014 - Bootstrap Confidence Intervals

EPIB 607

Sahir Rai Bhatnagar
Department of Epidemiology, Biostatistics, and Occupational Health
McGill University

sahir.bhatnagar@mcgill.ca

slides compiled on September 26, 2021

Review of Confidence Intervals

The Bootstrap

Sampling Distribution

Definition 1 (Sampling Distribution).

- The sampling distribution of a statistic is the distribution of values taken by the statistic in **all possible samples of the same size** from the same population.
- The standard deviation of a sampling distribution is called a **standard error**

Sampling Distributions

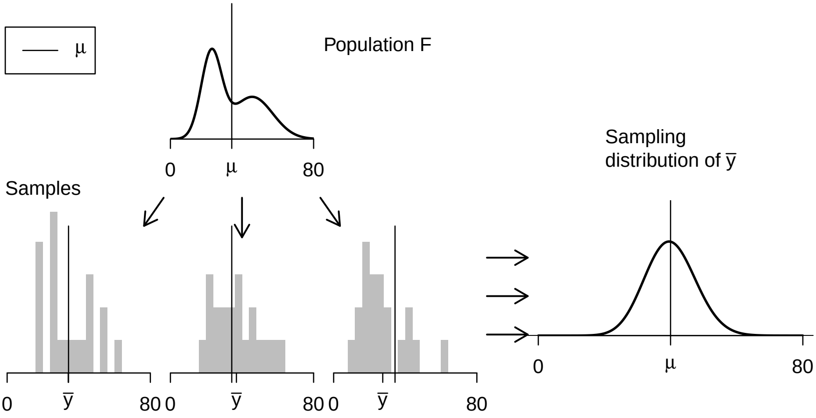


Figure: Ideal world. Sampling distributions are obtained by drawing repeated samples from the population, computing the statistic of interest for each, and collecting (an infinite number of) those statistics as the sampling distribution

Sampling Distribution

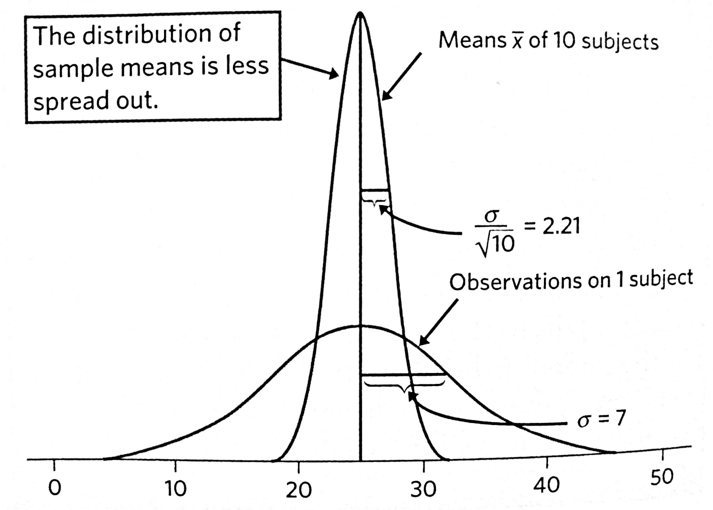


Figure: Averages are less variable than individual observations

Traditional way to calculate CIs

How to construct a CI for the population mean?

- The **CLT** gives us that $\bar{y} \sim \mathcal{N}(\mu, \sigma/\sqrt{n})$ is approximately true when n is large.
- We can standardize, to get $Z = \frac{\bar{y}-\mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0, 1)$.
- To find a CI with confidence level $\mathcal{C} = 1 - \alpha$, we must calculate the critical value z^* such that

$$P(-z^* < Z < z^*) = \mathcal{C} = 1 - \alpha \tag{1}$$

  where α is the significance level
  - That is, we want the value z^* that gives a *lower tail probability* of $(1 - \mathcal{C})/2 = \alpha/2$.
  - Often this value is denoted $z^* = z_{\alpha/2}$; thus we have

$$P(Z < -z_{\alpha/2}) = \alpha/2,$$

    and

$$P(Z > z_{\alpha/2}) = \alpha/2.$$

Traditional way to calculate CIs

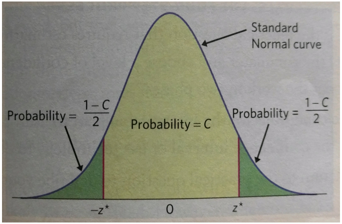


Figure: The critical value $z^\star$ is the number that catches central probability $\mathcal{C}$ under a standard normal $\mathcal{N}(0,1)$ curve between $-z^\star$ and $z^\star$

We can use this probability statement about the standardized version of the sample mean $(\bar{y}-\mu)/\sigma/\sqrt{n}$, to place bounds on where we think the true mean lies by examining the probability that $\bar{y}$ is within $z^\star \cdot \frac{\sigma}{\sqrt{n}}$ of μ

$$
\begin{aligned}
\mathcal{C} &= P\left(-z^\star \leq \frac{\bar{y}-\mu}{\sigma/\sqrt{n}} \leq z^\star\right) \\
&= P\left(-z^\star \frac{\sigma}{\sqrt{n}} \leq \bar{y}-\mu \leq +z^\star \frac{\sigma}{\sqrt{n}}\right) \\
&= P\left(-\bar{y}-z^\star \frac{\sigma}{\sqrt{n}} \leq -\mu \leq -\bar{y}+z^\star \frac{\sigma}{\sqrt{n}}\right) \\
&= P\left(\bar{y}+z^\star \frac{\sigma}{\sqrt{n}} \geq \mu \geq \bar{y}-z^\star \frac{\sigma}{\sqrt{n}}\right) \\
&= P\left(\bar{y}-z^\star \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{y}+z^\star \frac{\sigma}{\sqrt{n}}\right) \\
&= 1-\alpha
\end{aligned}
$$

We call the interval $\left(\bar{y}-z^\star \frac{\sigma}{\sqrt{n}}, \bar{y}+z^\star \frac{\sigma}{\sqrt{n}}\right)$ a **(1-α)100% confidence interval** for μ.

Confidence intervals for depths of the ocean

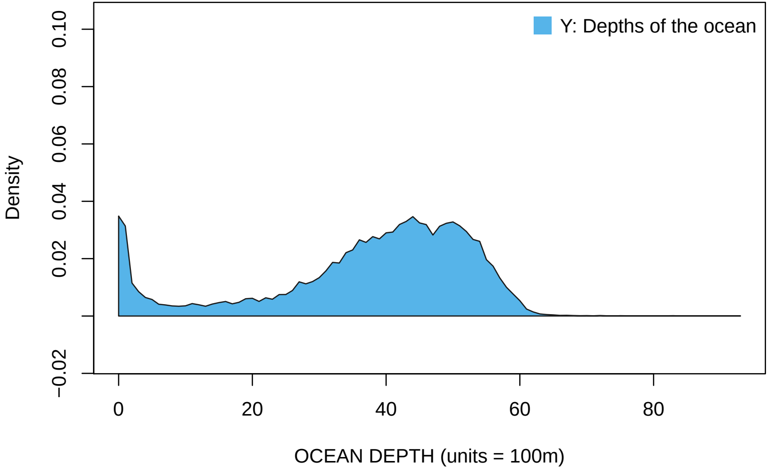


Figure: The original data distribution of sampled depths of the ocean. Note that it has multiple modes and not Normal looking.

The CLT is ’kicking in’ at $n = 16$

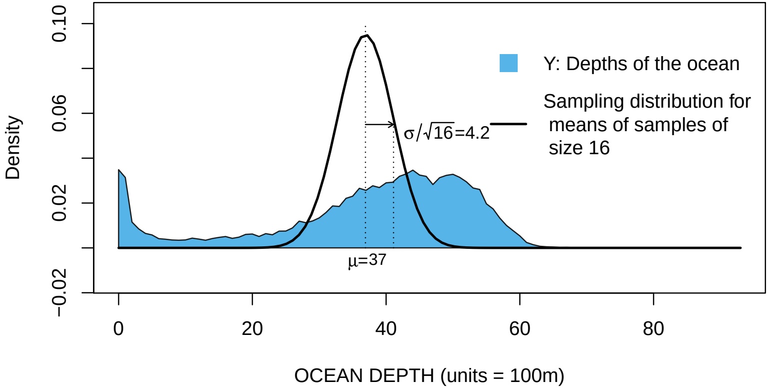


Figure: The sampling distribution for the mean depth of the ocean with samples of size $n = 16$, looks normal (centered at $\mu = 37$ and SD equal to $\sigma/\sqrt{16}$)

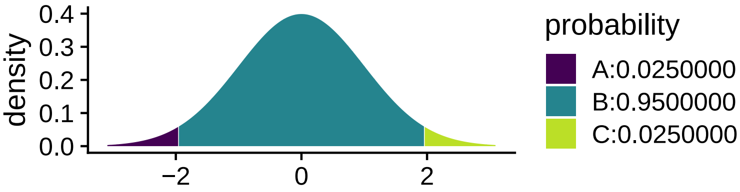
Since CLT has 'kicked in', we use it to construct a CI

We want to construct a $\mathcal{C} = 95\%$ confidence interval for the mean. Level of significance is $\alpha = 1 - \mathcal{C} = 0.05$

1. by the CLT $\rightarrow \bar{y} \sim \mathcal{N}(mean = 37, sd = \sigma/\sqrt{16} = 4.2)$

2. The critical value $z^\star$ such that $P(Z < -z^\star) = P(Z > z^\star) = \alpha/2 = 0.025$ is given by

```
mosaic::xqnorm(p = c(0.025, 0.975))
```




```
## [1] -1.959964  1.959964
```

3. 95% CI for μ: $(37 - 1.96 \cdot 4.2, 37 + 1.96 \cdot 4.2) = [29, 45]$

Alternative way of calculating CI with CLT: `qnorm`

- In the previous slides we used the standard normal $\mathcal{N}(0,1)$ to calculate the critical value $z^\star$ needed for the CI
- We were able to use the $\mathcal{N}(0,1)$ for two reasons:
    1. the CLT
    2. the formula used to calculate the CI is based on standardizing $\bar{y} \to \frac{\bar{y}-\mu}{\sigma/\sqrt{n}}$
- There is an alternative, **yet equivalent**, way to calculate the CI without standardizing $\bar{y}$, and without using the $\pm$ formula
- This is accomplished using `qnorm`
- Note: we **still need the CLT regardless** of whether we use the $\pm$ formula or `qnorm`

68% Confidence interval using `qnorm`

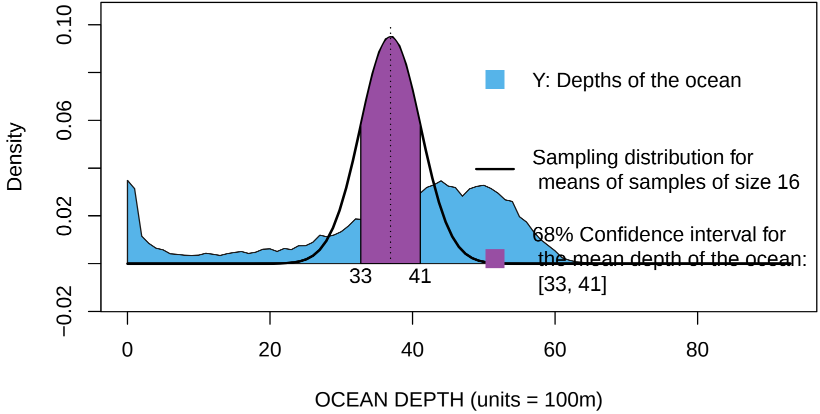


Figure: 68% Confidence interval calculated using `qnorm(p = c(0.16,0.84), mean = 37, sd = 4.2)`

95% Confidence interval using `qnorm`

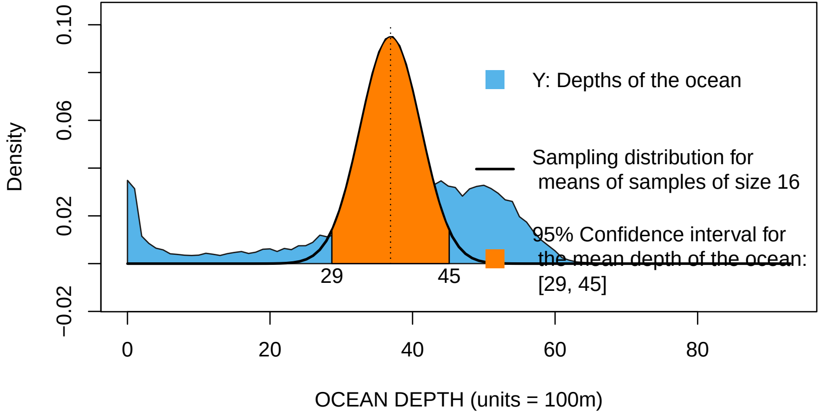


Figure: 95% Confidence interval calculated using
`qnorm(p = c(0.025,0.975), mean = 37, sd = 4.2)`

Motivation for the Bootstrap

- The $\pm$ and `qnorm` methods to calculate a CI both require the CLT

Q: What happens if the CLT hasn't 'kicked in'? Or you don't believe the CLT?

A: Bootstrap

Review of Confidence Intervals

The Bootstrap

Ideal world: known sampling distribution

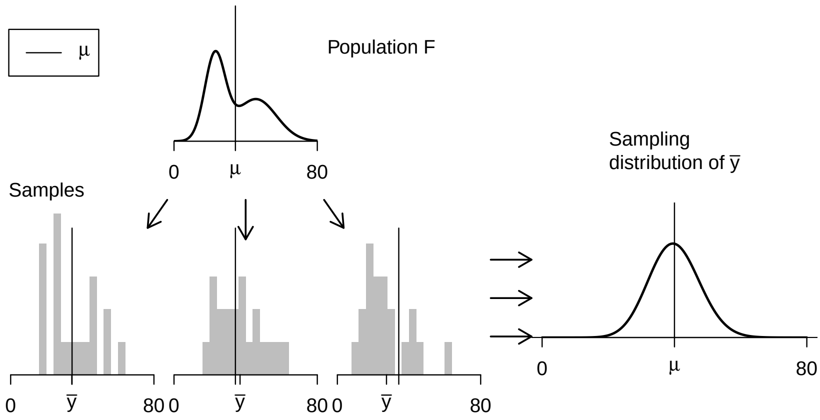


Figure: Ideal world. Sampling distributions are obtained by drawing repeated samples from the population, computing the statistic of interest for each, and collecting (an infinite number of) those statistics as the sampling distribution

Reality: use the bootstrap distribution instead

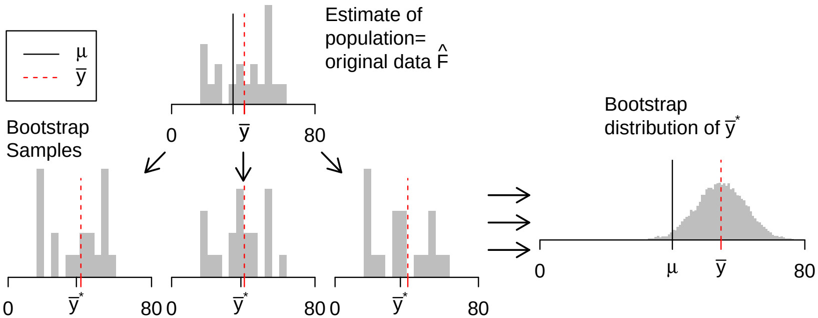
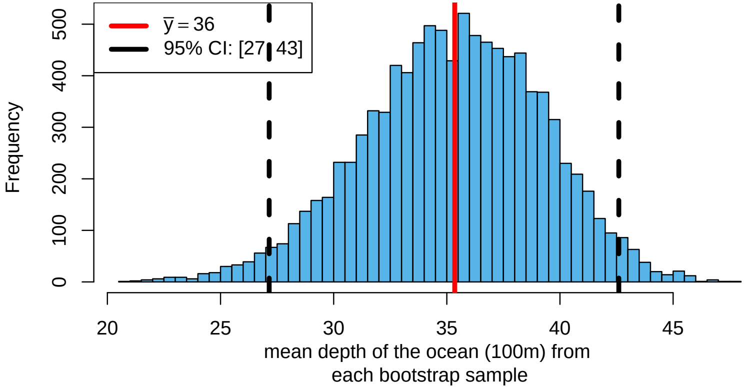


Figure: Bootstrap world. The bootstrap distribution is obtained by drawing repeated samples from an estimate of the population, computing the statistic of interest for each, and collecting those statistics. The distribution is centered at the observed statistic ($\bar{y}$), not the parameter (μ).

Main idea: simulate your own sampling distribution

```
R <- replicate(B, {
  dplyr::sample_n(depths.n.20, size = N, replace = TRUE) %>%
  dplyr::summarize(r = mean(alt)) %>%
  dplyr::pull(r)
})
CI_95 <- quantile(R, probs = c(0.025, 0.975))
```



Bootstrap code 1

```r
# function for sampling ocean depths
source("https://raw.githubusercontent.com/sahirbhatnagar/EPIB607/master/labs/
      003-ocean-depths/automate_water_task.R")

# from the in-class exercise
index.n.20 <- c(2106,2107,2108,2109,2110,2111,2112,
2113,2114,2115,2116,2117,2118,2119,
2120,2121,2122,2123,2124,2125)

# get depths of ocean sample n=20
depths.n.20 <- automate_water_task(index = index.n.20,
               student_id = 260194225, type = "depth")

# change to 100m units
depths.n.20$alt = round(depths.n.20$alt/100,0)

B <- 10000; N <- nrow(depths.n.20)
R <- replicate(B, {
dplyr::sample_n(depths.n.20, size = N, replace = TRUE) %>%
dplyr::summarize(r = mean(alt)) %>%
dplyr::pull(r)
})

CI_95 <- quantile(R, probs = c(0.025, 0.975))
```

Bootstrap code 2

```r
# plot sampling distribution
hist(R, breaks = 50, col = "#56B4E9",
main="",
xlab = "mean depth of the ocean (100m) from each bootstrap sample")

# draw red line at the sample mean
abline(v = mean_depth, lty =1, col = "red", lwd = 4)

# draw black dotted lines at 95% CI
abline(v = CI_95[1], lty =2, col = "black", lwd = 4)
abline(v = CI_95[2], lty =2, col = "black", lwd = 4)

# include legend
library(latex2exp)
legend("topleft",
legend = c(TeX("$\\bar{y} = 36$"),
sprintf("95%% CI: [%.f, %.f]",CI_95[1], CI_95[2])),
lty = c(1,1),
col = c("red","black"), lwd = 4)
```

Session Info

```
R version 4.0.4 (2021-02-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Pop!_OS 21.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.13.so

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods  
[8] base     

other attached packages:
 [1] latex2exp_0.4.0   DT_0.16           mosaic_1.7.0      Matrix_1.3-2     
 [5] mosaicData_0.20.1 ggformula_0.9.4   ggstance_0.3.4    lattice_0.20-41  
 [9] kableExtra_1.2.1  socviz_1.2        gapminder_0.3.0   here_0.1         
[13] NCStats_0.4.7     FSA_0.8.30        forcats_0.5.1     stringr_1.4.0    
[17] dplyr_1.0.7       purrr_0.3.4       readr_1.4.0       tidyr_1.1.3      
[21] tibble_3.1.3      ggplot2_3.3.5     tidyverse_1.3.0   knitr_1.33       

loaded via a namespace (and not attached):
 [1] fs_1.5.0           lubridate_1.7.9    RColorBrewer_1.1-2 webshot_0.5.2     
 [5] httr_1.4.2         rprojroot_2.0.2    backports_1.2.1    utf8_1.2.2        
 [9] R6_2.5.1           DBI_1.1.1          colorspace_2.0-2   withr_2.4.2       
[13] tidyselect_1.1.1   gridExtra_2.3      leaflet_2.0.3      curl_4.3.2        
[17] compiler_4.0.4     cli_3.0.1          rvest_1.0.0        pacman_0.5.1      
[21] xml2_1.3.2         ggdendro_0.1.22    labeling_0.4.2     mosaicCore_0.8.0  
[25] scales_1.1.1       digest_0.6.27      foreign_0.8-81     rmarkdown_2.9.7   
[29] rio_0.5.16         pkgconfig_2.0.3    htmltools_0.5.2    highr_0.9         
[33] dbplyr_1.4.4       fastmap_1.1.0      htmlwidgets_1.5.3  rlang_0.4.11      
[37] readxl_1.3.1       rstudioapi_0.13    farver_2.1.0       generics_0.1.0    
[41] jsonlite_1.7.2     crosstalk_1.1.1    zip_2.2.0          car_3.0-9         
[45] magrittr_2.0.1     Rcpp_1.0.7         munsell_0.5.0      fansi_0.5.0       
[49] abind_1.4-5        lifecycle_1.0.0    stringi_1.7.3      carData_3.0-4     
[53] MASS_7.3-53.1      plyr_1.8.6         grid_4.0.4         blob_1.2.1        
[57] ggrepel_0.8.2      crayon_1.4.1       cowplot_1.1.0      haven_2.3.1       
[61] splines_4.0.4      hms_1.0.0          pillar_1.6.2       reprex_0.3.0      
[65] glue_1.4.2         evaluate_0.14      data.table_1.14.0  modelr_0.1.8      
[69] vctrs_0.3.8        tweenr_1.0.1       cellranger_1.1.0   gtable_0.3.0      
[73] polyclip_1.10-0    assertthat_0.2.1   TeachingDemos_2.12 xfun_0.25         
[77] ggforce_0.3.2      openxlsx_4.1.5     broom_0.7.2        viridisLite_0.4.0 
```