Ejercicio 7

José Antonio Álvarez Ocete

Importamos los paquetes necesarios:

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## Warning: package 'purrr' was built under R version 3.6.3

## Warning: package 'dplyr' was built under R version 3.6.3

## Warning: package 'forcats' was built under R version 3.6.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(gapminder)
```

```
## Warning: package 'gapminder' was built under R version 3.6.3
```

```r
library(comprehenr)
library(ggplot2)
library(dplyr)
theme_set(theme_bw())
```

Implementamos una función que calcula el coeficiente de asimetría muestral de una muestra dada. Utilizaremos el estimador natural: sustituir la esperanza por el promedio de los valores, la media por la media muestral y la varianza por la varianza muestral, siendo conscientes de que esta varianza muestra es s^2 y no σ^2.

```r
asymmetry_coefficient <- function(muestra) {
  mean <- mean(muestra)
  sigma <- sd(muestra)
  mean((muestra - mean)^3 / sigma)
}
```

Implementamos una función que pinta los intervalos alrededor del valor real θ, sirviéndonos del código proporcionado en las diapositivas.

```r
plot_intervals <- function(intervals, hits, theta, method_name, distribution_name) {
  m <- nrow(intervals)
  df <- data.frame(ic_min <- intervals[,1],
                   ic_max <- intervals[,2],
                   ind = 1:m,
                   hits = hits)

  gg <- ggplot(df) +
    geom_linerange(aes(xmin=ic_min, xmax=ic_max, y=ind, col=hits)) +
    scale_color_hue(labels = c("NO", "SÍ")) +
    geom_vline(aes(xintercept = theta), linetype = 2) +
    theme_bw() +
    labs(y = 'Muestras', x = 'Intervalos (nivel 0.95)',
         title = paste('IC -', method_name, 'para', distribution_name, sep=' '))
  print(gg)
}
```

Implementamos dos funciones auxiliares adicionales. La primera muestra una distribución (normal o exponencial) `n` veces. La segunda calcula el intervalo de confianza del coeficiente de asimetría utilizando el método proporcionado por parámetro.

```r
sample_dist <- function(distribution, n) {
  if (distribution == 'normal') {
    rnorm(n, 0, 1)
  } else if (distribution == 'exponential') {
    rexp(n, 1)
  } else {
    print('Distribution not supported')
  }
}

compute_interval <- function(method, gammas_bootstrap, gamma_original, alfa) {
  if (method == 'Método híbrido') {
    # Metodo hibrido
    n = length(gammas_bootstrap)
    T_bootstrap <- sqrt(n) * (gammas_bootstrap - gamma_original)
    ic_min <- gamma_original -  quantile(T_bootstrap, 1-alfa/2)/sqrt(n)
    ic_max  <- gamma_original -  quantile(T_bootstrap, alfa/2)/sqrt(n)
    c(ic_min, ic_max)
  } else if (method == 'Método normal') {
    # Metodo normal
    et_boostrap <- sd(gammas_bootstrap)
    ic_min <- gamma_original + qnorm(alfa/2, 0, 1)*et_boostrap
    ic_max  <- gamma_original - qnorm(alfa/2, 0, 1)*et_boostrap
    c(ic_min, ic_max)
  } else if (method == 'Método percentil') {
    # Metodo percentil
    ic_min <- quantile(gammas_bootstrap, alfa/2)
    ic_max  <- quantile(gammas_bootstrap, 1-alfa/2)
    c(ic_min, ic_max)
  } else {
    print('Method not supported')
  }
}
```

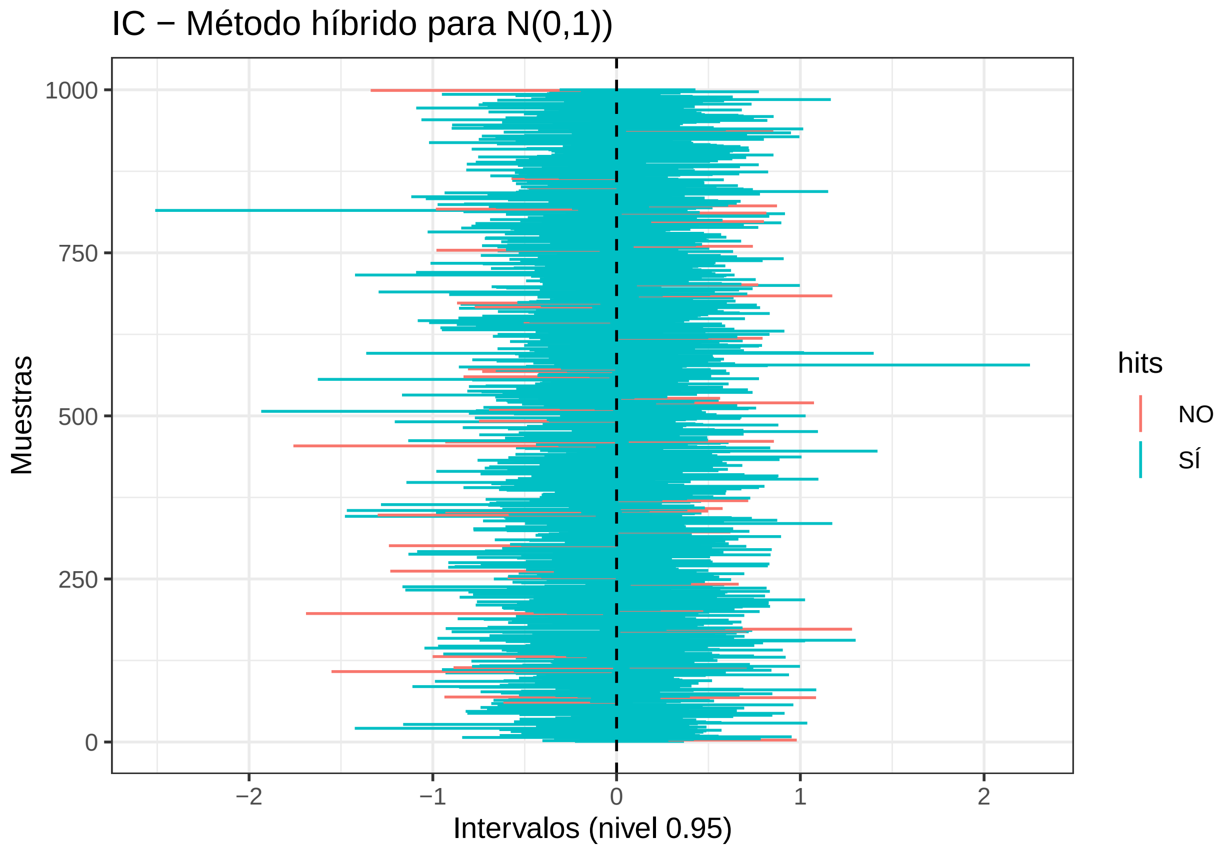
Utilizando las funciones anteriores implementamos una función final que, dada una distibución y un método, calcula `n=100` muestras originales, remuestrea `R=1000` veces y calcula el intervalor de confianza para el coeficiente de asimetría. Repetimos este proceso `m=1000` veces y dibujamos los distintos intervalos de confianza. Adicionalmente se mostrará la precisión del método.

```
exercise_2 <- function (distribution, method) {
  R <- 1000
  n <- 100
  m <- 1000

  alfa <- 0.05
  theta = if (distribution == 'normal') 0 else 2

  intervalos <- NULL
  aciertos <- NULL

  for (i in 1:m) {
    # Usar la distribución para muestrear
    muestra_original <- sample_dist(distribution, n)
    gamma_original <- asymmetry_coefficient(muestra_original)

    # Muestreo boostrap y computar los estimadores boostrap
    muestras_bootstrap <- sample(muestra_original, n*R, rep = TRUE)
    muestras_bootstrap <- matrix(muestras_bootstrap, nrow = n)
    gammas_bootstrap <- apply(muestras_bootstrap, 2, asymmetry_coefficient)

    # Computar el intervalo y el acierto
    interv <- compute_interval(method, gammas_bootstrap, gamma_original, alfa)
    intervalos <- rbind(intervalos, interv)
    aciertos <- c(aciertos, interv[1] < theta & interv[2] > theta)
  }

  # Cálculo del accuracy
  acc = 100 * length(aciertos[aciertos %in% TRUE]) / m
  print(paste(method, ' - accuracy: ', acc, '%', separator=''))

  # Gráfico
  distribution_name = if (distribution == 'normal') 'N(0,1))' else 'exp(1)'
  plot_intervals(intervalos, aciertos, theta,
                 method, distribution_name)
}
```

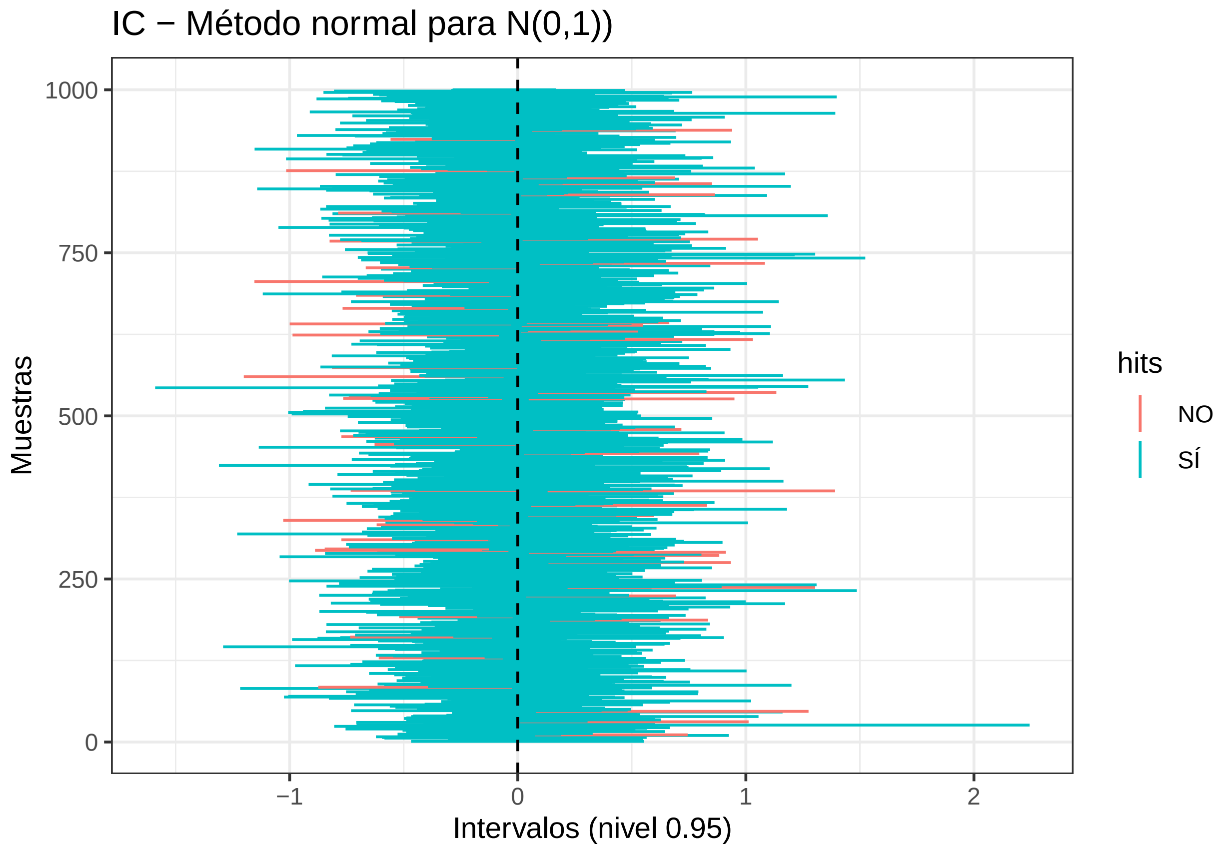
Utilizamos la función anterior para mostrar los distintos intervalos de confianza para la normal, así como su precisión.

```
set.seed(123)

methods <- c('Método híbrido', 'Método normal', 'Método percentil')
for (method in methods) {
  exercise_2('normal', method)
}
```

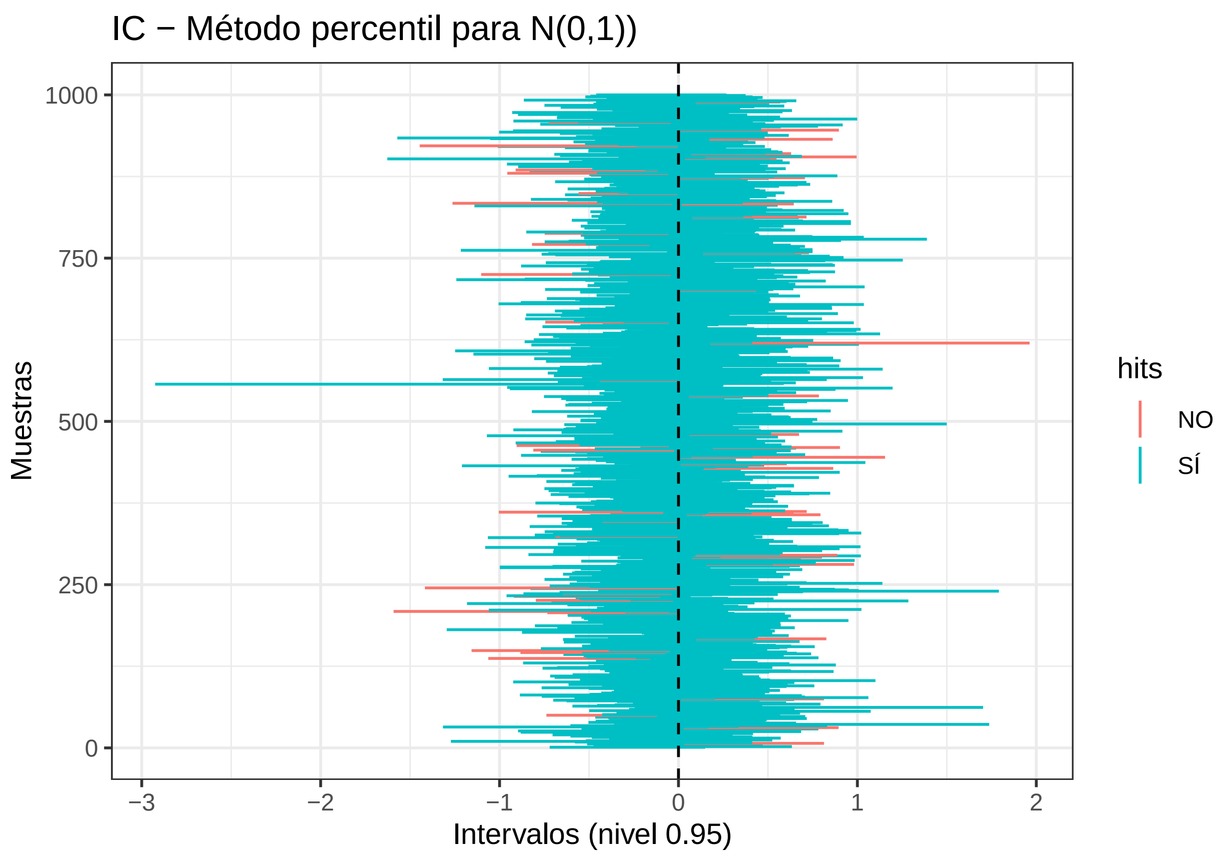
```
## [1] "Método híbrido  - accuracy:  95.2 % "
```




```
## [1] "Método normal  - accuracy:  94.8 % "
```



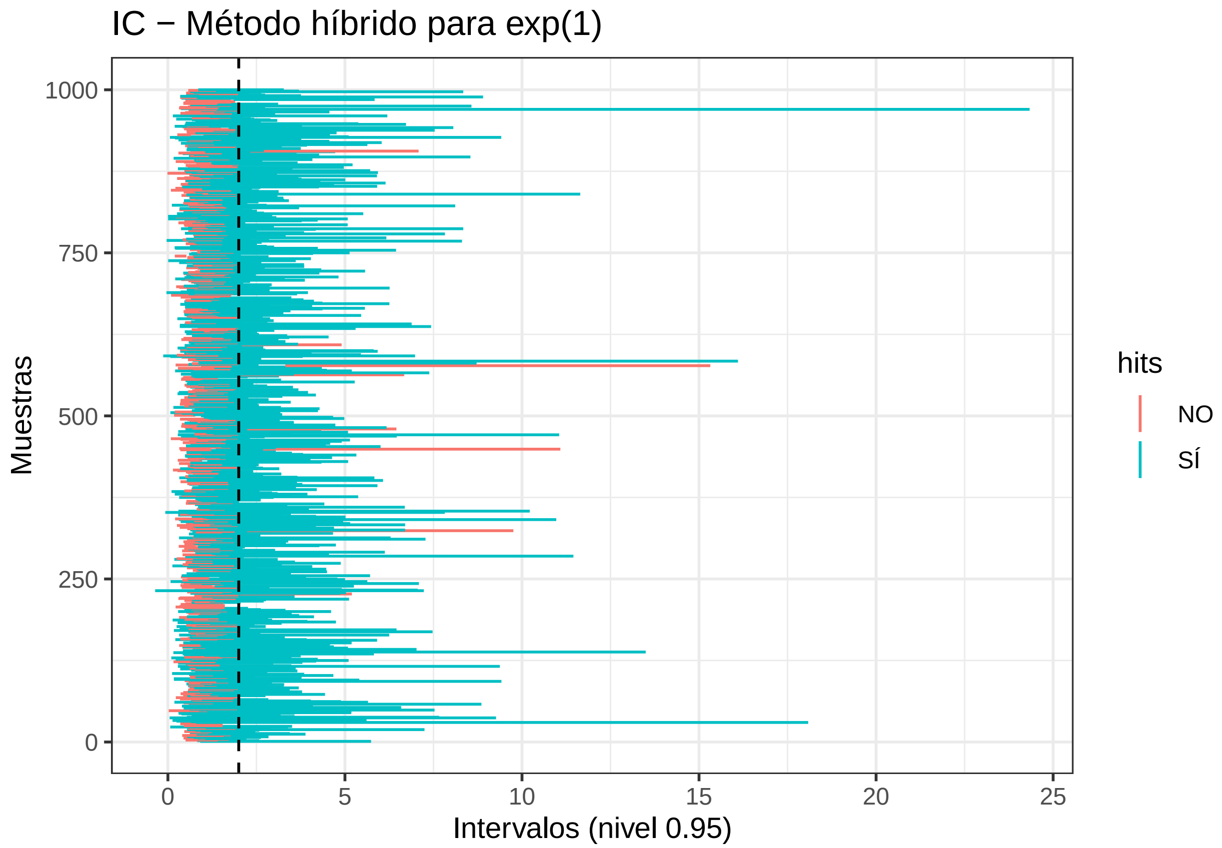

```
## [1] "Método percentil  - accuracy:  94.6 % "
```



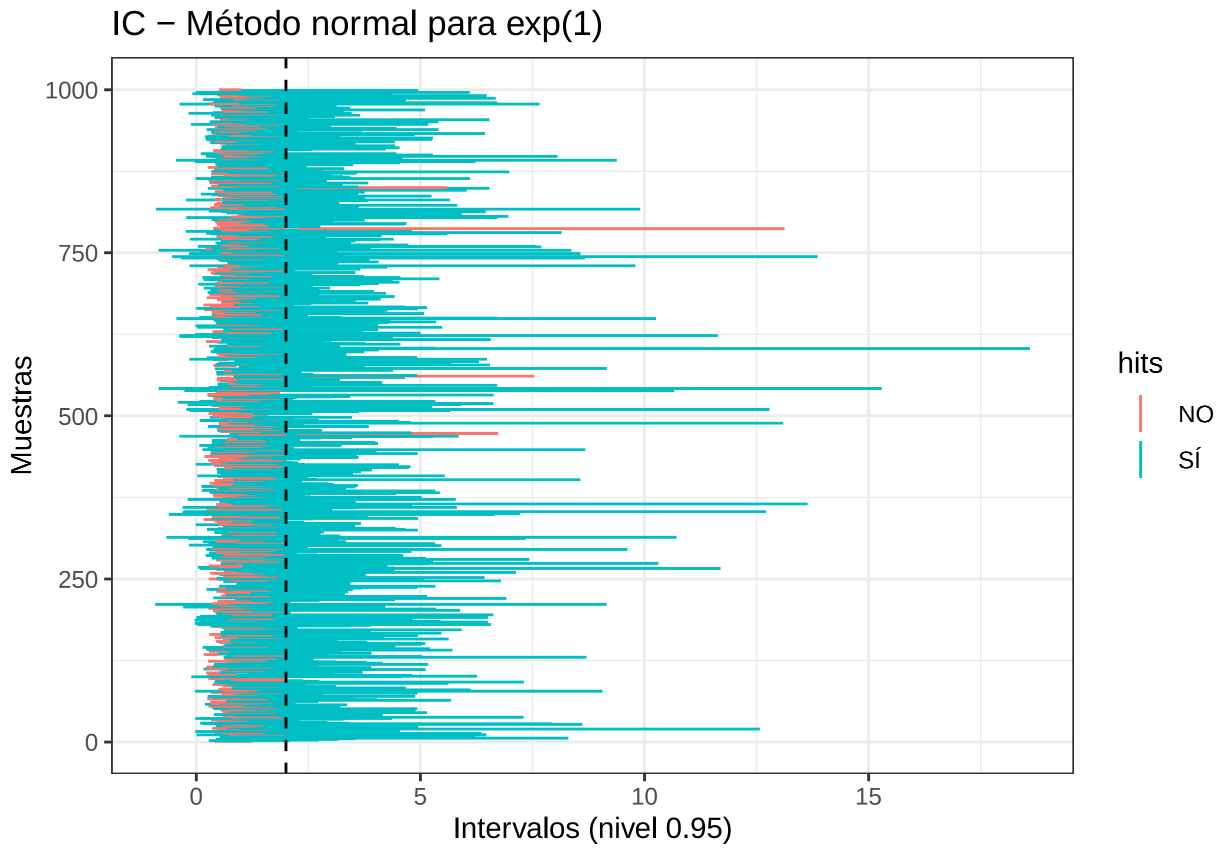
Obtenemos así precisiones cercanas al 95%. Este resultado era de esperar pues tomamos `alfa=0.05`. Repetimos el experimento para la distribución exponencial con parámetro $\lambda = 1$.

```
methods <- c('Método híbrido', 'Método normal', 'Método percentil')
for (method in methods) {
  exercise_2('exponential', method)
}
```

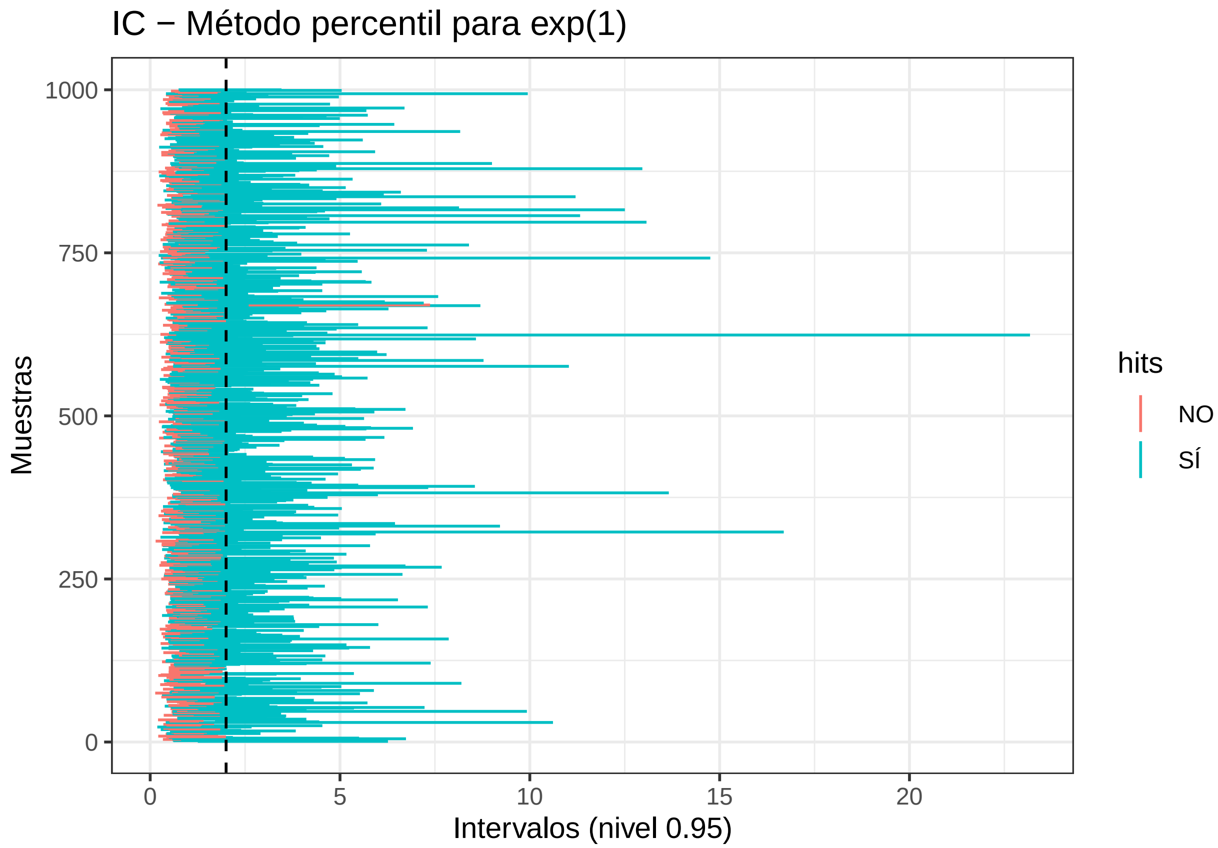
```
## [1] "Método híbrido  - accuracy:  65.3 % "
```




```
## [1] "Método normal  - accuracy:  68 % "
```




```
## [1] "Método percentil  - accuracy:  63.6 % "
```



En este caso, el coeficiente de asimetría real es 2. Sin embargo, vemos como obtenemos una precisión mucho menor: entorno al 65% en los diferentes métodos. Esto puede deberse a que el coeficiente de asimetría no se estime correctamente utilizando boostrap.