Plug-in y Bootstrap

Francisco Paz

28/8/2019

```r
library(pander) #Paquetería para sacar tablas más bonitas
library(tidyverse)
```

Plug-in y Bootstrap

Recordemos que el **plug-in** es un método para estimar parámetros a partir de muestras. La estimación de un parámetro $\theta = t(P)$ se define como:

$$\hat{\theta} = t(P_n)$$

Aquí solo estamos obteniendo una estimación y en ningun momento hablamos de presición

Vamos a generar una población utilizando la generación de números aleatorios para una normal$N(10, 2)$. Esto representa la distribución de cierta caracteríztica. Extraemos una muestra aleatoria x, y con ella queremos inferir la media de la población (En la mayoría de los problemas vamos a desconocer μ o σ y queremos encontrar su verdadero valor)

```r
set.seed(9516)
y <- rnorm(1000,10,2)
mu <- mean(y)
mu
```

```
## [1] 10.06365
```

De la forma en la que planteamos el problema, es fácil ver que el verdadero valor de $\mu = 10.06$. Ahora veamos la estimación con el principio del plug-in. Primero tomemos una muestra de y

```r
y_mues <- sample(y, size = 200, replace = FALSE)
```

Ahora, calculamos el estimador plug-in para la muestra.¿Cómo la harías?

```r
mean(y_mues)
```

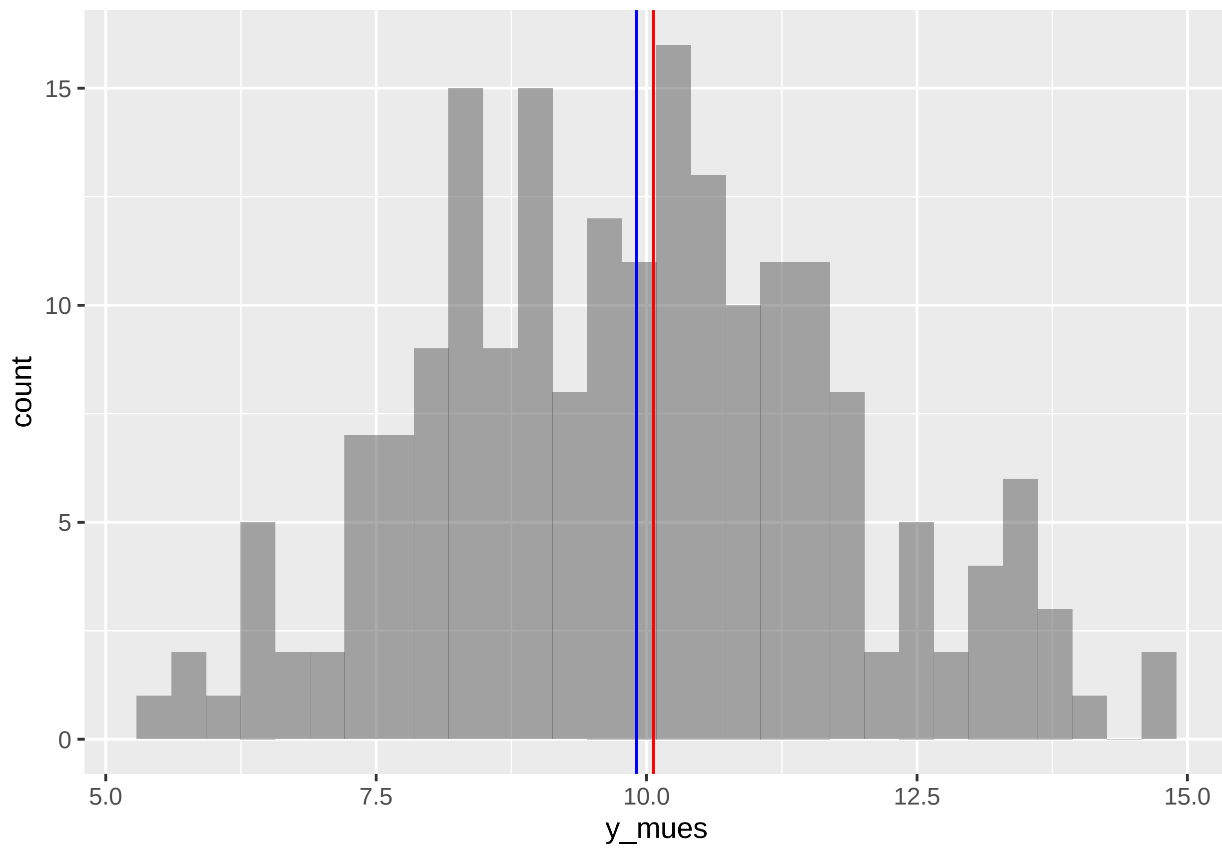
```
## [1] 9.907772
```

Está es la estimación mediante plug-in. Podemos observar que existe una diferencia entre la media de y y $y_{muestra}$.

¿cómo se relaciona esto con bootstrap?

Para empezar a generar intuición acerca de lo que estamos viendo, veamos una gráfica de nuestra muestra.

```r
y_datos <- as.data.frame(y_mues)
ggplot(y_datos, aes(x = y_mues )) +
  geom_histogram(alpha = 0.5) +
  geom_vline(xintercept = mu, col = 'red') +
  geom_vline(xintercept = mean(y_mues), col ='blue')
```



Hagamos una función que nos ayude a calcular.

```
Bootstrap <- function(x) {
  n <-length(x)
  muestra_boot <- sample(x, size = n, replace = TRUE)
  muestra_boot
}
```

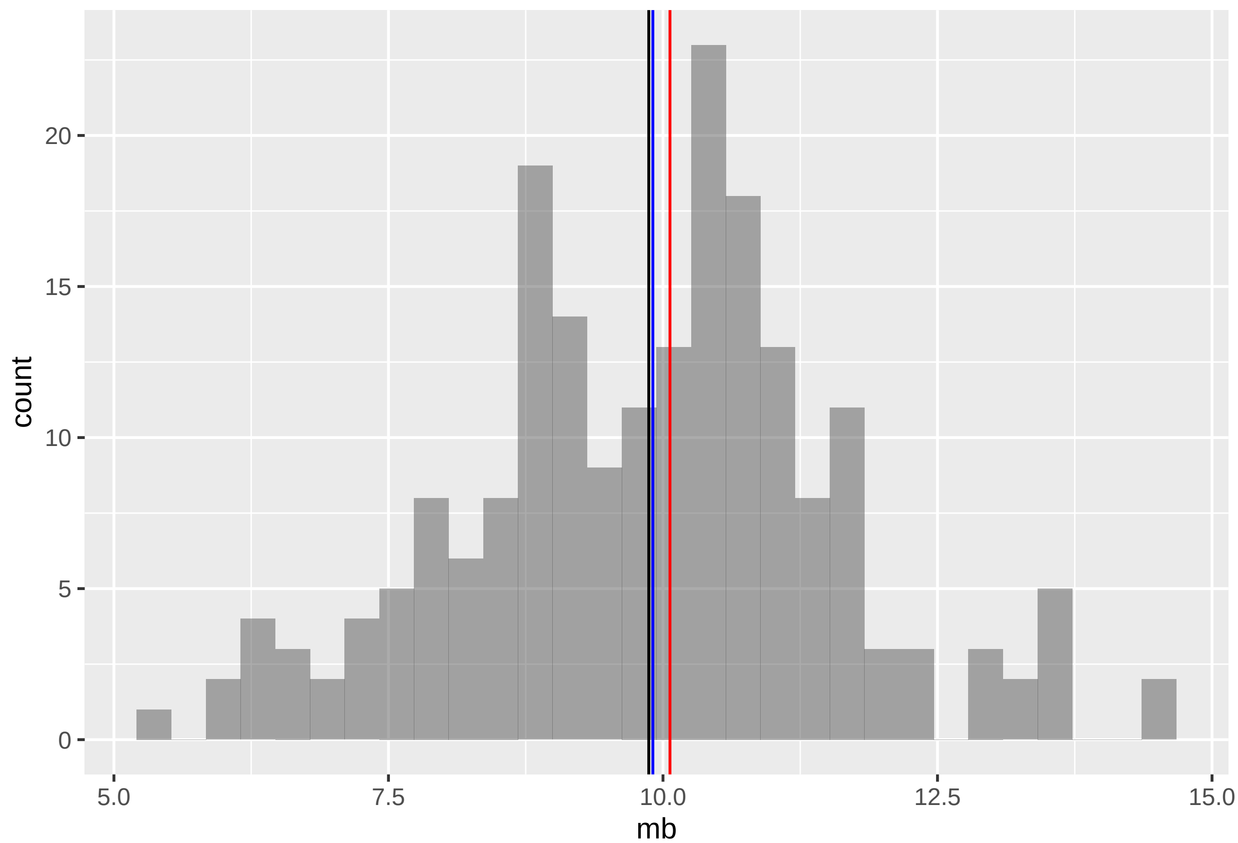
Veamos lo que ocurre con **una** muestra bootstrap

```
mb <- Bootstrap(y_mues)
mean(mb)
```

```
## [1] 9.87051
```

¿Cuál es el significado de este valor?

```
mb_datos <- as.data.frame(mb)
ggplot(mb_datos,aes(x = mb)) +
  geom_histogram(alpha = 0.5) +
  geom_vline(xintercept = mu, col='red')+
  geom_vline(xintercept = mean(y_mues), col ='blue')+
  geom_vline(xintercept = mean(mb), col = 'black')
```



Para lo siguiente, redefinitemos la función que calcula bootstrap

```
Bootstrap <- function(x) {
  n <-length(x)
  muestra_boot <- sample(x, size = n, replace = TRUE)
  mean(muestra_boot)
}
```

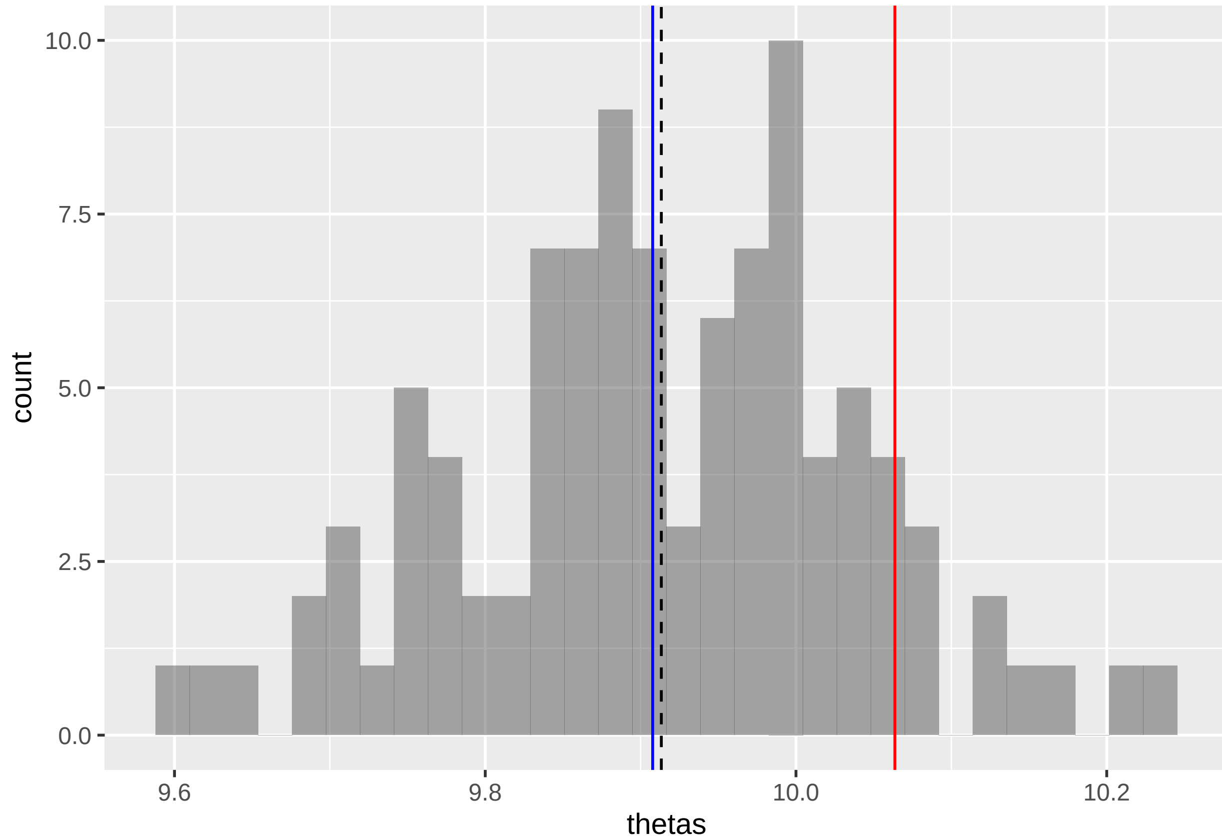
Calculamos **B** muestras bootstrap

```
B <- 100
thetas <- rerun(B,Bootstrap(y_mues)) %>% flatten_dbl()
mean(thetas)
```

```
## [1] 9.913347
```

Y ahora si, quiero cuantificar la incertidumbre. Se presenta el histograma de las thetas, que recordemos estos son las medias para cada bootstrap

```
t_datos <- as.data.frame(thetas)
ggplot(t_datos, aes(x = thetas)) +
  geom_histogram(alpha = 0.5)+
  geom_vline(xintercept = mu, col ='red')+
  geom_vline(xintercept = mean(y_mues), col= 'blue')+
  geom_vline(xintercept = mean(thetas),col = 'black',
             linetype = "dashed")
```



¿Qué fue lo que ganamos?

Definimos la función de error estandar.

```
se <- function(x) sqrt(sum((x - mean(x)) ^ 2)) / length(x)
```

```
se(thetas)
```

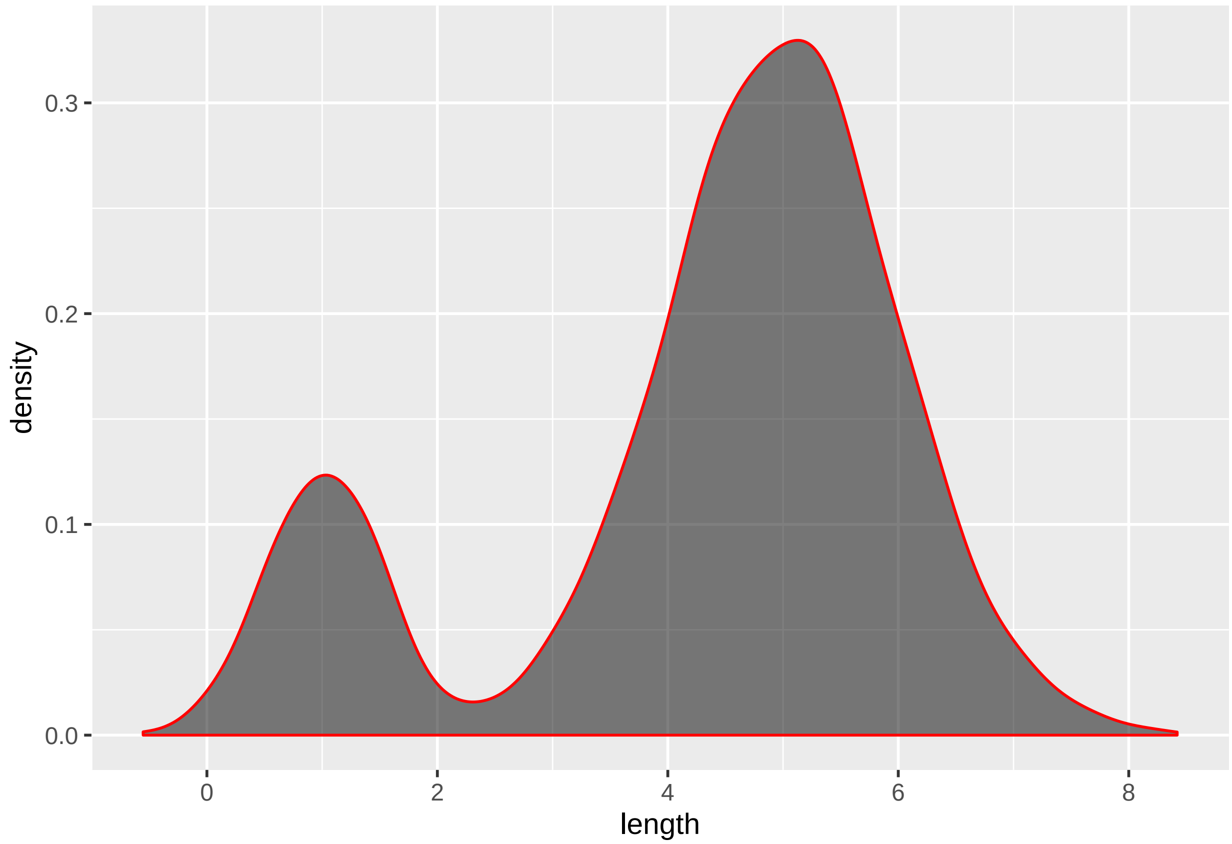
```
## [1] 0.01261552
```

Veamos un ejemplo un poco más complejo.

```
set.seed(5865)
a <- data.frame(length = rnorm(1000, 1, 0.5))
b <- data.frame(length = rnorm(5000, 5, 1))

a$pob <- 'a'
b$pob <- 'b'

datos <- as.data.frame(rbind(a, b))
ggplot(datos,aes(x = length)) + geom_density(col='red',fill = 'black', alpha = 0.5)
```




```r
mean(datos$length)
```

```
## [1] 4.33122
```

La siguiente función nos da una estimación(plug-in) del error estandar para la media:

```r
se <- function(x) sqrt(sum((x - mean(x)) ^ 2)) / length(x)
se(datos$length)
```

```
## [1] 0.02261769
```

```r
sd(datos$length)
```

```
## [1] 1.752105
```

¿Cuál es la diferencia entre error estandar y desviación estandar?

A continuación se presentaran ejemplos acerca del comportamiento del bootstrap no paramétrico

Pensemmos en una primera muestra de la distribución antes planteada

Aquí vamos

```r
B <- 5000
thetas <- rerun(B,Bootstrap(datos$length)) %>% flatten_dbl()
mean(thetas)
```

```
## [1] 4.331096
```

```r
se(thetas)
```

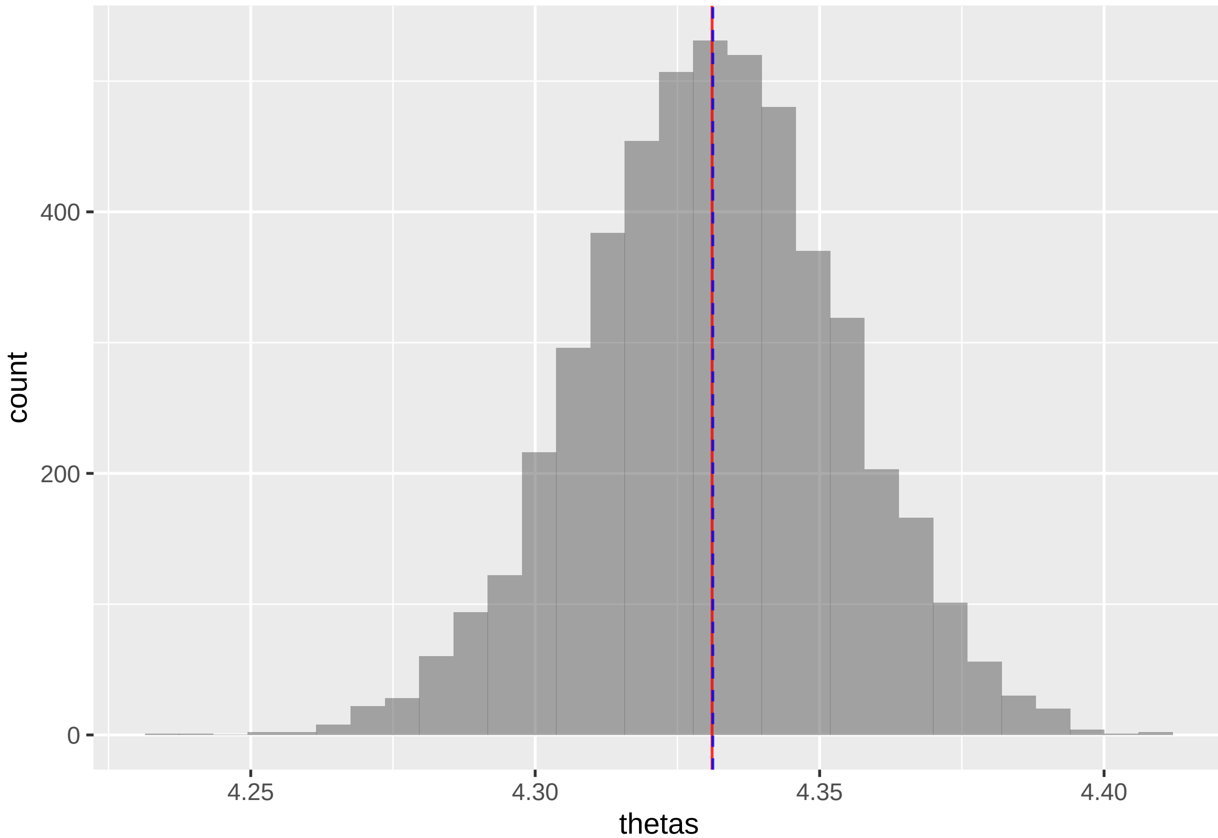
```
## [1] 0.0003198712
```

```r
sd(thetas)
```

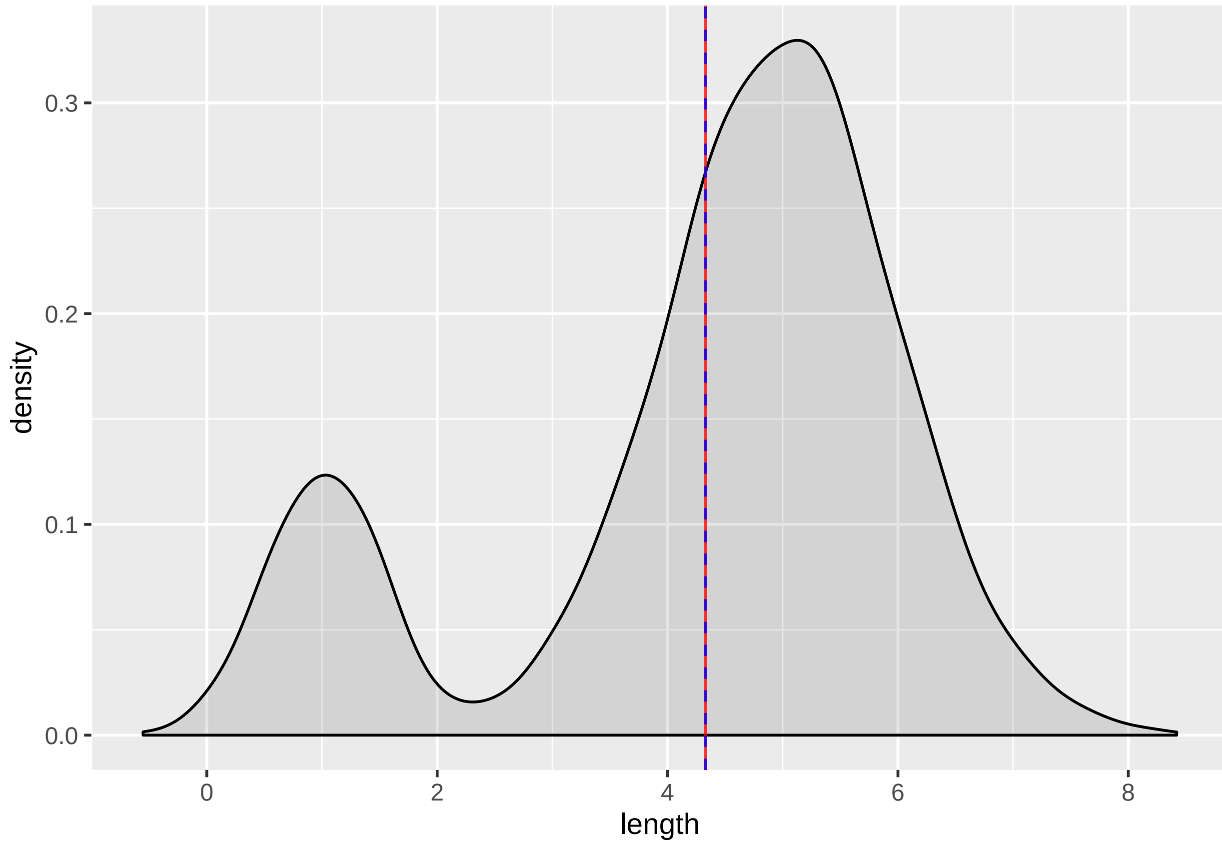
```
## [1] 0.02262057
```

Ahora veamos que sucede con todos los casos combinaddos:

```r
th <- as.data.frame(thetas)
ggplot(th) + geom_histogram(aes(thetas),alpha = 0.5) +
  geom_vline(xintercept = mean(thetas), col = 'red',alpha = 0.8) +
  geom_vline(xintercept = mean(datos$length),  color = "blue",
             linetype = "dashed",alpha = 0.8)
```




```r
ggplot(datos,aes(x = length)) +
  geom_density(col='black',fill = 'black', alpha = 0.1) +
  geom_vline(xintercept = mean(thetas), col = 'red',alpha = 0.8) +
  geom_vline(xintercept = mean(datos$length),  color = "blue",
             linetype = "dashed",alpha = 0.8)
```




```
seMediaBoot <- function(x, B){
    thetas_boot <- rerun(B, Bootstrap(x)) %>% flatten_dbl()
    se(thetas_boot)
}

B_muestras <- data_frame(n_sims = c( 5, 50, 150, 400, 1000, 1500, 3000,
    5000, 10000, 20000)) %>%
    mutate(est = map_dbl(n_sims, ~seMediaBoot(x = datos$length, B = .)))
B_muestras
```

```
## # A tibble: 10 x 2
##    n_sims      est
##     <dbl>    <dbl>
##  1      5 0.0102
##  2     50 0.00294
##  3    150 0.00189
##  4    400 0.00119
##  5   1000 0.000713
##  6   1500 0.000599
##  7   3000 0.000417
##  8   5000 0.000321
##  9  10000 0.000227
## 10  20000 0.000159
```

Aquí estamos viendo como se reduce, es decir, se reduce la desviación estandar de la distibución bootstrap. Veamos que sucede con la desviación estandar

```r
seMediaBoot <- function(x, B){
    thetas_boot <- rerun(B, Bootstrap(x)) %>% flatten_dbl()
    sd(thetas_boot)
}

B_muestras <- data_frame(n_sims = c( 5, 50, 150, 400, 1000, 1500, 3000,
    5000, 10000, 20000)) %>%
    mutate(est = map_dbl(n_sims, ~seMediaBoot(x = datos$length, B = .)))
B_muestras
```

```
## # A tibble: 10 x 2
##    n_sims    est
##     <dbl>  <dbl>
##  1      5 0.0217
##  2     50 0.0219
##  3    150 0.0232
##  4    400 0.0224
##  5   1000 0.0229
##  6   1500 0.0230
##  7   3000 0.0226
##  8   5000 0.0226
##  9  10000 0.0226
## 10  20000 0.0226
```