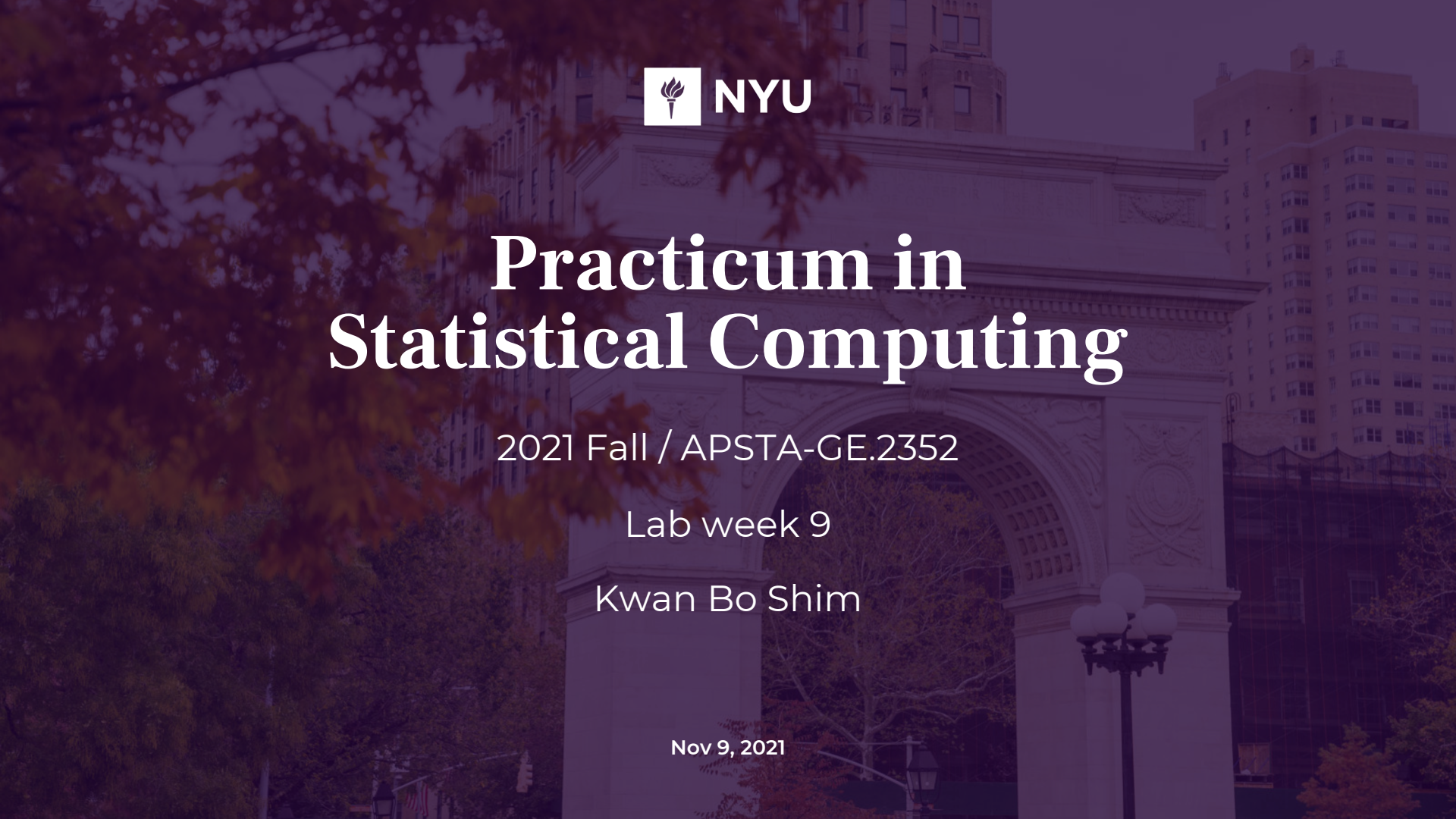


NYU

Practicum in Statistical Computing

2021 Fall / APSTA-GE.2352

Lab week 9

Kwan Bo Shim

Nov 9, 2021

PART 01

Week 9

PART 01

Week 9

1. Poll -> end of the class
2. Attendance
3. For-loop exercise solution

NYU

PART 01

Week 9

Recap

- Ordinary Least Square regression
- Using lm function
- Using optim function

NYU

Week 9

Recap

- Ordinary least squares (OLS) is the method of regression that takes as input a **response variable, y**, and a set of input **variables,** $x_1,\ldots,x_p$ for estimating the unknown parameters (coefficients on x values) in a linear regression model.
- It uses these to estimate the linear combination of those 'x's that minimizes the squared distance between the predicted and observed values.

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_p x_p + \epsilon$$

We know: **x1,x2, ... Xp**

y1, y2, ... yp

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_p x_p + \epsilon$$

We know: **x1, x2, …. Xp**
y1, y2, … yp

Want to know: **all betas** (unknown)

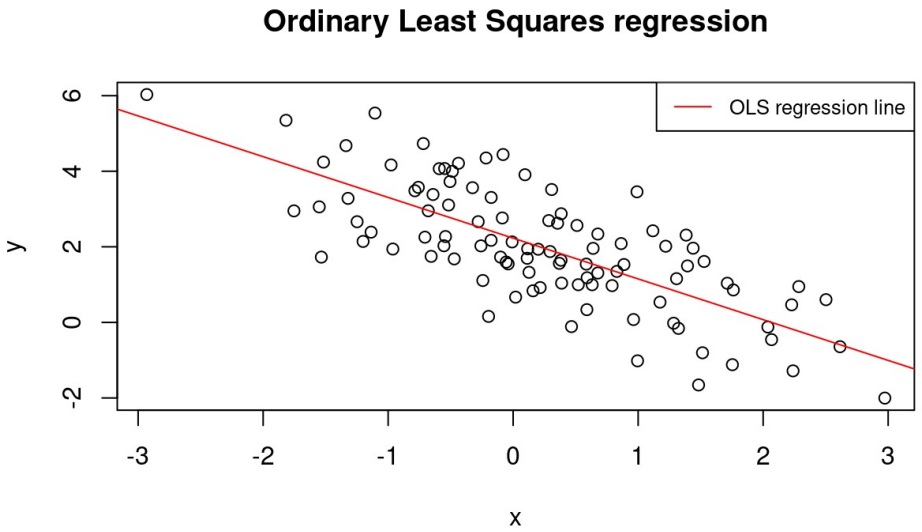
$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_p x_p + \epsilon$$

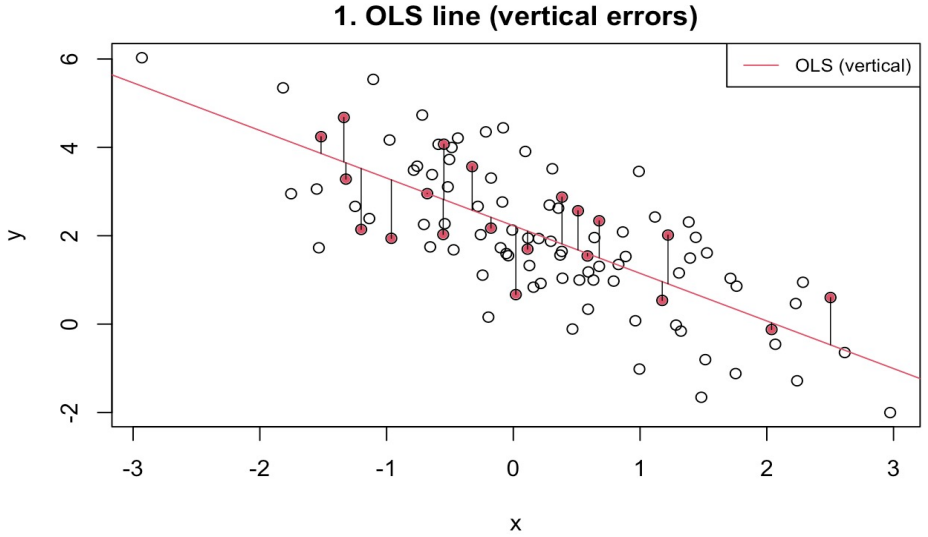
We know: **x1,x2, …. Xp**
y1, y2, … yp

Want to know: **all betas** (unknown)

How?

-> by making sum of squared residuals **minimized!**





$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_p x_p + \epsilon$$

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_p x_p + \epsilon$$

We know: **x1,x2, …. Xp**
y1, y2, … yp

Want to know: **all betas** (unknown)

How?

-> by making sum of squared residuals **minimized!**

$$\widehat{\beta_1}, \widehat{\beta_2}, \ldots\ldots, \widehat{\beta_p} = argmin \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 y_{1i} + \beta_2 y_{2i} + \beta_3 y_{3i} \ \ldots\ldots + \beta_p y_{pi}))$$

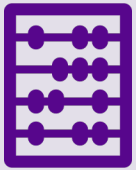
2

$$\widehat{\beta_1}, \widehat{\beta_2}, \ldots\ldots, \widehat{\beta_p} = \underset{}{argmin} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 y_{1i} + \beta_2 y_{2i} + \beta_3 y_{3i} \ \ldots\ldots + \beta_p \ y_{pi}))$$

argmin f(x) = value of x for which minimize f(x) attains its minimum

```
# Example:
f(x) = x + 10
list <- c(10,11, 12)

min( f( c(10, 11, 12) )  )
[1] 20

argmin(f( c(10,11, 12) ) )
[1] 10
```

PART 02



Grid search

visualize

- In this section, we will use grid-search (gridblock) approach to find the optimal point
- **beta.grid** is a matrix of (500*500)×2, representing a range of (β0,β1) pairs.
- The function **calculate.rss** takes as input a vector of betas, along with the data x and y and calculates the corresponding RSS.
- The apply function will call calculate.rss on each pair of betas in beta.grid.
- The x and y arguments to that function are named explicitly and thus supplied by the … arguments to apply.

visualize

```
calculate.rss <- function(betas, x, y) {
  sum((y - betas[1] - betas[2] * x)^2)
}
```

Function to calculate sum of squared residuals

```
beta.from   <- -10
beta.to     <- 10
beta.n.grid <- 500
beta_0 <- seq(beta.from, beta.to, length.out = beta.n.grid)
beta_1 <- seq(beta.from, beta.to, length.out = beta.n.grid)

beta_0.grid <- rep(beta_0, times = beta.n.grid)
beta_1.grid <- rep(beta_1, each  = beta.n.grid)
beta.grid   <- cbind(beta_0.grid, beta_1.grid)

rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)
min.index <- which.min(rss)
beta.grid[min.index,]
```

visualize

```
calculate.rss <- function(betas, x, y) {
  sum((y - betas[1] - betas[2] * x)^2)
}
```

```
beta.from   <- -10
beta.to     <- 10
beta.n.grid <- 500
beta_0 <- seq(beta.from, beta.to, length.out = beta.n.grid)
beta_1 <- seq(beta.from, beta.to, length.out = beta.n.grid)
```

```
beta_0.grid <- rep(beta_0, times = beta.n.grid)
beta_1.grid <- rep(beta_1, each  = beta.n.grid)
beta.grid   <- cbind(beta_0.grid, beta_1.grid)

rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)
min.index <- which.min(rss)
beta.grid[min.index,]
```

Create grid blocks
From -10 to 10 by 500

visualize

```
calculate.rss <- function(betas, x, y) {
  sum((y - betas[1] - betas[2] * x)^2)
}

beta.from   <- -10
beta.to     <- 10
beta.n.grid <- 500
beta_0 <- seq(beta.from, beta.to, length.out = beta.n.grid)
beta_1 <- seq(beta.from, beta.to, length.out = beta.n.grid)

beta_0.grid <- rep(beta_0, times = beta.n.grid)
beta_1.grid <- rep(beta_1, each  = beta.n.grid)
beta.grid   <- cbind(beta_0.grid, beta_1.grid)

rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)
min.index <- which.min(rss)
beta.grid[min.index,]
```

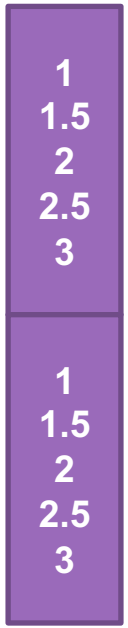
What is this?

visualize

Think of this as **beta_0**

```
rep(beta_0, times = 1)
```

visualize




```
rep(beta_0, times = 2)
```

visualize


```
rep(beta_0, times = 3)
```

visualize

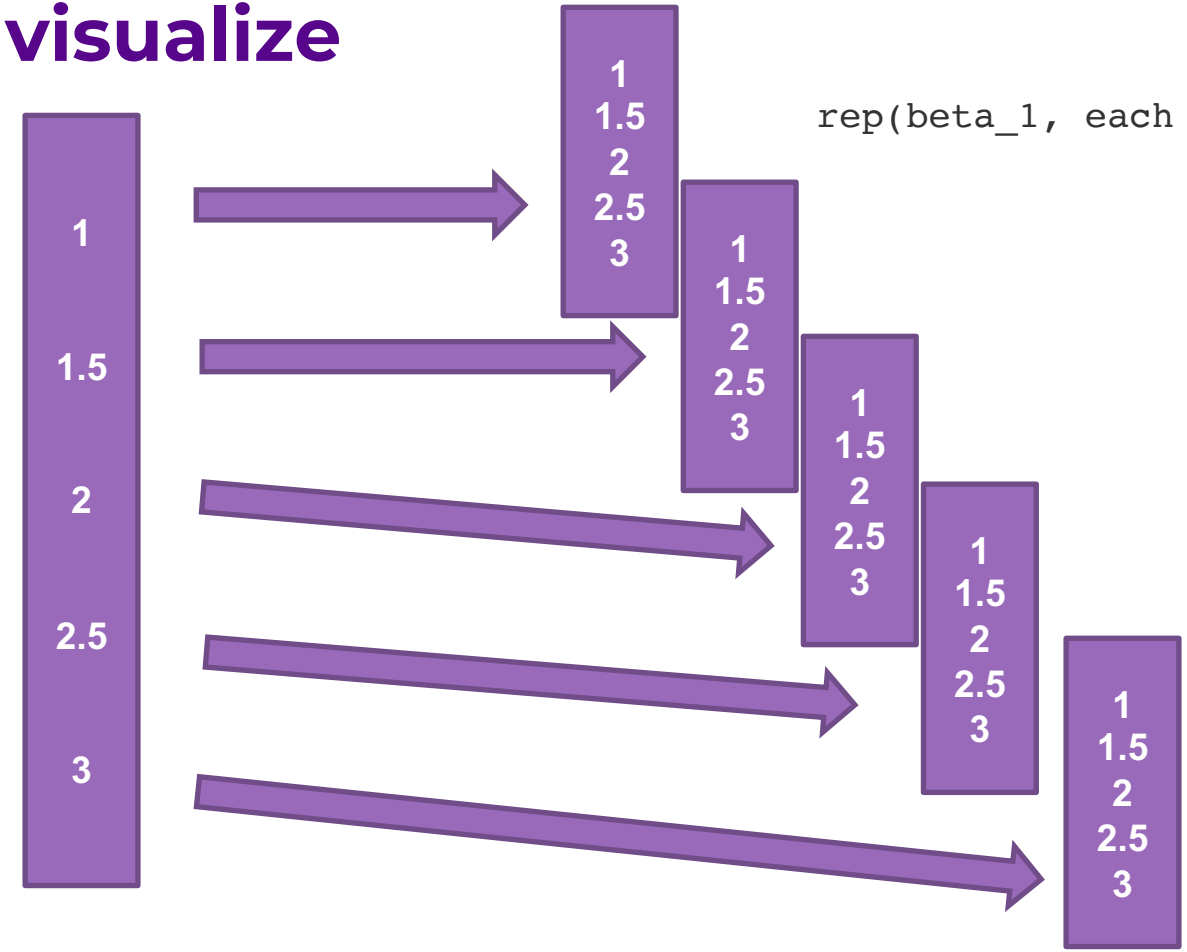
1
1.5
2
2.5
3

Now, what is

```
rep(beta_1, each  = beta.n.grid)
```

visualize

```
rep(beta_1, each  = beta.n.grid)
```



visualize

```
calculate.rss <- function(betas, x, y) {
  sum((y - betas[1] - betas[2] * x)^2)
}

beta.from   <- -10
beta.to     <- 10
beta.n.grid <- 500
beta_0 <- seq(beta.from, beta.to, length.out = beta.n.grid)
beta_1 <- seq(beta.from, beta.to, length.out = beta.n.grid)

beta_0.grid <- rep(beta_0, times = beta.n.grid)
beta_1.grid <- rep(beta_1, each  = beta.n.grid)
beta.grid   <- cbind(beta_0.grid, beta_1.grid)

rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)
min.index <- which.min(rss)
beta.grid[min.index,]
```

Remember apply family?

visualize

```
rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)

min.index <- which.min(rss)

beta.grid[min.index,]
```

visualize

```
rss <- apply(beta.grid, 1, calculate.rss, x = x, y = y)
```

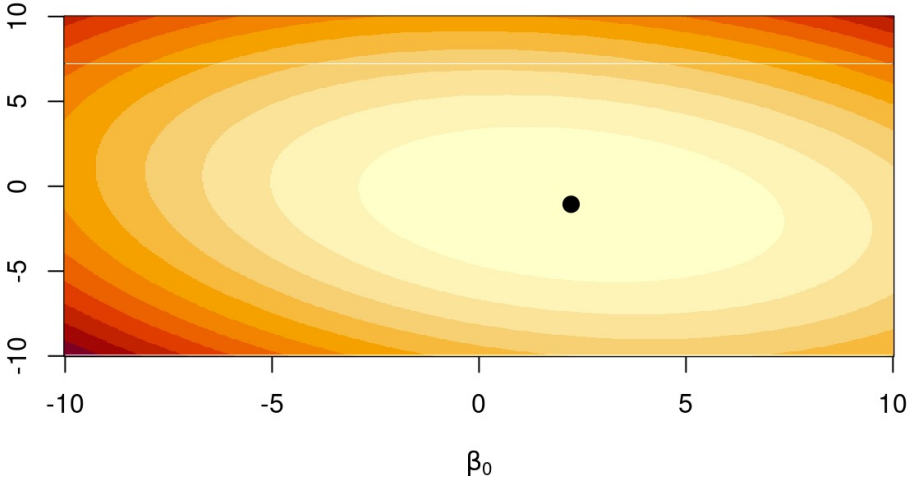
All Beta rowwise function input x / y

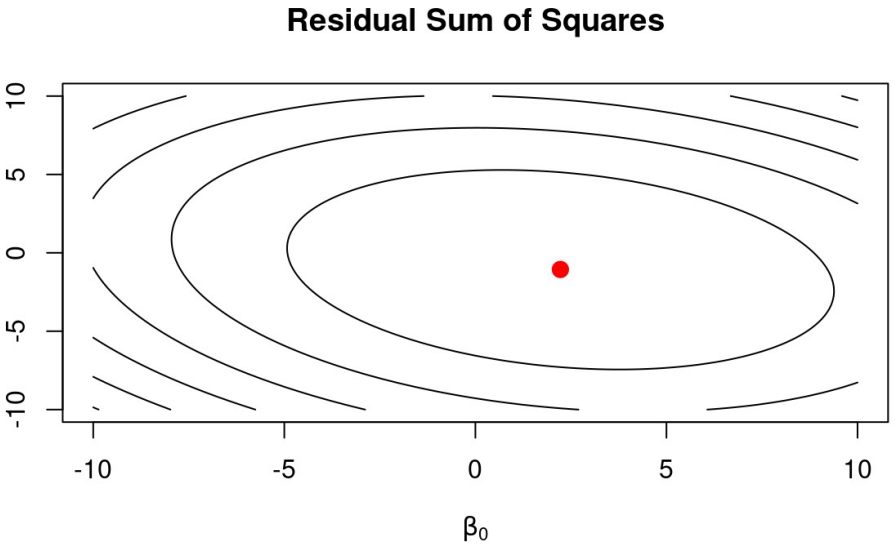
```
min.index <- which.min(rss)
```

Find the minimum beta

```
beta.grid[min.index,]
```

visualize

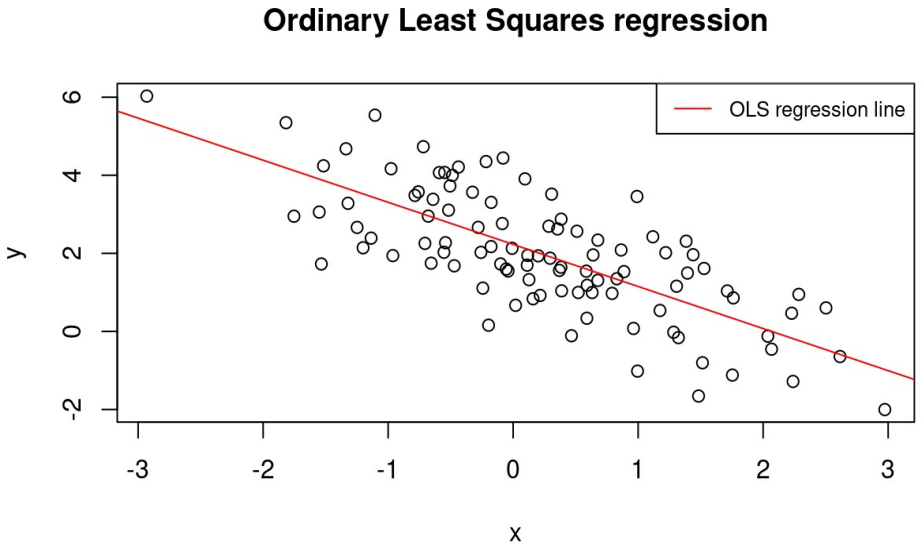


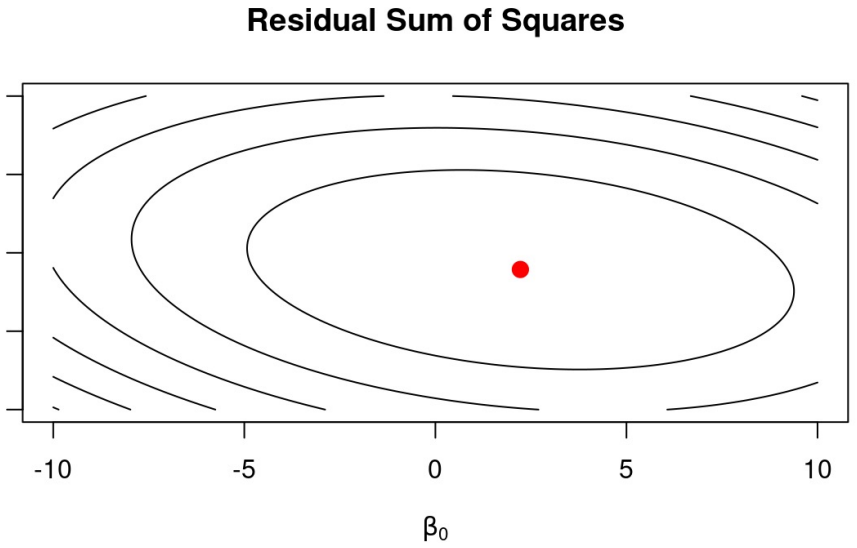


NYU

PART 01

visualize

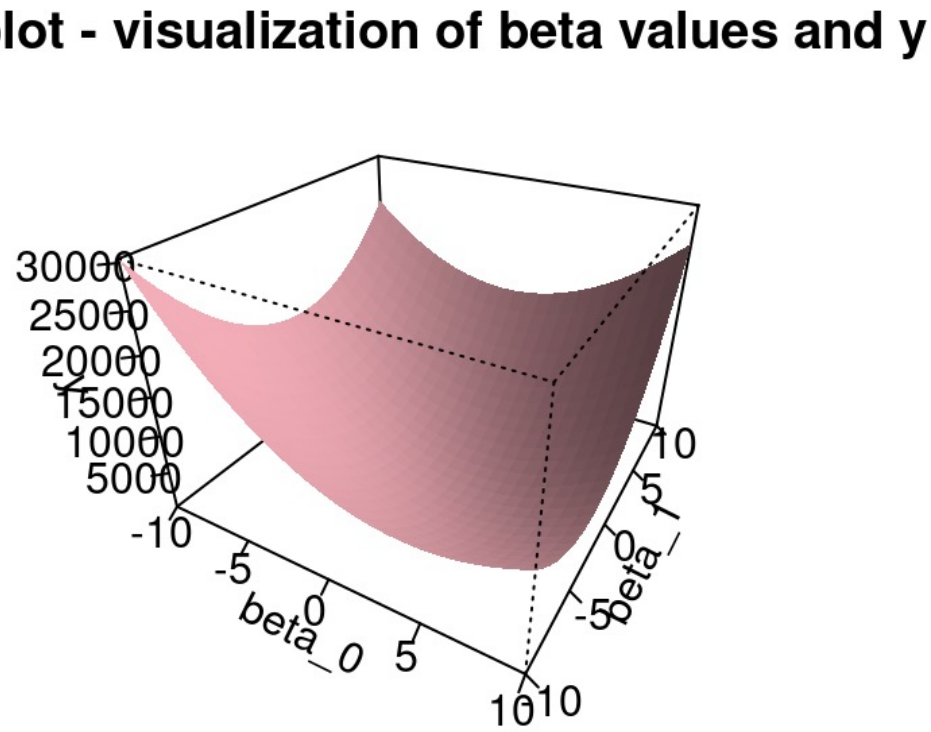




Is it expected?

NYU

visualize


3D plot - visualization of beta values and y
30000
25000
20000
15000
10000
5000
-10
-5
0
5
10
beta_0
-5
0
5
10
beta_1

PART 03

optim function

Optim function

- **optim** function is a general-purpose optimization with six different optimization methods such as Nelder-Mead, quasi-Newton and conjugate-gradient algorithms.

- There are several arguments such as **par=**, which defines the initial values for the parameters and **fn=** is the function that we want to optimize.

- **fn** must be written so that the first argument is the *complete* set of parameters that need to be optimized over.

- You can “parse” (separate) them once inside the function, **fn**, but you must follow this syntax or optim will not run properly.

Optim function

```
# calculating RSS function
calculate.rss <- function(betas, x, y) {
  sum((y - betas[1] - betas[2] * x)^2)
}

# optim function
fit.ols <- optim(par = c(0, 0),
                 fn = calculate.rss,
                 x = x,
                 y = y)

# check the result
fit.ols
```

Initial value of the parameter

Function we want to optimize

Optim function

```
# Results of fit.ols
$par
[1] 2.228626 -1.078387

$value
[1] 115.6954

$counts
function gradient
   89       NA

$convergence
[1] 0

$message
NULL
```

Best sets of parameter

Optim function

```
# compare the output with lm function
fit.lm <- lm(y ~ x)
coef(fit.lm)
```

```
(Intercept)    x
2.228414       -1.078243
```

Results of using linear regression

```
# Results of fit.ols
$par
[1] 2.228626  -1.078387
```

Results of using optim function

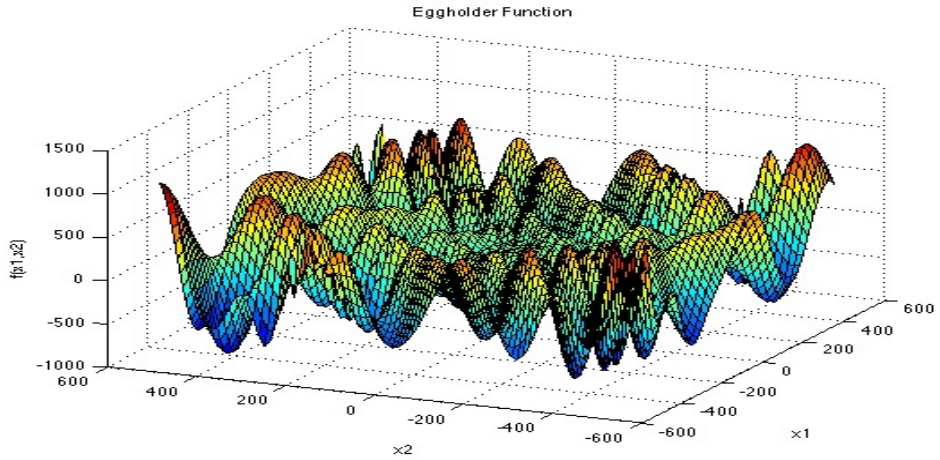
PART 04

challenge

Optimization

- **Eggholder function** is another commonly used function in optimization problem in applied mathematics as it has large number of local minima. The equation of the function can be represented as follows:

$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right)$$



Optimization

- **Eggholder function** is another commonly used function in optimization problem in applied mathematics as it has large number of local minima. The equation of the function can be represented as follows:
- x is in [-512, 512]
- Global minima is **f(x1,x2) = −959.64** at **(x1, x2)=(512, 404.23)**
- Let's test optim function to find out!

Optimization

```
# Results of fit.ols
egg_funct2<- function(x){
  -(x[2]+47)*sin(sqrt(abs(x[2]+(x[1]/2)+47)))-
x[1]*sin(sqrt(abs(x[1]-(x[2]+47))))
}

# define the function
f.egghold <- function(x) { f.x <- egg_funct2(x)
  return(f.x)
}

# optimize (minimize) the function using SANN
method
out.egghold <- optim(c(500,400),
                     f.egghold,
                     method = "SANN")
```

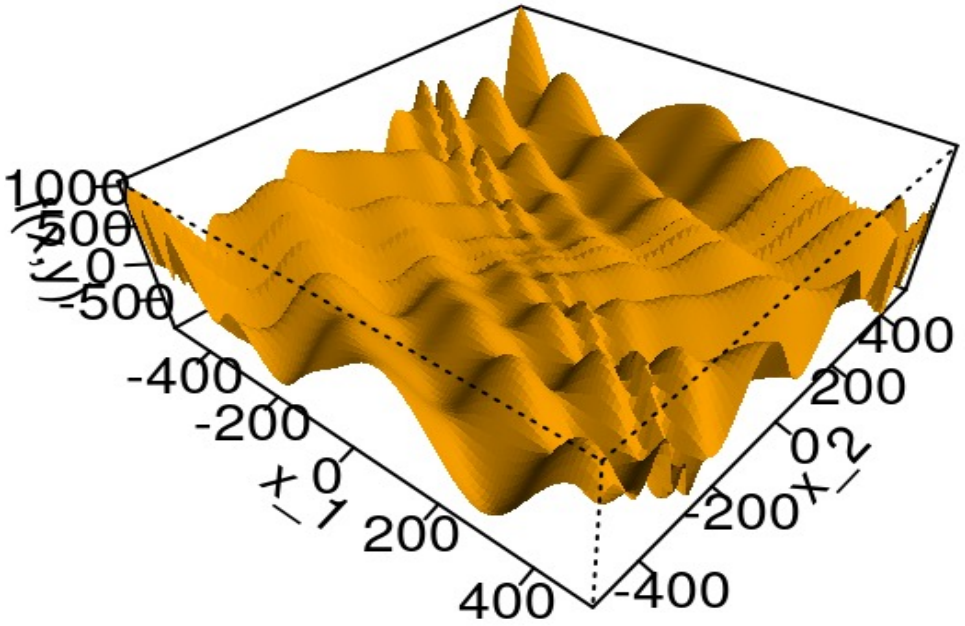
Initial value of the parameter

Function we want to optimize

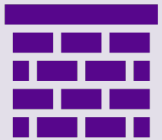
Optimization

```
$par
[1] 522.1463 413.3018

$value
[1] -976.911

$counts
function gradient
10000     NA

$convergence
[1] 0

$message
NULL
```

Not bad at all!



PART 05



Project 1

Project 1

4.1.
Data on U.S. citizens were collected in 2000 by the U.S. Census Bureau. The data set ***LatinoEd.csv-***—a subset of the Census data-—contains data for a sample of Latino immigrants who were naturalized U.S. citizens living in Los Angeles. The variable **Achieve** in this data set provides a measure of educational achievement (see the codebook for more information regarding the interpretation of this variable).

Project 1

a) Construct a plot of the kernel density estimate for the marginal distribution of the educational achievement variable. Discuss all interesting features of this plot.

b) Examine plots of the variable Achieve conditioned on the variable English to compare the educational achievement of Latino immigrants who are fluent in English and those who are not. Create a single publishable display that you believe is the best visual representation of the results of this analysis. In constructing this display, think about the substantive points you want to make and create a graph that best allows you to highlight these conclusions. Write a brief paragraph explaining why you chose to construct your graph the way you did and how it helps answer the research question of interest.

c) Compute appropriate numerical summaries of achievement conditioned on English fluency. Use these summaries (along with evidence culled from your plot) to provide a full comp*ariso*n of the d*istribut*i*ons* of achievement scores for immigrants who are fluent in English and those immigrants who are not fluent in English. Be sure to make comparisons between the measures of center and variation in the distributions. Use the problem context to help you write your answer. Be selective in what you report, remembering that you want to be succinct yet thorough.

End of class Poll

We really appreciate the feedback!