to compute the log likelihood values, while holding β_1 and β_4 at various values. (this takes 1 full minute on my laptop :-()

```
> for(i in 1:100) {
        for(j in 1:100) myloglik2[i,j] <- logLik(glm(V14~
           offset(V1*beta1range[i]+V4*beta4range[j])+V2+V5,
                family=binomial(link=logit), data=cleveland))
                }
```

The maximum of the log likelihood remains to be -181.9481. In the plot we will shave off 5.99/2 = 2.995732 (where 5.99 =`qchisq(0.95, df=2)`) from the top.

```
> contour(x=beta1range,y=beta4range,z=myloglik2,levels=c(-182, -181.9481-2.995732),
                  xlab="beta for V1", ylab="beta for V4")
```



Notes when compute log likelihood with R function `survreg`:

1. The function `logLik` do not work with `survreg` yet. Instead, we use `survreg(....)$loglik[2]`

2. when covariate is of categorical type, you may need to convert it into numerical type first.

1 Use function `findUL` to find confidence intervals

Finding confidence interval by trial and error or use graphics are tedious sometimes. We have a function `findUL` in the package **emplik** for doing this automatically.

The required inputs of `findUL` are: (1) `fun`. a function that will return the '-2LLR'. To be specific, a function that will return a list, one of which is called '-2LLR', which is the -2 log likelihood ratio. This function should have inputs, the first input, called `theta`, is the parameter value, and the rest of the inputs are usually the data. (2) MLE. This input of `fun` is just the MLE of the parameter (same parameter of the first input of fun). Do not need to be exact. This is where search starts.

There are two optional inputs for the `findUL`. (1) `step` This is the step size of the search. Included here to avoid the situation like the confidence interval is of length

~ 1000 and the search only change by 0.01 each step. A reasonable `step` will be $\sim$ half the standard deviation of the MLE. Smaller step usually works as well, just slightly slower, but too large a step can result in failure of finding the confidence interval. So, when in doubt, use a smaller value, as long as it is of same order of magnitude to the SD of MLE.

(2) `level` Defalut is 3.84. This is the 95% quantile of chi square 1. Set it to 2.70 if you want a 90% confidence interval instead.

Example 1 Find the 95% confidence interval for the mean, with right censored data.

Since our parameter is the mean, which is $\int t dF(t)$, so the `fun` in the `el.cen.EM2` is just the function $f(t) = t$:

```
>  myfun <- function(theta, x, d) {
             el.cen.EM2(x=x, d=d, fun=function(t){t}, mu=theta)
               }
```

We then call the `findUL` to find out the confidence interval

```
> library(emplik)
> data(smallcell)
> findUL(step=6, fun=myfun, MLE=720, x=smallcell$survival, d=smallcell$indicator)
```

Here 720 is not the MLE but a reasonable guess. Other values within 660 and 860 will all work.

Example 2 Median. Same data. Change `myfun` to

```
>  myfun <- function(theta, x, d) {
             el.cen.EM2(x=x, d=d, fun=function(t){ as.numeric(t <= theta) }, mu=0.5)
              }
> findUL(step=6, fun=myfun, MLE=520, x=smallcell$survival, d=smallcell$indicator)
```

Technically you should first find the MLE of the median, and then enter the value in the above function. But 520 is a good enough guestimator.

Example 3 Survival probability at a given time. Take the time to be 365.25. Notice we shall work with $F(365.25)$ which is the one minus the survival probability at one year.

```
>  myfun <- function(theta, x, d) {
           el.cen.EM2(x=x, d=d, fun=function(t){ as.numeric(t <= 365.25) }, mu=theta)
               }
> findUL(step=0.06, fun=myfun, MLE=0.3, x=smallcell$survival, d=smallcell$indicator)
```

Again, you should first work out the MLE of the survival probability at 365.25, and this is given by the Kaplan-Meier (or 1- Kaplan-Meier). But MLE=0.3 is a good guestimator.

Caution: In all cases, if in the output, the `Uvalue` and `Lvalue` is larger than 3.84, this indicate something is wrong. They should both be close to 3.84, sometimes slightly below, due to discreteness (as is in the median case).

Finally, an example of 90% confidence interval of 1 - survival probability at one year.

```
> myfun <- function(theta, x, d) {
             el.cen.EM2(x=x, d=d, fun=function(t){ as.numeric(t <= 365.25) }, mu=theta)
                }
> findUL(step=0.06, fun=myfun, MLE=0.3, level=2.7055, x=smallcell$survival, d=smallcell$
```