```
my_sum <- function(x,y) {
        return (x + y)

}
```

```
my_sum <- function(x,y){
        x + y

}
```

```
my_sum(1,2)
```

output: 3

## Question 18

$$Y_i = 1 + X_i + \varepsilon_i \qquad \varepsilon_i \sim N(0, \sigma^2)$$

$$X_i \sim \text{Uniform}(0, 1)$$

want to calculate a 95% CI for $\beta_1$ $(\beta_1 = 1)$

Given observed data $\underbrace{(X_1, Y_1), \dots, (X_n, Y_n)}_{\text{come from the model}}, \underbrace{(X_{n+1}, Y_{n+1})}_{\text{outlier}}$

Q: how the outlier impacts performance of our CI?

A : How does the presence of an outlier impact our ability to estimate $\beta_1$?

D : (next Slide)

E : $\beta_1$

M : Fit a linear regression model with X as the explanatory variable, Y as the response using the lm function in R. Then calculate a 95% CI for $\beta_1$

P : Coverage of 95% CIs for $\beta_1$ (Compare coverage as we manipulate the outlier)

Data generating:

$(X_1, Y_1)$, $\ldots\ldots$, $(X_n, Y_n)$ $\Leftarrow$

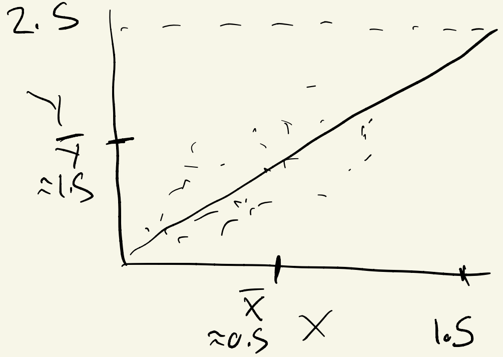
$$\begin{cases} X_i \sim \text{Uniform}(0,1) \\ Y_i = 1 + X_i + \varepsilon_i \\ \quad (\beta_0 = 1, \beta_1 = 1) \\ n = 10 \text{ or } 20,\ 100,\ 1000 \end{cases}$$

$\varepsilon_i \sim N(0, \sigma^2)$

create outlier: $(X_{n+1}, Y_{n+1}) =$

Possible values of $X_{n+1}$: 0.5, 0.75, 1, 1.5

Possible values of $Y_{n+1}$: 1.5, 2.5, 5, -1



Look at all possible combinations of
$n$, $X_{n+1}$, $Y_{n+1}$

(in this example: 4×4×4 = 64 different combos)

# Probability simulations: Advice

- Start with a plan for what you want to simulate
    - what event do you want the probability of?
    - what steps are involved in the process?
      (e.g. in theater problem:
        - handle person 1
        - handle persons 2-99
        - handle person 100)
    - How might you represent this in code? what are the main building blocks?
      (e.g. if I need to choose a random seat:
        - vector? → need to know which seats are available
        - sample? → need a way to make a random choice)

- Estimating a probability empirically: repeat process many times
    - specify # of repetitions
    - iterate (for loop, usually)
    - store the results

Usual structure:

```
set.seed(...)
nsim <- ...

results <- ....

for (i in 1:nsim) {
    # initialize the game (process, etc.   (e.g. make all seats
                                                  empty))
    # run the process
    # check whether the event happened   (e.g. was the final
                                                seat free?))
    # store the result
}

# report the estimated probability
```