Getting to know R

EC 425/525, Lab 1 Solutions

Edward Rubin

08 April 2019

Exercises

1. Using the tools we've covered, generate a dataset $(n = 50)$ such that

$$y_i = 12 + 1.5 x_i + \varepsilon_i$$

where $x_i \sim N(3, 7)$ and $\varepsilon_i \sim N(0, 1)$.

2. Estimate the relationship via OLS using only matrix algebra. Recall

$$\hat{\beta}_{\text{OLS}} = \left(X'X\right)^{-1} X'y$$

3. ***Harder*** Write a function that estimates OLS coefficients using matrix algebra. Compare your results with the canned function from `R` (`lm`).

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Example solution: Part 1

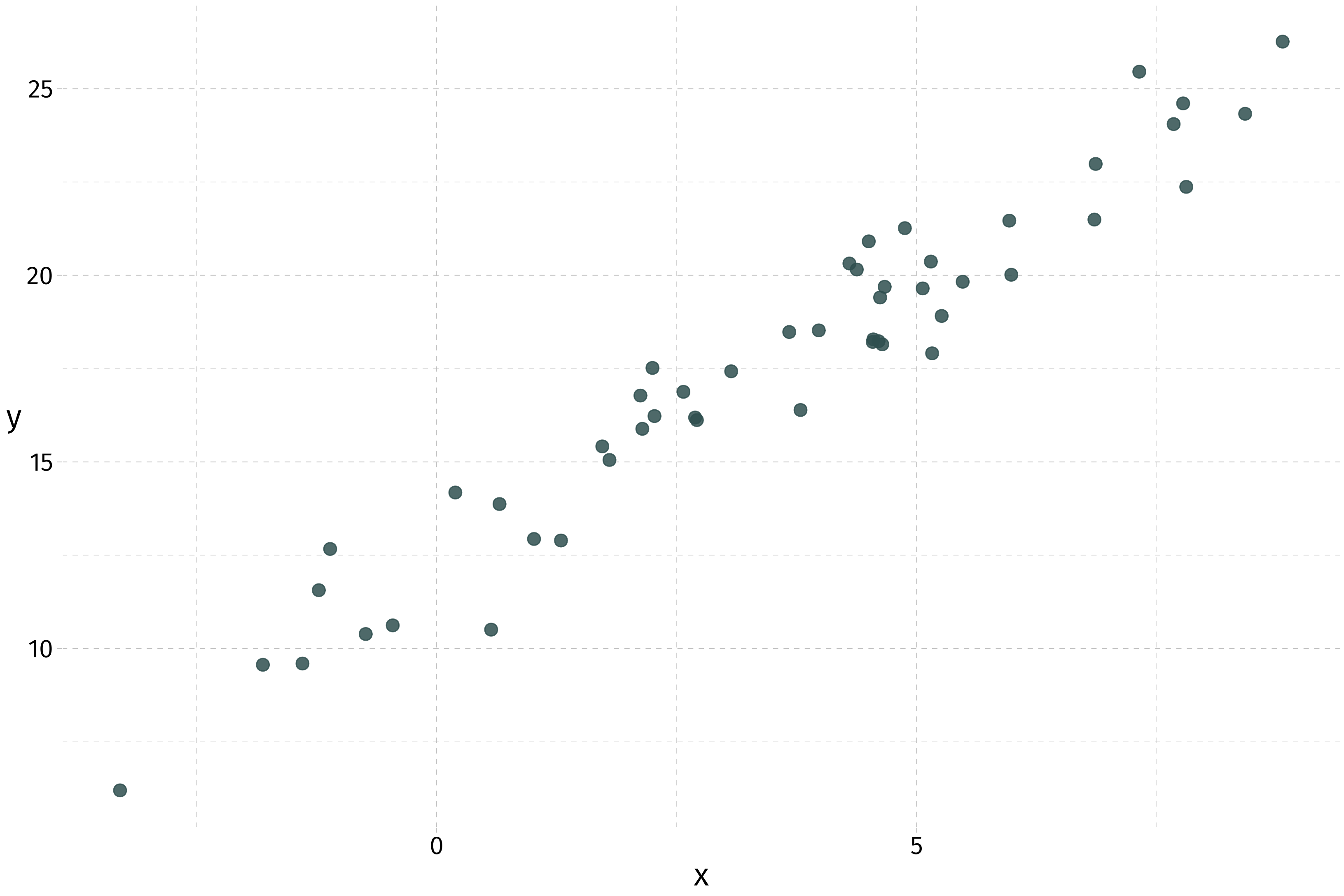
1. Using the tools we've covered, generate a dataset ($n = 50$) such that

$$y_i = 12 + 1.5x_i + \varepsilon_i$$

where $x_i \sim N(3, 7)$ and $\varepsilon_i \sim N(0, 1)$.

```
1. Set seed
2. Set sample size n=50
3. Generate x~N(3,7)
4. Generate ε~N(0,1)
5. Calculate y
   y ≔ 12 + 1.5 x + ε
```

```r
# Set seed
set.seed(12345)
# Set sample size
n ← 50
# Generate x~N(3,7)
x ← rnorm(
  n = n, mean = 3, sd = sqrt(7)
)
# Generate ε~N(0,1)
ε ← rnorm(n = n)
# Calcualte y
y ← 12 + 1.5 * x + ε
```



25
20
15
10
y
0
5
x

Example solution: Part 2

2. Estimate the relationship via OLS using only matrix algebra. Recall

$$\hat{\beta}_{\text{OLS}} = \left(X'X\right)^{-1} X'y$$

```
1. Convert y to matrix
2. Create X matrix: [1 x]
3. OLS matrix math
```

```r
# Convert y to matrix
y_m <- as.matrix(y)
# Create X matrix
X_m <- cbind(1, x)
# Matrix math
XX <- t(X_m) %*% X_m
Xy <- t(X_m) %*% y_m
b_ols <- solve(XX) %*% Xy
```

- `cbind` is *column-binding* its arguments (`1` and `x`).
- *Alternatives:*
  - `matrix(data = c(rep(1, n), x), ncol = 2, byrow = F)`
  - `as.matrix(data.frame(1, x))`

Example solution: Part 2

2. Estimate the relationship via OLS using only matrix algebra. Recall

$$\hat{\beta}_{\text{OLS}} = \left(X'X\right)^{-1} X'y$$

How did we do?

Our estimates:

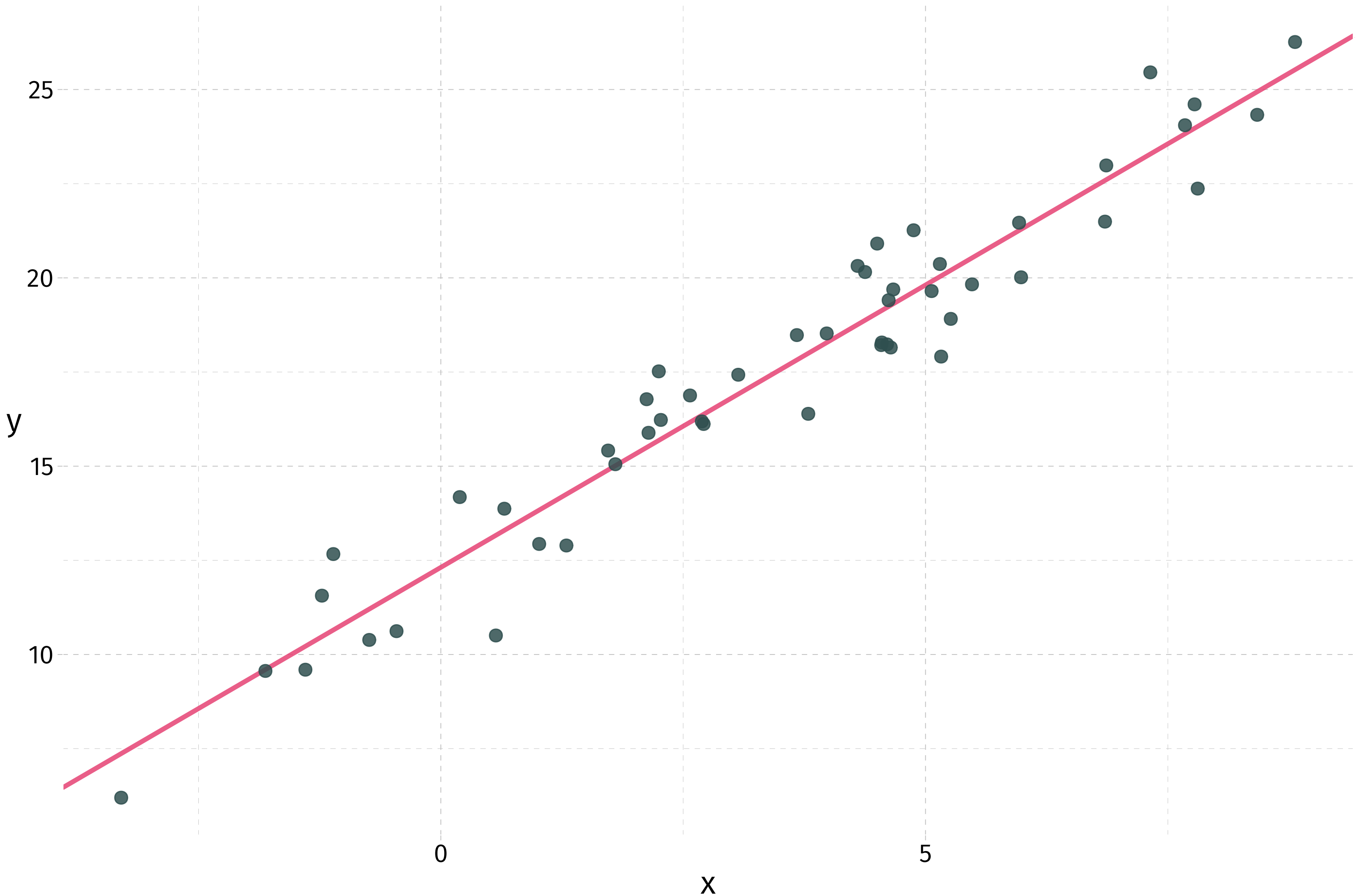
```
b_ols
```

```
#>          [,1]
#>    12.313159
#> x   1.499329
```

R's estimates:

```
lm(y ~ x)
```

```
#>
#> Call:
#> lm(formula = y ~ x)
#>
#> Coefficients:
#> (Intercept)            x
#>      12.313        1.499
```


25
20
15
10
y
0
5
x

Example solution: Part 3

3. ***Harder*** Write a function that estimates OLS coefficients using matrix algebra. Compare your results with the canned function from `R` (`lm`).

```
1. Convert data to matrix
2. Optional: Add a column of 1s
3. Calculate: (X'X)^-1(X'y)
```

Our function should take arguments

- `y` (the outcome matrix)
- `x` (covariates)
- an optional argument for whether we add an intercept ot `x`

Example solution: Part 3

3. ***Harder*** Write a function that estimates OLS coefficients using matrix algebra. Compare your results with the canned function from `R` (`lm`).

```
1. Convert data to matrix
2. Optional: Add a column of 1s
3. Calculate: (X'X)^-1(X'y)
```

```r
b_ols <- function(y, x, add_int = F) {
  # Force 'y' to matrix
  Y <- as.matrix(y)
  # Force 'x' to matrix
  X <- as.matrix(x)
  # If desired: Add intercept
  if (add_int == T) X <- cbind(1, X)
  # Matrix math
  b <- solve(t(X) %*% X) %*% t(X) %*% y
  # Done
  return(b)
}
```

Example solution: Part 3

3. ***Harder*** Write a function that estimates OLS coefficients using matrix algebra. Compare your results with the canned function from `R` (`lm`).

```
1. Convert data to matrix
2. Optional: Add a column of 1s
3. Calculate: (X'X)^-1 (X'y)
```

Us

```r
b_ols(y = y, x = x, add_int = T)
```

```
#>              [,1]
#> [1,] 12.313159
#> [2,]  1.499329
```

Canned `R`

```r
lm(y ~ x)
```

```
#>
#> Call:
#> lm(formula = y ~ x)
#>
#> Coefficients:
#> (Intercept)            x
#>      12.313        1.499
```

Example solution: Part 4

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Simulation outline

```
One iteration:

 1. Generate data via DGP (x, ε, and y)
 2. Estimate OLS coefficients

Repeat n=10,000 times...
```

Example solution: Part 4

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Let's write a function for one iteration

```
one_iter <- function(iter, b0, b1, n) {
  # Generate x~N(3,7)
  x <- rnorm(n = n, mean = 3, sd = sqrt(7))
  # Generate ε~N(0,1)
  ε <- rnorm(n = n)
  # Calcualte y
  y <- b0 + b1 * x + ε
  # Regress y and x with our function
  b_est <- b_ols(y = y, x = x, add_int = T)
  # Include iteration and convert to vector
  b_est <- c(iter, b_est)
  # Return
  return(b_est)
}
```

Example solution: Part 4

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Now we run the function 10,000 times...[†]

```r
library(parallel)
# Run the simulation (parallelized)
sim_list ← mclapply(
  # The function we want to 'repeat'
  FUN = one_iter,
  # The values we want to use/vary
  X = 1:1e4,
  # Number of cores
  mc.cores = 4,
  # Other arguments/parameters for 'one_iter'
  b0 = 12, b1 = 1.5, n = 50
)
```

† We'll talk more about `lapply` and parallelization in the future.

Example solution: Part 4

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Q What does our `list` named `sim_list` look like?

```
# First element
sim_list[[1]]
```

```
#> [1]  1.000000 12.117939  1.468596
```

```
# Last element
tail(sim_list, 1)
```

```
#> [[1]]
#> [1] 10000.000000    12.217868     1.500868
```

A It's made up of each iteration's vector of results.

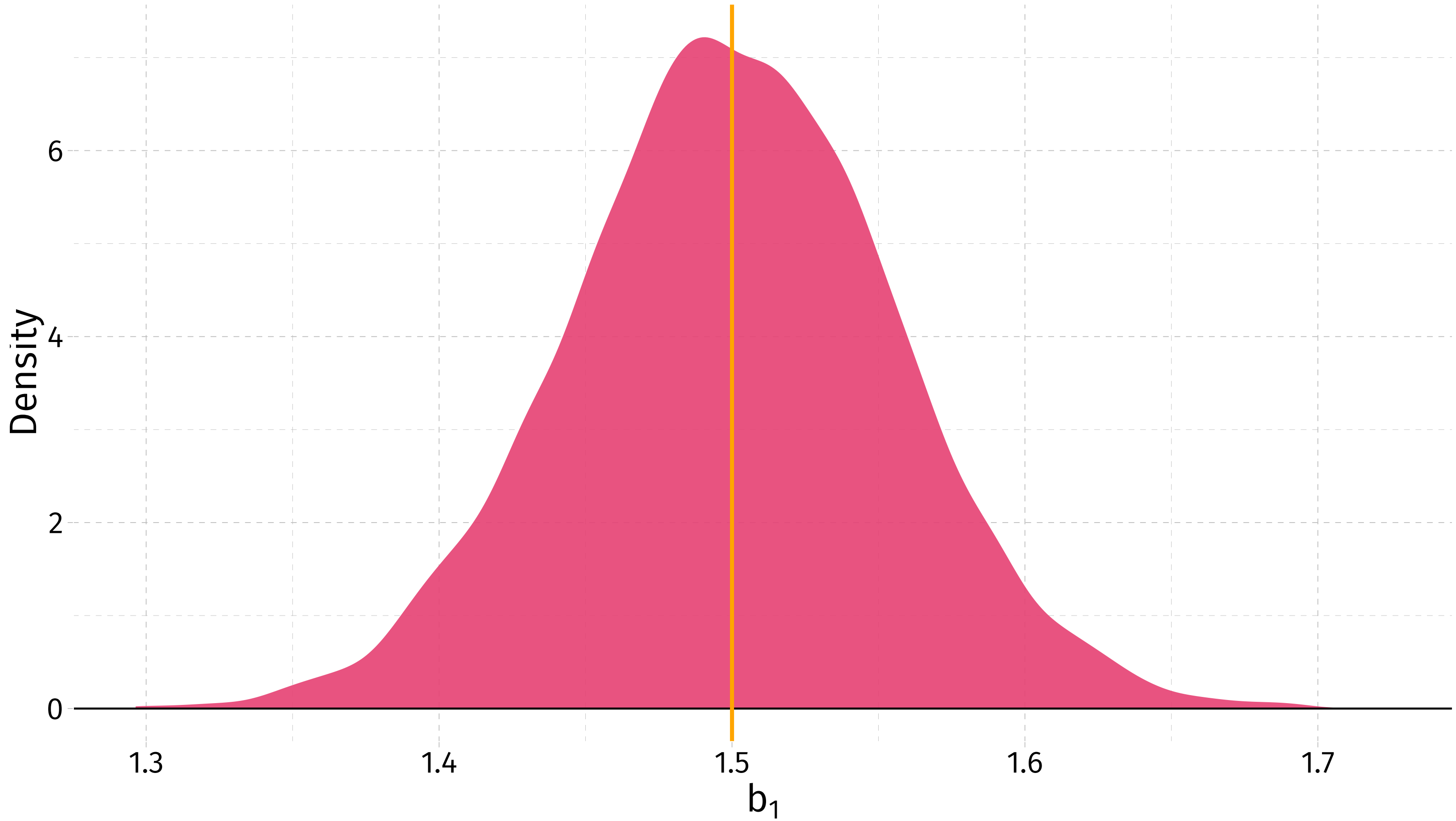
Example solution: Part 4

4. ***Hardest*** Bring it all together: Use your DGP (1) and function (3) to run a simulation that illustrates the unbiasedness of OLS.

Let's bind the individual vectors together into a single data frame.

```r
# Bind together the vectors (outputs matrix)
sim_df <- do.call("rbind", sim_list)
# Covert to data frame
sim_df <- data.frame(sim_df)
# Name our columns
names(sim_df) <- c("iter", "b0", "b1")
```

Density of our estimates for β_1 via OLS (mean $\hat{\beta}_1 = 1.5$; $\beta_1 = 1.5$)



Q Does this simulation tell us about *consistency* or *unbiasedness*?

R code from the density plot

```
ggplot(data = sim_df, aes(x = b1)) +
  geom_density(color = NA, fill = red_pink, alpha = 0.9) +
  ylab("Density") +
  xlab(expression(b[1])) +
  geom_hline(yintercept = 0, color = "black") +
  geom_vline(xintercept = 1.5, size = 1, linetype = "solid", color = orange) +
  theme_pander(base_family = "Fira Sans Book", base_size = 20)
```