Importance sampling

Qinqing Li

2024-08-31

Three alternatives for Poisson parameter confidence intervals

Asymptotic Normality of the Maximum Likelihood Estimator (MLE)

Thm. : under mild assumptions, the difference between the MLE of θ and the true value of θ becomes approx. Gaussian matrix. With covariance matrix: inverse Fisher Information matrix.

$\hat{\theta}_{ML} - \theta \approx N(0, \tilde{H}(\theta)^{-1})$

Note information is inversely proportional to uncertainty (variance-covariance), the more information the less uncertainty. Therefore, we use the inverse Fisher matrix to quantify the uncertainty, instead of using the Fisher Information matrix to measure the amount of information.

when we have 1 param, converge to Normal distribution:

$\frac{\hat{\theta}_{ML}-\theta}{\sqrt{var(\hat{\theta}_{ML})}} \to N(0,1)$ and

$var(\hat{\theta}_{ML}) = [\tilde{H}(\theta)^{-1}]_{11}$

Interval construction

```
# Two-tailed test: 2.5% in each tail for 95% confidence level
# z1 <- qnorm(0.975)
# z2 <- qnorm(0.025)

CI1 <- function(y, alpha = 0.05) {
  lambda_hat <- sum(y)/length(y)
  lwr <- lambda_hat - sqrt(lambda_hat/length(y)) * qnorm(1 - alpha / 2)
  upr <- lambda_hat - sqrt(lambda_hat/length(y)) * qnorm(alpha / 2)
  return(c(lwr = lwr, upr = upr))
}

CI2 <- function(y, alpha = 0.05) {
  y_bar <- sum(y)/length(y)
  lwr <- (max(0,sqrt(y_bar)-qnorm(0.975)/(2*sqrt(length(y)))))**2
  upr <- (sqrt(y_bar) - qnorm(0.025)/(2*sqrt(length(y))))**2
  return(c(lwr = lwr, upr = upr))
}

CI3 <- function(y, alpha = 0.05) {
  n<-length(y)
  theta_hat<-log(mean(y))
  lwr <- theta_hat - qnorm(1 - alpha / 2)/sqrt(n*exp(theta_hat))
```

```r
  upr <- theta_hat - qnorm(alpha / 2)/sqrt(n*exp(theta_hat))
  return(c(lwr = exp(lwr), upr = exp(upr)))
}
```

```r
set.seed(12)
y <- rpois(n = 5, lambda = 2)
```

```r
CI <- rbind(
  "Method 1" = CI1(y),
  "Method 2" = CI2(y),
  "Method 3" = CI3(y)
)
colnames(CI) <- c("Lower", "Upper")
```

Will all three methods always produce a valid interval? Ans: Not always, use n = 5 and lambda = 0.1, method 1 gives CI (-0.192, 0.592).

Bayesian credible intervals

side note: frequentist approach to statistical inference: there exists a true value for parameter θ, take repeated samplings to interpret the true value. Bayesian approach: θ is a r.v. Investigator has prior beliefs about θ before any observation of data, summarised in prior distribution $\pi(\theta)$. When data $Y = y$ is observed, the extra information is combined prior to obtaining posterior distribution $\pi(\theta|x)$ for θ given $Y = y$.

transforming between 2 distributions: $p(\theta) = p(\lambda)\frac{d\lambda(\theta)}{d\theta}$

Importance sampling

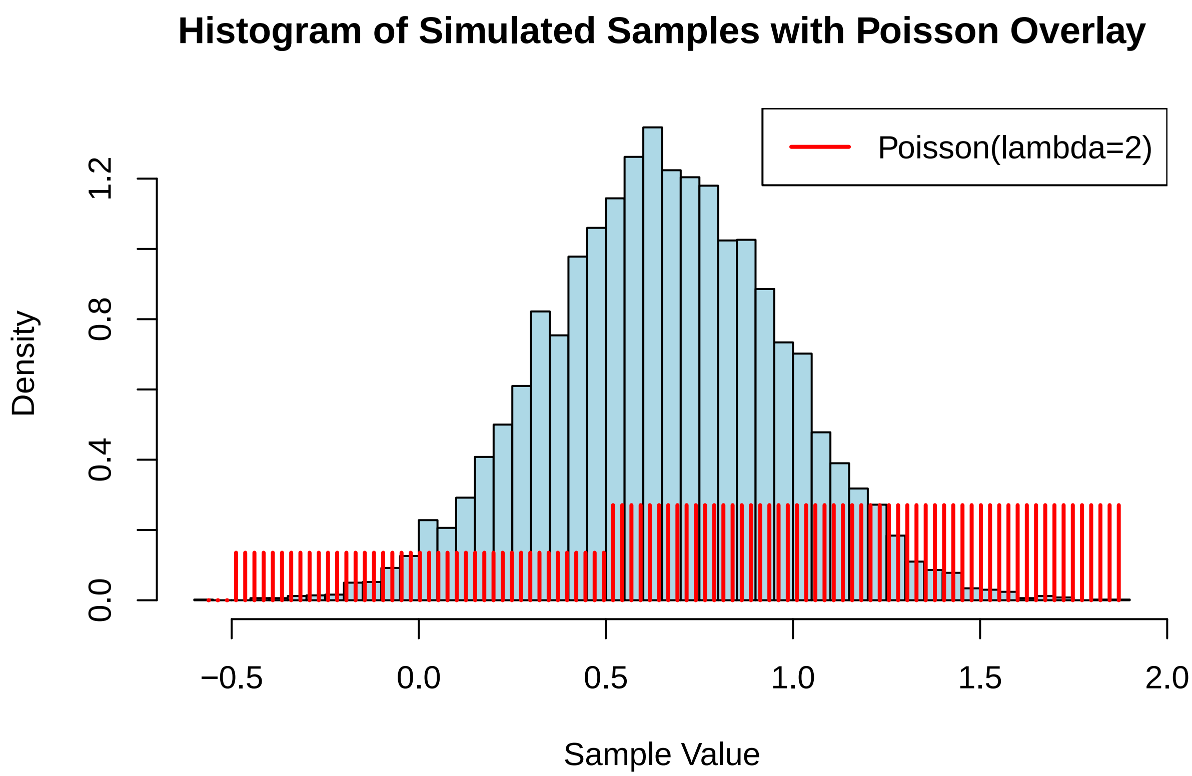
```r
sim_sample <- function(y,m,a) {
  set.seed(123)
  n<- length(y)
  var_est <- 1/(1+n*mean(y))
  samples <- rnorm(m, mean = log(1 + sum(y)) - log(a + n), sd = sqrt(var_est))
  return(samples)
}
```

plot the simulated sample

```r
hist(samples1, breaks = 50, probability = TRUE, main = "Histogram of Simulated Samples with Poisson Ove
     xlab = "Sample Value", col = "lightblue", border = "black")

# Add Poisson distribution overlay
lambda <- 2
x_vals <- seq(min(samples1), max(samples1), length.out = 100)
pois_vals <- dpois(round(x_vals), lambda)
lines(x_vals, pois_vals, type = "h", col = "red", lwd = 2)

# Add legend
legend("topright", legend = "Poisson(lambda=2)", col = "red", lwd = 2)
```



solution code below. note x and sample1 are both sim sample

```
a <- 0.2
m <- 10000
n<-length(y)
set.seed(123)
x <- rnorm(m, mean = log(1+sum(y)) - log(a + n), sd = 1 / sqrt(1 + sum(y)))
```

calculate unormalised importance weights: (clearly weight do not add up to 1 yet.)

```
unnormalised_imp_w <- function(y,a = 1/5) {
  n <- length(y)
  #note log_posteror_distri is proportional to likelihood times prior,
  #where the constant of proportionality is chosen to make the total mass of the
  #posterior distribution equal to 1
  log_posteror_distri <- (x * (1 + sum(y)) - (a + n) * exp(x))
  w <- log_posteror_distri - dnorm(x, mean = log(1+sum(y)) - log(a + n),
                                   sd = 1 / sqrt(1 + sum(y)), log = TRUE)
  w1 <- exp(w - max(w))
  return(w1)
}
sum(unnormalised_imp_w(y))
```

```
## [1] 1042.125
```

```
#n is length of y
#a is 1/5
log_weights <- (x * (1 + sum(y)) - (a + n) * exp(x)) -
 dnorm(x, mean = log(1 + sum(y)) - log(a + n), sd = 1 / sqrt(1 + sum(y)), log = TRUE)
weights <- exp(log_weights - max(log_weights))
sum(weights)
```

```
## [1] 1042.125
```

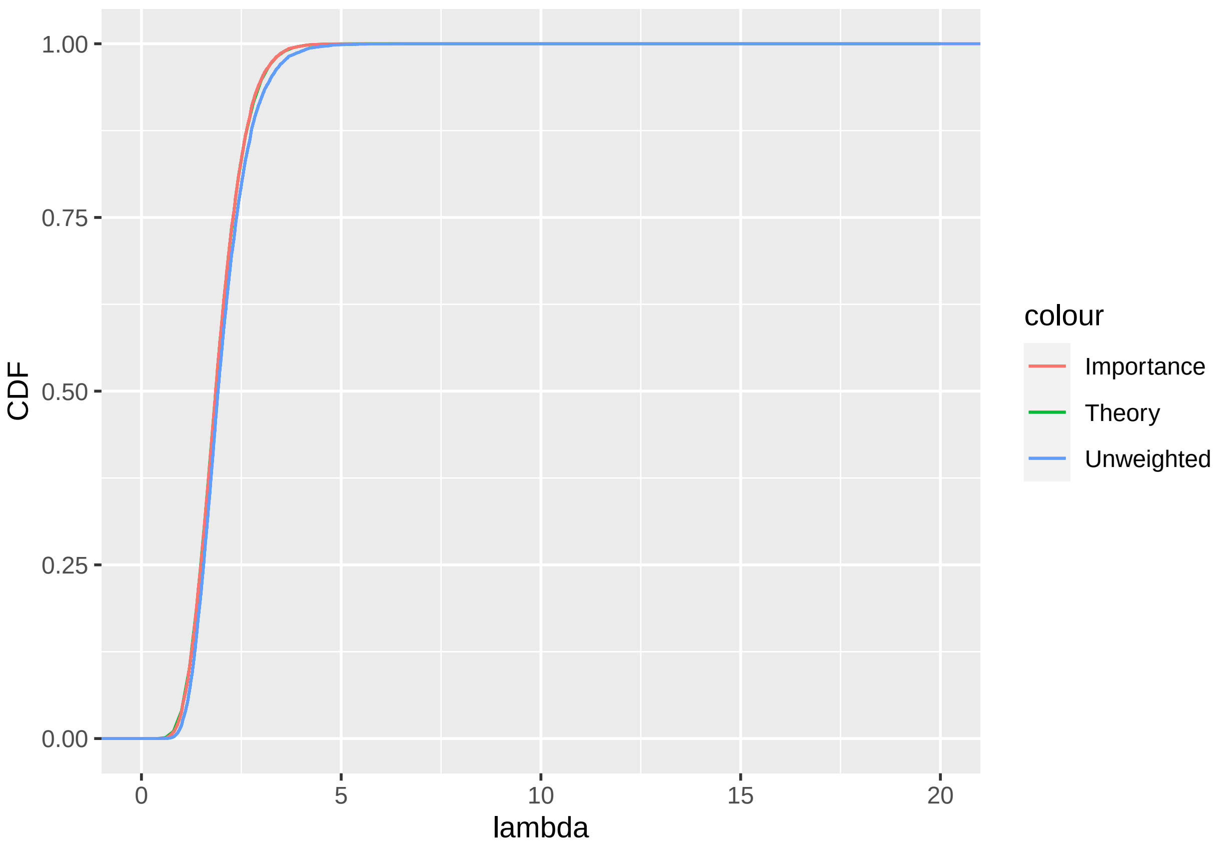
```r
theta_interval <- wquantile(x, probs = c(0.025, 0.975), weights = weights)
theta_interval
```

```
## [1] -0.06914057  1.18771247
```

```r
lambda_interval <- exp(theta_interval)
lambda_interval
```

```
## [1] 0.9331955 3.2795705
```

```r
ggplot(data.frame(lambda = exp(x), weights = weights)) +
  xlim(0, 20) + ylab("CDF") +
  geom_function(fun = pgamma, args = list(shape = 1 + sum(y), rate = a + n),
                mapping = aes(col = "Theory")) +
  stat_ewcdf(aes(lambda, weights = weights, col = "Importance")) +
  stat_ecdf(aes(lambda, col = "Unweighted"))
```




```r
# Plotting updated to include a plot of the importance weights
p1 <-
  ggplot(data.frame(lambda = exp(x), weights = weights)) +
  ylab("CDF") +
  geom_function(fun = pgamma, args = list(shape = 1 + sum(y), rate = a + n),
                mapping = aes(col = "Theory")) +
  stat_ewcdf(aes(lambda, weights = weights, col = "Importance")) +
  stat_ecdf(aes(lambda, col = "Unweighted")) +
  scale_x_log10(limits = c(3, 20)) +
  labs(colour = "Type")
```

```r
p2 <- ggplot(data.frame(lambda = exp(x), weights = weights),
             mapping = aes(lambda, weights / mean(weights))) +
  ylab("Importance weights") +
  geom_point() +
  geom_line() +
  geom_hline(yintercept = 1) +
  scale_y_log10() +
  scale_x_log10(limits = c(3, 20))

# The patchwork library provides a versatile framework
# for combining multiple ggplot figures into one.
library(patchwork)
(p1 / p2)
```

```
## Warning: Removed 9218 rows containing non-finite values (`stat_ewcdf()`).

## Warning: Removed 9218 rows containing non-finite values (`stat_ecdf()`).

## Warning: Removed 9218 rows containing missing values (`geom_point()`).

## Warning: Removed 9218 rows containing missing values (`geom_line()`).
```