Lecture1_Code.R

alexh

2022-05-31

```
########## Lecture1_Code.R
# Creator: Alex Hoagland
# Created: 5/30/2022
# Last modified: 5/30/2022
#
# PURPOSE:
#   Runs Monte Carlo simulations to show that estimates of ATE correctly identify ATE (on average)
#   when we assume independence
#
# NOTES:
#   - uses the Tidyverse package and Dplyr
################################################################################


##### 1. Call all relevant packages
# install.packages('tidyverse') # if needed, install the package
library(tidyverse) # call the relevant library
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.3     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

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

```
##### 2. All user-defined functions
# This whole code chunk defines a function called "gap"
gap <- function() # This line names the function "gap" and tells R that it's a function. Any arguments
{ # Always have brackets around function definitions
  sdo <-  tibble(
    y1 = c(7,5,5,7,4,10,1,5,3,9),
    y0 = c(1,6,1,8,2,1,10,6,7,8),
    random = rnorm(10)
  ) %>% # The lines above define a data frame (referred to as a "tibble") with three variables and 10 o
    arrange(random) %>% # This line sorts the data frame by the random variable (i.e., a random sort)
    mutate( # Mutate is dplyr speak for creating new variables
      d = c(rep(1,5), rep(0,5)), # Assigns a treatment variable: treated status for first 5 observation
      y = d * y1 + (1 - d) * y0 # The switching equation to tell us which outcome we observe
    ) %>%
```

```
    pull(y) # This line keeps only the observed outcome

  sdo <- mean(sdo[1:5]-sdo[6:10]) # This is the outcome of interest: the estimated ATE based on observe

  return(sdo) # This line tells the function what to return (the outcome we want)
} # Don't forget to close the brackets

# Let's test the function
gap()
```

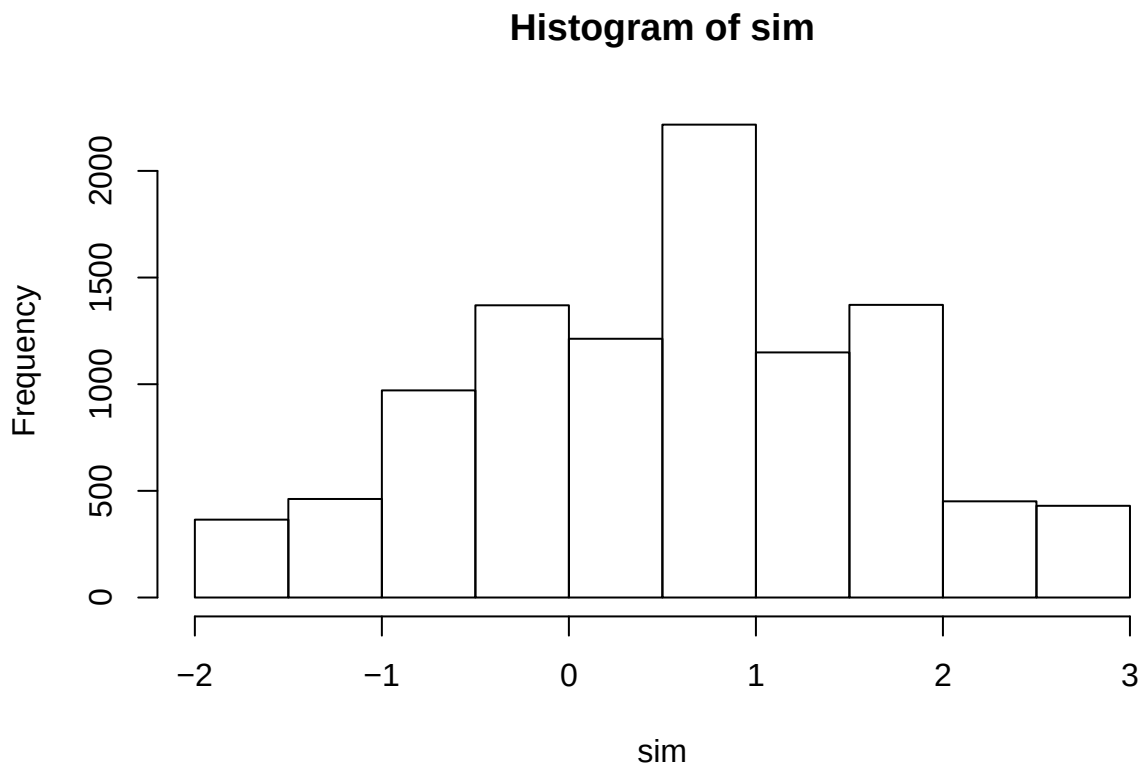
```
## [1] 2
```

```
test <- gap()


##### 3. First code chunk: monte carlo simulations
sim <- replicate(10000, gap()) # Run the gap function 10,000 times
mean(sim) # The average estimated effect is almost 0.6!
```

```
## [1] 0.63276
```

```
##### 4. Visualize the simulation results
hist(sim) # This kind of visualization is called "baseR"
```




```
# Another kind of visualization uses the "ggplot" format -- this is helpful for making nice, publicatio
sim <- tibble(sim) # We need a tibble for ggplot to work
histfig <- ggplot(data=sim, aes(sim))+geom_histogram(binwidth=.2,color='gray70',fill='cadetblue') +
```

```
  theme_minimal() + labs(x="Estimated ATE", y="",title="Estimated ATE vs. Truth") + geom_vline(xinterce
  # Notes:
  # First, need to tell ggplot what tibble object to look at (sim)
  # Then, the aes() command tells ggplot which variables to use. This depends on the particular geom()
  # Since we are making a one-variable histogram, we need only reference the variable we have (sim)
  # geom_histogram() makes the histogram.
  # See http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf for colors in R
  # theme_minimal() is a nice theme to use for the background. See themes here: https://ggplot2.tidyver
  # labs() defines the labels of interest
  # geom_vline() makes a vertical line at the true ATE (in this case, 0.6).
```