180128022-MAS6024-A3

180128022

PART 1 1.

$$\boldsymbol{y} = X\beta + \boldsymbol{\epsilon}\boldsymbol{\beta} = (\beta_0, \beta_1)^T \epsilon \sim N(0, 3^2 I_n)$$

Now, we can say that the Gaussian Density is given by,

$$p(\boldsymbol{y}|X) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} exp(-\frac{(\boldsymbol{y} - X\boldsymbol{\beta})^2}{2\sigma^2})$$

The log-likelihood is given by,

$$l(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) = -\frac{n}{2} log(2\pi) - \frac{n}{2} log(\sigma^2) - \frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\beta})^2$$

We have n = 50 and $\sigma^2 = 3^2$

$$l(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) = -\frac{50}{2} log(2\pi) - \frac{50}{2} log(\sigma^2) - \frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\beta})^2 l(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) = -\frac{50}{2}[log(2\pi) + log(3^2)] - \frac{1}{(2.3^2)}(\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta}) l(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$$

Given,

$$l(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) = c - \frac{1}{(18)}(\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta})$$

Comparing the two equations, we get,

$$c = -25 log(18\pi)$$

2. A function to calculate the log-likelihood for values of β_0 and β_1

```r
load('Assignment_3_part_1_data.RData')
# function to take user inputs for setting the range of beta-zero
user_input_beta_zero <- function(){
  # here we have used as.numeric to convert the user-inputs into numeric values
  beta_zero_initial_value <- as.numeric(readline("Enter the initial value of beta_zero: "))
  beta_zero_final_value <- as.numeric(readline("Enter the final value of beta_zero: "))
  increment_by <- as.numeric(readline("Enter the step size: "))
  beta_zero_seq<- seq(from = beta_zero_initial_value, to = beta_zero_final_value, by = increment_by)
  return(beta_zero_seq)
}

# function to take user inputs for setting the range of beta-zero
user_input_beta_one <- function(){
  beta_one_initial_value <- as.numeric(readline("Enter the initial value of beta_one: "))
  beta_one_final_value <- as.numeric(readline("Enter the final value of beta_one: "))
  increment_by <- as.numeric(readline("Enter the step size: "))
  beta_one_seq <- seq(from = beta_one_initial_value, to = beta_one_final_value, by = increment_by)
  return(beta_one_seq)
}

# function to combine the above the functions
link_func <- function(){
  beta_zero_seq <- user_input_beta_zero() # assign the value of beta-zero sequence
  beta_one_seq <- user_input_beta_one() # assign the value of beta-one sequence
  x <- assignment3_data[,1] # accessing the x column of the provided dataset
  y <- assignment3_data[,2] # accessing the y column of the provided dataset
  # using sapply instead of for loop
  iter <- sapply(X = beta_zero_seq, beta_one_seq = beta_one_seq, x=x, y=y, FUN = connect)
  return(list(iter, beta_one_seq, beta_zero_seq)) # returning the list
}

# function to calculate the likelihood
calculate_log_likelihood <- function(beta_zero, beta_one, x, y){
  parameter <- (beta_zero + x*beta_one) # summing up the parameters into a variable
  c <- -25*log(18*pi) # taking the value of c as calculated from part 1
  log_likelihood <- c - (1/18)*t(y - parameter)%*%(y - parameter) # applying the formula
  return(log_likelihood)
}

# function to connect the sapply operations
connect <- function(beta_one_seq, beta_zero_seq, x, y){
  sapply(X = beta_one_seq, beta_zero = beta_zero_seq, x=x, y=y, FUN = calculate_log_likelihood)
}
```

Now, calling the function:

```r
link_func()
```

```
## [[1]]
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
##  [1,] -131.5521 -131.5460 -131.5404 -131.5354 -131.5309 -131.5270
##  [2,] -131.2770 -131.2704 -131.2643 -131.2588 -131.2539 -131.2495
##  [3,] -131.0214 -131.0143 -131.0077 -131.0018 -130.9964 -130.9915
##  [4,] -130.7852 -130.7776 -130.7706 -130.7642 -130.7583 -130.7530
##  [5,] -130.5685 -130.5604 -130.5530 -130.5461 -130.5397 -130.5339
##  [6,] -130.3712 -130.3627 -130.3548 -130.3474 -130.3406 -130.3343
##  [7,] -130.1934 -130.1844 -130.1760 -130.1682 -130.1609 -130.1541
##  [8,] -130.0351 -130.0256 -130.0168 -130.0084 -130.0007 -129.9935
##  [9,] -129.8962 -129.8863 -129.8769 -129.8681 -129.8599 -129.8522
## [10,] -129.7768 -129.7664 -129.7566 -129.7473 -129.7386 -129.7305
## [11,] -129.6768 -129.6660 -129.6557 -129.6460 -129.6368 -129.6282
## [12,] -129.5963 -129.5850 -129.5743 -129.5641 -129.5544 -129.5453
## [13,] -129.5353 -129.5235 -129.5123 -129.5016 -129.4915 -129.4819
## [14,] -129.4938 -129.4815 -129.4698 -129.4586 -129.4481 -129.4380
## [15,] -129.4716 -129.4589 -129.4468 -129.4351 -129.4241 -129.4136
## [16,] -129.4690 -129.4558 -129.4432 -129.4311 -129.4196 -129.4086
## [17,] -129.4858 -129.4722 -129.4590 -129.4465 -129.4345 -129.4230
## [18,] -129.5221 -129.5080 -129.4944 -129.4814 -129.4689 -129.4570
## [19,] -129.5779 -129.5632 -129.5492 -129.5357 -129.5228 -129.5104
## [20,] -129.6531 -129.6380 -129.6235 -129.6095 -129.5961 -129.5832
## [21,] -129.7477 -129.7322 -129.7172 -129.7027 -129.6889 -129.6755
##              [,7]      [,8]      [,9]     [,10]     [,11]
##  [1,] -131.5236 -131.5208 -131.5185 -131.5169 -131.5157
##  [2,] -131.2457 -131.2424 -131.2397 -131.2375 -131.2359
##  [3,] -130.9872 -130.9835 -130.9803 -130.9776 -130.9756
##  [4,] -130.7482 -130.7440 -130.7403 -130.7372 -130.7347
##  [5,] -130.5287 -130.5240 -130.5198 -130.5163 -130.5132
##  [6,] -130.3286 -130.3234 -130.3188 -130.3148 -130.3113
##  [7,] -130.1480 -130.1423 -130.1373 -130.1327 -130.1288
##  [8,] -129.9868 -129.9807 -129.9752 -129.9702 -129.9657
##  [9,] -129.8451 -129.8385 -129.8325 -129.8271 -129.8222
## [10,] -129.7229 -129.7158 -129.7093 -129.7034 -129.6980
## [11,] -129.6201 -129.6126 -129.6056 -129.5992 -129.5934
## [12,] -129.5368 -129.5288 -129.5214 -129.5145 -129.5082
## [13,] -129.4729 -129.4645 -129.4566 -129.4492 -129.4425
## [14,] -129.4285 -129.4196 -129.4113 -129.4034 -129.3962
## [15,] -129.4036 -129.3942 -129.3854 -129.3771 -129.3694
## [16,] -129.3982 -129.3883 -129.3790 -129.3702 -129.3620
## [17,] -129.4122 -129.4018 -129.3920 -129.3828 -129.3742
## [18,] -129.4456 -129.4348 -129.4246 -129.4149 -129.4057
## [19,] -129.4985 -129.4873 -129.4765 -129.4664 -129.4568
## [20,] -129.5709 -129.5592 -129.5480 -129.5373 -129.5273
## [21,] -129.6628 -129.6505 -129.6389 -129.6278 -129.6172
##
## [[2]]
##  [1] 3.50 3.51 3.52 3.53 3.54 3.55 3.56 3.57 3.58 3.59 3.60 3.61 3.62 3.63
## [15] 3.64 3.65 3.66 3.67 3.68 3.69 3.70
##
## [[3]]
##  [1] 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70
```

3. Now, we have got the log-likelihood values of β_0 and β_1. Using these, we are going to write a function 'calculate_mle' calculate The Maximum Likelihood Estimates (MLE):

```r
calculate_mle <- function(){
  headings <- link_func() # assigning the values of our previous function
  create_grid <- headings[1] # creating a grid
  beta_zero <- headings[2] # beta-zero values
  beta_one <- headings[3] # beta-one values
  # row-column identifier
  identifier <- which(x = create_grid[[1]] == max(create_grid[[1]]), arr.ind = TRUE)
  rows <- identifier[1]
  columns <- identifier[2]
  beta_0 <- beta_zero[[1]][rows] # MLE for beta-zero
  beta_1 <- beta_one[[1]][columns] # MLE for beta-one
  print(beta_0)
  print(beta_1)
}
```

Now, calling the function:

```r
calculate_mle()
```

```
## [1] 3.65
## [1] 0.7
```

PART 2 1. We are going to write a function that simulates the movement of an object around a square based on user-inputs.

```r
# Writing a function to calculate the moves
predict_move <- function(){
  origination <- readline("Enter the origin vertex: ") # User-input for the starting vertex
  destination <- readline("Enter the ending vertex: ") # User-input for H or target
  destination <- unlist(strsplit(destination,",")) # unlisting the vectors
  n <- readline("Enter the number of moves: ") # user-input for number of moves
  p <- readline("Enter the probability: ") # user-input for the value of probability
  # taking a random binomial distribution for the moves
  randomness <- rbinom(n = as.integer(n), size = 1, prob = as.numeric(p))
```

```r
object_point <- c(1, 2)

# setting the values of x and y for each point of origin
if(origination == "V1"){
  x <- 1
  y <- 0
}
else if(origination == "V2"){
  x <- 1
  y <- 1
}
else if(origination == "V3"){
  x <- 0
  y <- 1
}
else{
  x <- 0
  y <- 0
}
# finding the direction of the point as per a random binomial distribution
for(l in 1:length(randomness)){
  if(randomness[l] == 1){
    randomness[l] = rbinom(n = 1, size = 1, prob = 0.5) + 1
  }
}
print(randomness)
object_point_list <- c()

# appending the values as per the fulfilled conditions
if(x == 1 & y == 0){
  object_point_list <- append(object_point_list, "V1")
}
else if(x == 1 & y == 1){
  object_point_list <- append(object_point_list, "V2")
}
else if(x == 0 & y == 1){
  object_point_list <- append(object_point_list, "V3")
}
else{
  object_point_list <- append(object_point_list, "V4")
}
# taking a for loop to ascertain the values for l in 0 to 2
for(l in randomness){
  if(l == 2){
    if(x == 1){
      x <- 0
    }
    else{
      x <- 1
    }
  }
  else if(l == 1){
    if(y == 1){
        y <- 0
      }
      else{
        y <- 1
      }
    }
    else{
      if(y == 0){
        y <- 1
      }
      else{
        y <- 0
      }
      if(x == 1){
        x <- 0
      }
      else{
        x <- 1
      }
    }
    # condition checking
    if(x == 1 & y == 0){
      object_point_list <- append(object_point_list, "V1")
    }
    else if(x == 1 & y == 1){
      object_point_list <- append(object_point_list, "V2")
    }
    else if(x == 0 & y == 1){
      object_point_list <- append(object_point_list, "V3")
    }
    else{
      object_point_list <- append(object_point_list, "V4")
    }

  }
  print(object_point_list) # list of vertices traversed
  print(intersect(destination, object_point_list)) #
  print(length(intersect(destination, object_point_list))/as.integer(n)) # probability

}
predict_move()
```

```
##  [1] 1 0 2 2 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 2 0 0 1 0 0 2 0 1 1 2 2 0 0 0
## [36] 0 0 2 1 0 0 1 0 0 2 0 0 1 0 1 0 2 0 1 0 0 0 0 0 0
##  [1] "V1" "V2" "V4" "V1" "V4" "V2" "V4" "V2" "V4" "V2" "V4" "V2" "V1" "V2"
## [15] "V4" "V2" "V1" "V3" "V1" "V3" "V4" "V1" "V3" "V1" "V2" "V4" "V2" "V3"
## [29] "V1" "V2" "V1" "V4" "V1" "V3" "V1" "V3" "V1" "V3" "V2" "V1" "V3" "V1"
## [43] "V2" "V4" "V2" "V3" "V1" "V3" "V4" "V2" "V1" "V3" "V2" "V4" "V3" "V1"
## [57] "V3" "V1" "V3" "V1" "V3"
## [1] "V2" "V3"
## [1] 0.03333333
```

2. When H and starting vertices are the same, the values converge to the average of the four sides.