Wilson Interval

Contents

1 Wilson Interval

In this document, I explain why we prefer Wilson interval over the standard normal appoximation confidence interval (i.e. Wald type CI) for a binomial proportion in the `glmdr` framework for the prediction.

1.1 Definition.

1.1.1 Coverage probability

Coverage probability compute how often a confidence interval will contain the true value of the parameter:

$C(p, n) = \mathbb{P}_p(p \in CI), 0 < p < 1.$

Usually, when we construct a confidence interval, we want the actual coverage probability to be close to the nominal confidence level. Because of the discrete nature of the binomial distribution, $C(p, n) \neq 1 - \alpha$.

1.1.2 Standard interval

Assume $\hat{p} \overset{\text{approx}}{\sim} N(p, \frac{p(1-p)}{n})$,

the standard normal approximation confidence interval CI_s is defined as: $CI_s = \hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. This interval is obtained by inverting the acceptance region of the Wald large-sample normal test.

Standard interval shows the low (or unacceptable) coverage probability when p is near $0, 1$ boundaries or small number of samples.

1.1.3 Wilson interval

Wilson interval is developed by Edwin Wilson (1927) and this interval shows more constant interval for different p and small number of n.

Wilson interval is given by

$CI_{Wilson} = \frac{1}{1+\frac{z^2}{n}}(\hat{p} + \frac{z^2}{2n}) \pm \frac{z}{1+\frac{z^2}{n}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}.$

Also, there is continuity corrected version of Wilson interval which aligns the minimum coverage probability.

1.2 Simulation

In this section, we will see how poor standard interval performs for different value of p.

1.2.1 Function

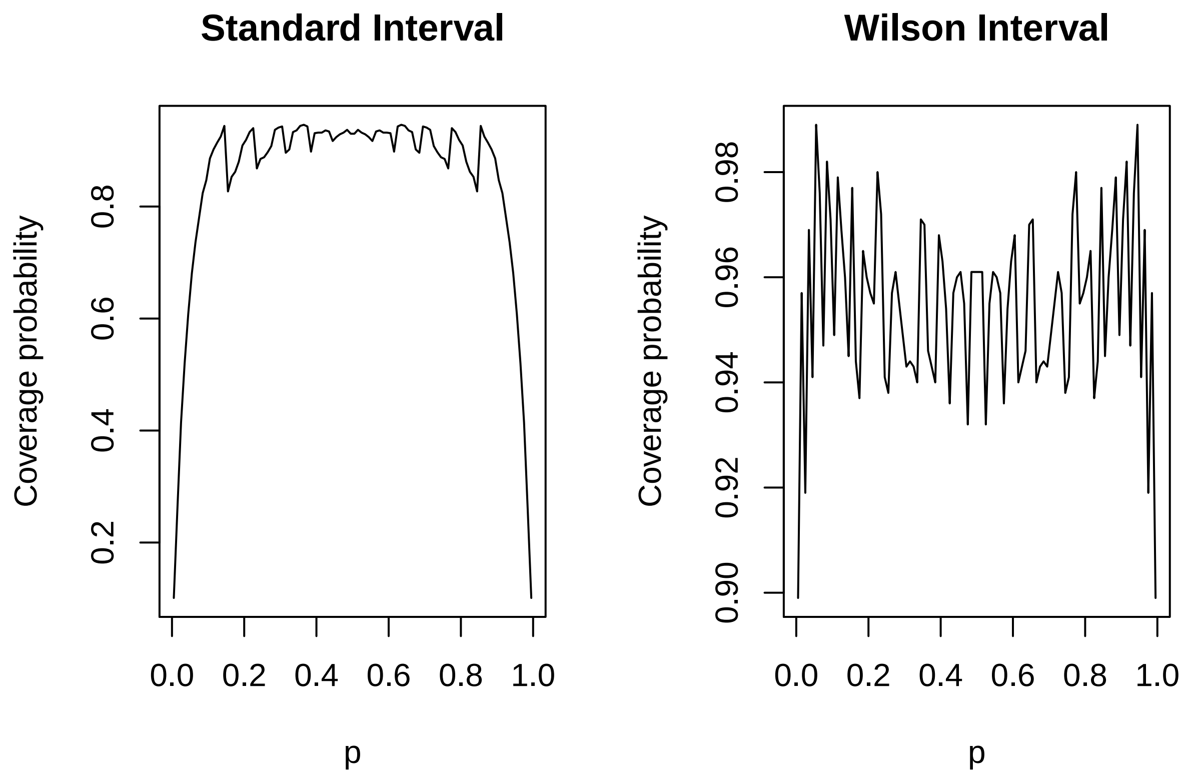
```r
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators

## Loading required package: parallel
```

```r
numCores <- detectCores()
registerDoParallel(numCores)
Wald_coverage_prob <- function(n_sample,true_p,conf.level,num_simulation){
  ans <- rep(0,num_simulation)
  for(i in 1:num_simulation){
    dat <- rbinom(n = n_sample, size=1, prob=true_p)
    p_hat <- mean(dat)
    se <- sqrt(p_hat * (1 - p_hat) / n_sample)
    z_star <- qnorm(p = (1-conf.level) / 2, lower.tail = FALSE)
    intervals <- c(lower = p_hat - z_star * se, upper = p_hat + z_star * se)
    ans[i] = (intervals[1] < true_p) & (intervals[2] > true_p)
  }
  mean(ans)
}
Wilson_coverage_prob <- function(n_sample,true_p,conf.level=0.95,num_simulation){
  ans <- rep(0,num_simulation)
  for(i in 1:num_simulation){
    dat <- rbinom(n = n_sample, size=1, prob=true_p)
    phat <- mean(dat)
    q <- 1-phat
    z_val <- qnorm(p = (1-conf.level) / 2, lower.tail = FALSE)
    part1 <- 1/(1+(z_val^2/n_sample))
    part2 <- phat+(z_val^2/(2*n_sample))
    part3 <- z_val/(1+(z_val^2/n_sample))
    part4 <- sqrt(((phat*q)/n_sample)+z_val^2/(4*n_sample^2))
    lower <- part1*part2 - part3*part4
    upper <- part1*part2 + part3*part4
    ans[i] = (lower < true_p) & (true_p < upper)
  }
  mean(ans)
}
Wilsoncc_coverage_prob <- function(n_sample,true_p,conf.level=0.95,num_simulation){
  ans <- rep(0,num_simulation)
  for(i in 1:num_simulation){
    dat <- rbinom(n = n_sample, size=1, prob=true_p)
    phat <- mean(dat)
    q <- 1-phat
    z_val <- qnorm(p = (1-conf.level) / 2, lower.tail = FALSE)
    lower_numerator <- 2*n_sample*phat+z_val^2-(z_val*sqrt(z_val^2-1/n_sample+4*n_sample*phat*q+(4*phat
    upper_numerator <- 2*n_sample*phat+z_val^2+(z_val*sqrt(z_val^2-1/n_sample+4*n_sample*phat*q-(4*phat
    denom <- 2*(n_sample+z_val^2)
    lower <- max(0,lower_numerator/denom)
    upper <- min(1,upper_numerator/denom)
    ans[i] = (lower < true_p) & (true_p < upper)
  }
  mean(ans)
}
```

1.2.2 Fixed n and different p

```
n_sample = 20
p_vals = seq(0.005,0.995,length.out=100)
ans = rep(0,length(p_vals))
ans2 = rep(0,length(p_vals))
ans3 = rep(0,length(p_vals))
ret <- foreach(p=p_vals) %dopar% {
  set.seed(42)
  Wald_coverage_prob(n_sample=n_sample,true_p=p,conf.level = 0.95,num_simulation=1000)
}
ans <- do.call(c,ret)
ret <- foreach(p=p_vals) %dopar% {
  set.seed(42)
  Wilson_coverage_prob(n_sample=n_sample,true_p=p,conf.level = 0.95,num_simulation=1000)
}
ans2 <- do.call(c,ret)

par(mfrow=c(1,2))
plot(p_vals,ans, type="l", xlab="p", ylab="Coverage probability", main="Standard Interval")
plot(p_vals,ans2, type="l", xlab="p", ylab="Coverage probability", main="Wilson Interval")
```



We can see the coverage probability for standard interval performs poor, especially near p is $0, 1$ boundaries. It means, when true probability of success, p , is near boundaries, we cannot trust standard interval. On the other hand, wilson interval shows much more stable results even p is $0, 1$ boundaries.

1.3 Prediction

In `glmdr` framework, the prediction procedure is as follows:

1. Generate two testing sets combining training sets and single new data point with $y = 0$ and $y = 1$ separately.
2. Fit the logistic model for each datasets.
3. Compute the estimated probability from each model, $\hat{p_0}$ and $\hat{p_1}$
4. Calculate the Akaike model weigths using AICc, w_0 and w_1.
5. Compute the weighted estimated probability, $\widehat{p^*} = w_0\hat{p_0} + w_1\hat{p_1}$.
6. Construct the **Wilson interval**.
7. Label 0 or 1 based on the mean of intervals, $\hat{y} = 1$ if mean of intervals ≥ 0.5, otherwise $\hat{y} = 0$.

Initially, we wanted to construct the confidence interval for weighted estimated probability from step 5 to provide more conservative result. Since the binary logistic regression models the probability of success that follows the binomial distribution, we consider (any) binomial proportion confidence interval for step 6. And as we can see in the simulation section, **standard confidence interval performs poorly when the probability of success is near $0, 1$ boundaries, and this is reason why we prefer Wilson interval over the standard confidence interval**.

1.4 Estimation (Inference)

Currently, we compute the confidence interval for bayesglm and lgostif/brglm2 as follows:

1. Fit the model.
2. Using `predict` function, find the estimated probability $\hat{p}$.
3. **Compute the Wilson interval for $\hat{p}$ for single trial $n = 1$**
4. Calculate the length between lower and upper bound.

Considering the Wald confidence interval for predicted probabilities in GLM,

$\text{logit}^{-1}\left(x_*^T\hat{\beta} \pm t_{\alpha/2}\sqrt{x_*^T(X^T\widehat{W}X)^{-1}x_*}\right)$ where $\widehat{W}$ is diagonal matrix of weights (i.e. variance of observation).

Simply plugging $\text{logit}^{-1}\left(x_*^T\hat{\beta}\right) = \hat{p}$ from bayesglm/lgostif/brglm2/OLS into Wilson interval may or may not make sense.

1.4.1 Question: Correct way to compute the Wald type confidence interval

I think we discuss this before but I would be grateful if we could double check (I am really sorry for bringing this again). Based on our conversation last meeting, I showed the Wald type interval for Agresti complete separation example (right panel of figure on next page, I used `type="link"` then took inverse logit for the result of confidence interval formula) and you pointed out left panel of figure (`"type=response"`) was correct one and I should have set the `type ="response"` in `predict` function before computing the confidence interval for predicted probabilities. But considering the Wald confidence interval for predicted probabilities for completely degenerated case, $\text{logit}^{-1}\left(x_*^T\hat{\beta} \pm t_{\alpha/2}\sqrt{x_*^T(X^T\widehat{W}X)^{-1}x_*}\right)$,

the diagnal value of $\widehat{W}$ is close to 0 (no variance for each observation) so, $(X^T\widehat{W}X)^{-1}$, diagonal value of variance-covariance matrix of $\hat{\beta}$ is high and confidence interval for predicted probabilities becomes 0 fow lower bound and 1 for upper bound. Also, I referred to the Agresti, 3rd categorical data analysis page 170 to confirm my calculation.

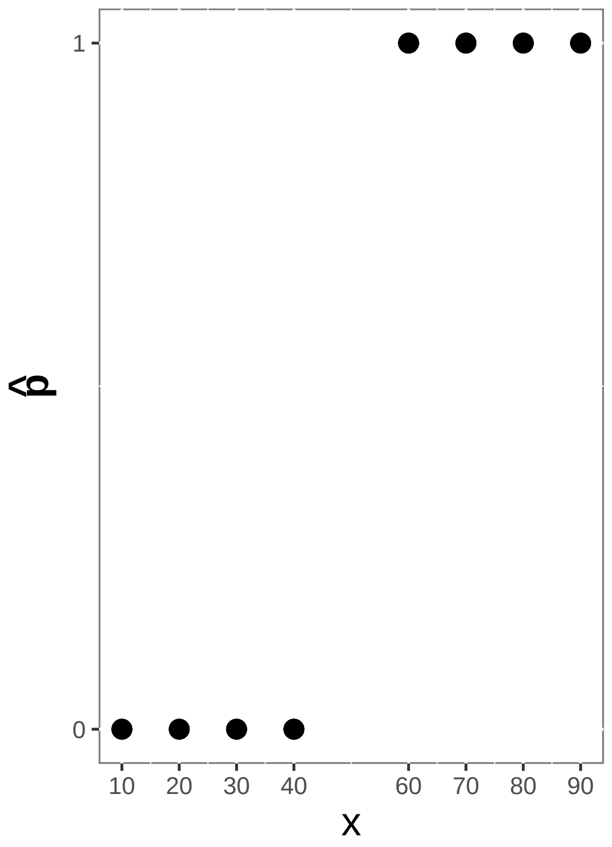
```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

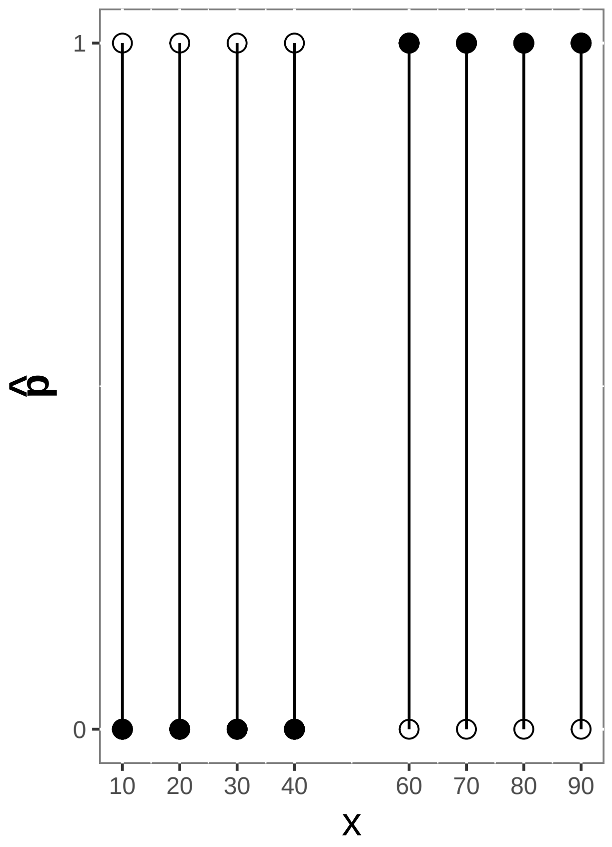
```r
library(gridExtra)
library(latex2exp)
x <- c(10,20,30,40,60,70,80,90)
y <- c(0,0,0,0,1,1,1,1)
glm_mod <- glm(y~x,family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
pred1 <- predict(glm_mod, type="response", se.fit=TRUE)
lower1 <- pred1$fit - 1.96 * pred1$se.fit
upper1 <- pred1$fit + 1.96 * pred1$se.fit

pred2 <- predict(glm_mod, type="link", se.fit=TRUE)
lower2 <- glm_mod$family$linkinv(pred2$fit - 1.96 * pred2$se.fit)
upper2 <- glm_mod$family$linkinv(pred2$fit + 1.96 * pred2$se.fit)

plot1 <- ggplot(data=as.data.frame(cbind(x,y,lower1,upper1)),aes(x=x,y=y)) +
  geom_point(size=3) +
  geom_point(aes(x=x, y=lower1),size=3,pch=1) +
  geom_point(aes(x=x, y=upper1),size=3,pch=1) +
  geom_point(size=3) +
  geom_segment(aes(x = x, y = y, xend = x, yend = lower1)) +
  geom_segment(aes(x = x, y = y, xend = x, yend = upper1)) +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"), axis.title=element_text(siz
  labs(x="x", y= TeX('$\\textbf{\\hat{p}}$')) +
  scale_x_continuous(breaks = x)+
  scale_y_continuous(breaks = c(0.0,1.0))

plot2 <- ggplot(data=as.data.frame(cbind(x,y,lower2, upper2)),aes(x=x,y=y)) +
  geom_point(size=3) +
  geom_point(aes(x=x, y=lower2),size=3,pch=1) +
  geom_point(aes(x=x, y=upper2),size=3,pch=1) +
  geom_point(size=3) +
  geom_segment(aes(x = x, y = y, xend = x, yend = lower2)) +
  geom_segment(aes(x = x, y = y, xend = x, yend = upper2)) +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"), axis.title=element_text(siz
  labs(x="x", y= TeX('$\\textbf{\\hat{p}}$')) +
  scale_x_continuous(breaks = x)+
  scale_y_continuous(breaks = c(0.0,1.0))

grid.arrange(plot1,plot2, ncol=2)
```



$\hat{p}$
1
0
10
20
30
40
60
70
80
90
x



$\hat{p}$
1
0
10
20
30
40
60
70
80
90
x