Test-Implementation-05-31-2021

Sixtus Dakurah

5/31/2021

Define the variables:

B : Total amount available for disbursement.

$S_i, \quad i = 1, 2, ..., s$: Denote segment i with a total of s segments.

L : Loan amount.

P : The price variable.

Step 1: Fit the logistic model for a given segment. Here the segment indexes will be omitted and all variables will be assumed to pertain to a single segment.

The design matrix is of the form:

$X = [j, x_1, x_2, x_3]$ where $x_l = [x_{1,l}, ..., x_{m,l}]'$; $l = 1$: Segment, $l = 2$: Price, $l = 3$: Loan Amount.

The model for predicting the probability of booking is of the form:

$$\pi(X, \beta) = [\sigma(z_i), ..., \sigma(z_m)]' \tag{1}$$

Where $\sigma(z_k) = [1 + exp\{-z_k\}]^{-1}$ and $z_k = x_k\beta$, $k = 1, ..., m$ the number of examples.

An expressive form for $x_k\beta = \beta_0 + \beta_1 x_{k,1} + \beta_2 x_{k,2} + \beta_3 x_{k,3} + \beta_{1,2} x_{k,1} * x_{k,2} + \beta_{1,3} x_{k,1} * x_{k,3}$

We now simulate some data:

```
set.seed(111)
# set the number of examples and segments
m <- 300; m1 <- 100; m2<-100; m3 <- 100; s <- 3
# create the segment groups
x1 <- c(rep('s1', m1), rep('s2', m2), rep('s3', m3))
# generate the prices
x2 <- c(
  rlnorm(m1, meanlog = 1, sdlog = 1),
  rlnorm(m2, meanlog = 3, sdlog = 1),
  rlnorm(m3, meanlog = 5, sdlog = 1)
  )
# generate the loan amount
x3 <- c(
  rep(rlnorm(1, meanlog = 11, sdlog = 5), m1),
  rep(rlnorm(1, meanlog = 3, sdlog = 3), m2),
  rep(rlnorm(1, meanlog = 15, sdlog = 7), m3)
  )
# generate the prob
prob.level1 <- function(val){
```

```r
  ifelse(val < quantile(x2[1:m1], 0.25), runif(1, 0.4, 1), runif(1, 0, 1)) }
prob.level2 <- function(val){
  ifelse(val < quantile(x2[(m1+1):(m1+m2)], 0.25), runif(1, 0.4, 01), runif(1, 0, 1)) }
prob.level3 <- function(val){
  ifelse(val < quantile(x2[(m1+m2+1):m], 0.25), runif(1, 0.4, 01), runif(1, 0, 1)) }
probs <- c(

  do.call(rbind, lapply(x2[1:m1], prob.level1)),
  do.call(rbind, lapply(x2[(m1+1):(m1+m2)], prob.level2)),
  do.call(rbind, lapply(x2[(m1+m2+1):m], prob.level3))

)
data.sim <- data.frame(x1 = as.factor(x1), x2, x3, probs,
                       y = as.factor(ifelse(probs > 0.5, 's', 'f')))
# the price can not be more than the loan amount
data.sim <- data.sim %>% mutate(x3 = ifelse(x2 > x3, 10*x3, x3))

head(data.sim)
```

```
##   x1        x2     x3      probs y
## 1 s1 3.4391375 122474 0.03215337 f
## 2 s1 1.9527998 122474 0.83040273 s
## 3 s1 1.9904807 122474 0.43669723 f
## 4 s1 0.2718933 122474 0.55923602 s
## 5 s1 2.2913106 122474 0.71349150 s
## 6 s1 3.1276384 122474 0.91685147 s
```

```r
summary(data.sim)
```

```
##    x1            x2                   x3                 probs           y
##  s1:100   Min.   :   0.1208   Min.   :    59.8   Min.   :0.000673   f:131
##  s2:100   1st Qu.:   4.1426   1st Qu.:    59.8   1st Qu.:0.346875   s:169
##  s3:100   Median :  21.6488   Median :122474.0   Median :0.555124
##           Mean   : 102.3133   Mean   :311209.4   Mean   :0.538676
##           3rd Qu.:  96.6912   3rd Qu.:811045.9   3rd Qu.:0.775436
##           Max.   :2040.9550   Max.   :811045.9   Max.   :0.998970
```

We now build a logistic model using the data from segment 1

```r
data.segment1 <- data.sim %>% filter(x1 == 's1')
#glm.segment1.fit <- glm(y~x1 + x2 + x3 + I(x1*x2) +
#I(x1*x3), data = data.segment1, family = binomial)
glm.segment1.fit <- glm(y~x1 + x2, data = data.sim, family = binomial)
summary(glm.segment1.fit)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = binomial, data = data.sim)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4345  -1.2805   0.9415   1.0765   1.4925
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.2429467  0.2014747   1.206    0.228
## x1s2         0.3446049  0.2902566   1.187    0.235
## x1s3        -0.1807773  0.3226387  -0.560    0.575
## x2          -0.0003811  0.0005886  -0.648    0.517
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 411.06  on 299  degrees of freedom
## Residual deviance: 406.02  on 296  degrees of freedom
## AIC: 414.02
##
## Number of Fisher Scoring iterations: 4
```

Extract the coefficients:

```
seg1.intercept <- ((glm.segment1.fit)$coefficients)[[1]]
coef2 <- ((glm.segment1.fit)$coefficients)[[2]]
coef3 <- ((glm.segment1.fit)$coefficients)[[3]]
seg2.intercept <- seg1.intercept +  ((glm.segment1.fit)$coefficients)[[2]]
seg3.intercept <- seg1.intercept +  ((glm.segment1.fit)$coefficients)[[3]]
price.coef <- ((glm.segment1.fit)$coefficients)[[4]]
```

The fitted model has the form:

$\hat{\sigma}(z_k) = [1 + exp\{-z_k\}]^{-1}$ This will now be different

$\pi(x_2, \hat{\beta}) = [\hat{\sigma}(z_1), ..., \hat{\sigma}(z_k)]'$

```
# get the predictions
segment1.pred.probs <- predict(glm.segment1.fit, type = "response")
data.sim <- data.sim %>%
  mutate(pred.probs = segment1.pred.probs,
         pred.y = as.factor(ifelse(pred.probs > 0.5, 's', 'f')))
head(data.sim)
```

```
##   x1        x2     x3      probs y pred.probs pred.y
## 1 s1 3.4391375 122474 0.03215337 f  0.5601168      s
## 2 s1 1.9527998 122474 0.83040273 s  0.5602563      s
## 3 s1 1.9904807 122474 0.43669723 f  0.5602528      s
## 4 s1 0.2718933 122474 0.55923602 s  0.5604142      s
## 5 s1 2.2913106 122474 0.71349150 s  0.5602246      s
## 6 s1 3.1276384 122474 0.91685147 s  0.5601460      s
```

```
table(data.sim$pred.y, data.sim$y)
```

```
##
##       f   s
##   f  30  19
##   s 101 150
```

Step 2: We now optimize over the price variable.

```
# Compute the total amount available for disbursement
(B <- sum(data.sim$x3))
```

```
## [1] 93362811
```

```
B <- 00
```

Denote the profit as:

$\rho(x_2, x_4) = x_2 - x_4$ where the new variable x_4 is the cost of booking.

The expected profit is then given as:

$$E\left[\rho(x_2, x_4)|x_{\{1,i\}}, x_2\right] = \sum_{k=1}^{m_i} \pi_i(x_{\{k,2\}}, \hat{\beta})\rho(x_{\{k,2\}}, x_{\{k,4\}}) \tag{2}$$

We also have the expected loan amount of the form:

$$E\left[x_3|x_{\{1,i\}}, x_2\right] = \sum_{k=1}^{m_i} \pi_i(x_{\{k,2\}}, \hat{\beta}) * x_{k,3} \tag{3}$$

Where $\pi_i(x_{\{k,2\}}, \hat{\beta})$ is assumed to be constructed for segment i.

The optimization problem is now of the form:

$$argmin_{x_2} E\left[\rho(x_2, x_4)|x_{\{1,i\}}, x_2\right] \qquad s.t. \quad \sum_{i=1}^{s} E\left[x_3|x_{\{1,i\}}, x_2\right] < B \tag{4}$$

This optimization problem is not well-defined and a solution can not be obtained if we employ single-variable optimization.

This will be feasible if we resort to vector-valued optimization in which case the $\{k\}$ in (2) and (3) will be mute. But that too will not be useful in this context as we're not looking to optimize for all known price points.

A modification that will make optimization feasible

Let $\rho(x_{\{k,2\}}, x_{\{k,4\}}) = x_{\{k,2\}} - x_{\{k,4\}}$ be the profit for a randomly chosen price point in a given segment i.

The expected profit is now given as:

$$E\left[\rho(x_{\{k,2\}}, x_{\{k,4\}})|x_{\{k_i,1\}}, x_{\{k,2\}}\right] = \pi_i(x_{\{k,2\}}, \hat{\beta})\rho(x_{\{k,2\}}, x_{\{k,4\}}) \tag{5}$$

We also have the expected loan amount of the form:

$$E\left[x_{\{k,3\}}|x_{\{k_i,1\}}, x_{\{k,2\}}\right] = \pi_i(x_{\{k,2\}}, \hat{\beta}) * x_{k,3} \tag{6}$$

We now have the optimization problem:

$$argmin_{x_{\{k,2\}}} E\left[\rho(x_{\{k,2\}}, x_{\{k,4\}})|x_{\{k_i,1\}}, x_{\{k,2\}}\right] \qquad s.t. \quad \sum_{i=1}^{s} E\left[x_{\{k,3\}}|x_{\{k_i,1\}}, x_{\{k,2\}}\right] < B \tag{7}$$

We note that **B** used here does not make the constraint global. Maybe it's possible to workout a constraint that makes the B have global properties?

The Lagrangian is of the form:

$$F(x_{k,2}, \lambda) = E\left[\rho(x_{\{k,2\}}, x_{\{k,4\}})|x_{\{k_i,1\}}, x_{\{k,2\}}\right] - \lambda \left[\sum_{i=1}^{s} E\left[x_{\{k,3\}}|x_{\{k_i,1\}}, x_{\{k,2\}}\right] - B\right] \tag{8}$$

For ease of manipulation let:

$f(x_{k,2}) = E\left[\rho(x_{\{k,2\}}, x_{\{k,4\}})|x_{\{k_i,1\}}, x_{\{k,2\}}\right]$ and $c(x_{k,2}) = \sum_{i=1}^{s} E\left[x_{\{k,3\}}|x_{\{k_i,1\}}, x_{\{k,2\}}\right] - B$

Using Newton's method, we have the following derivations:

Opt 1: Quadratic Taylor series expansion for some chosen starting values $x^o_{k,2}, \lambda^o$:

$$F(x_{k,2}, \lambda) \approx F(x^o_{k,2}, \lambda^o) + (x_{k,2} - x^o_{k,2}) \left.\frac{\partial F}{\partial x_{k,2}}\right|_0 + (\lambda - \lambda^o) \left.\frac{\partial F}{\partial \lambda}\right|_0 + \frac{1}{2}(x_{k,2} - x^o_{k,2})^2 \left.\frac{\partial^2 F}{\partial x^2_{k,2}}\right|_0 + (x_{k,2} - x^o_{k,2}) * (\lambda - \lambda^o) \left.\frac{\partial^2 F}{\partial x_{k,2} \partial \lambda}\right|_0 \tag{9}$$

Opt 2: Inserting (8) into (9), we have:

$$\begin{aligned} F(x_{k,2}, \lambda) \approx F(x^o_{k,2}, \lambda^o) + (x_{k,2} - x^o_{k,2}) \left\{ \left.\frac{\partial f}{\partial x_{k,2}}\right|_0 - \lambda^o \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 \right\} - (\lambda - \lambda^o) c(x^o_{k,2}) + \\ \frac{1}{2}(x_{k,2} - x^o_{k,2})^2 \left\{ \left.\frac{\partial^2 f}{\partial x^2_{k,2}}\right|_0 - \lambda \left.\frac{\partial^2 c}{\partial x^2_{2,k}}\right|_0 \right\} - (x_{k,2} - x^o_{k,2}) * (\lambda - \lambda^o) \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 \end{aligned} \tag{10}$$

Note: the last part, the derivative is w.r.t. only the $x_{k,2}$, as that of λ goes to 1.

Opt 3: That the maximum is achieved at $x_{k,2}$ requires the necessary condition:

$$\frac{\partial F}{\partial x_{k,2}} = \left.\frac{\partial f}{\partial x_{k,2}}\right|_0 + (x_{k,2} - x^o_{k,2}) \left\{ \left.\frac{\partial^2 f}{\partial x^2_{k,2}}\right|_0 - \lambda \left.\frac{\partial^2 c}{\partial x^2_{2,k}}\right|_0 \right\} - \lambda \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 = 0 \tag{11}$$

Opt 4: We now derive the gradient update rules

We can easily see that:

$$\Delta x_{k,2} = \frac{\left\{ \lambda \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 - \left.\frac{\partial f}{\partial x_{k,2}}\right|_0 \right\}}{\left\{ \left.\frac{\partial^2 f}{\partial x^2_{k,2}}\right|_0 - \lambda \left.\frac{\partial^2 c}{\partial x^2_{2,k}}\right|_0 \right\}} \tag{12}$$

Similarly for a first order expansion of the constraints, we can solve a new value of λ as follows:

$$c(x_{k,2}) = c(x^o_{k,2}) + \Delta x_{k,2} \left.\frac{\partial c}{\partial x_{k,2}}\right|_{x^o_{k,2}} = 0 \tag{13}$$

If we put (12) into (13), we have the following value of λ:

$$\lambda = \frac{\left\{ \left.\frac{\partial f}{\partial x_{k,2}}\right|_0 \right\}}{\left\{ \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 \right\}} - \frac{\left\{ \left.\frac{\partial^2 f}{\partial x^2_{k,2}}\right|_0 - \lambda \left.\frac{\partial^2 c}{\partial x^2_{2,k}}\right|_0 \right\} * c(x^o_{k,2})}{\left\{ \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 \right\}^2} \tag{14}$$

Putting this into (12) will give final update rule.

Step 3: Algorithm

The individual components for (12) are derived as follows:

$$\begin{aligned}\left\{ \left.\frac{\partial f}{\partial x_{k,2}}\right|_0 \right\} &= \left\{ \left.\frac{\partial \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\rho(x_{\{k,2\}}, x_{\{k,4\}})\right]}{\partial x_{k,2}}\right|_0 \right\} \\ &= \left\{ \left.\frac{\partial \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]}{\partial x_{k,2}}\rho(x_{\{k,2\}}, x_{\{k,4\}}) + \pi_i(x_{\{k,2\}}, \hat{\beta})\frac{\partial \left[\rho(x_{\{k,2\}}, x_{\{k,4\}})\right]}{\partial x_{k,2}}\right|_0 \right\} \\ &= \left\{ \hat{\beta}_2 exp\{-z_k\} \left[\hat{\sigma}(z_k)\right]^2 * \rho(x_{\{k,2\}}, x_{\{k,4\}}) + \pi_i(x_{\{k,2\}}, \hat{\beta})\Big|_0 \right\}\end{aligned} \tag{15}$$

$$\begin{aligned}\left\{ \left.\frac{\partial^2 f}{\partial x^2_{k,2}}\right|_0 \right\} &= \left\{ \frac{\partial}{\partial x_{k,2}} \left.\left[\hat{\beta}_2 exp\{-z_k\}\hat{\sigma}(z_k) * \rho(x_{\{k,2\}}, x_{\{k,4\}}) + \pi_i(x_{\{k,2\}}, \hat{\beta})\right]\right|_0 \right\} \\ &= \left(-\hat{\beta}^2_2 exp\{-z_k\} * \left[\hat{\sigma}(z_k)\right]^2 + 2\hat{\beta}^2_2 \left[exp\{-z_k\}\right]^2 * \left[\hat{\sigma}(z_k)\right]^3\right) * \rho(x_{\{k,2\}}, x_{\{k,4\}}) + \\ &+ \hat{\beta}_2 exp\{-z_k\} \left[\hat{\sigma}(z_k)\right]^2 + \hat{\beta}_2 exp\{-z_k\} \left[\hat{\sigma}(z_k)\right]^2 |_0\end{aligned} \tag{16}$$

$$\begin{aligned}\left\{ \left.\frac{\partial c}{\partial x_{k,2}}\right|_0 \right\} &= \left\{ \left.\frac{\partial \left[\sum_{i=1}^{s} \left[\pi_i(x_{\{k,2\}}, \hat{\beta}) * x_{k,3}\right] - B\right]}{\partial x_{k,2}}\right|_0 \right\} \\ &= \left\{ \sum_{i=1}^{s} \hat{\beta}_2 exp\{-z_k\} \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]^2 * x_{k,3}\Big|_0 \right\}\end{aligned} \tag{17}$$

$$\begin{aligned}\left\{ \left.\frac{\partial^2 c}{\partial x^2_{k,2}}\right|_0 \right\} &= \left\{ \sum_{i=1}^{s} \hat{\beta}_2 \frac{\partial exp\{-z_k\}}{\partial x_{k,2}} * \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]^2 * x_{k,3} + \hat{\beta}_2 exp\{-z_k\} * \frac{\partial}{\partial x_{k,2}} \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]^2 * x_{k,3}\Big|_0 \right\} \\ &= \left\{ \sum_{i=1}^{s} -\hat{\beta}^2_2 exp\{-z_k\} * \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]^2 * x_{k,3} + 2\hat{\beta}^2_2 exp\{-2z_k\} * \left[\pi_i(x_{\{k,2\}}, \hat{\beta})\right]^3 * x_{k,3}\Big|_0 \right\}\end{aligned} \tag{18}$$

We can now optimize as follows:

Input: $\hat{\beta}_0^*, \hat{\beta}_2$ the estimated coefficients from the logistic model.

Output: $x_{\{k,2\}}$ the optimized price variable.

-1 Initialize points $x^o_{\{k,2\}}$ and λ^o.

-2 Whiles not (11) do:

- Compute (15), (16), (17) and (18).

- Obtain a new $x_{\{k,2\}}$ using (12) and (14).

Step 4: Implementation

```
# Initialize the price variables for the three segments:
x21 <- mean((data.sim %>% filter(x1=="s1"))$x2)
x22 <- mean((data.sim %>% filter(x1=="s2"))$x2)
x23 <- mean((data.sim %>% filter(x1=="s3"))$x2)
x31 <- mean((data.sim %>% filter(x1=="s1"))$x3)
x32 <- mean((data.sim %>% filter(x1=="s2"))$x3)
x33 <- mean((data.sim %>% filter(x1=="s3"))$x3)
# Initialize the lambda
lambda <- 0.2
```

```
K <- 100
loan_amount_vector <- c(B, B, B)
booking_cost <- 10
variables_vector <- c(1, 0, 0, x21)
coefficients_vector <- c(seg1.intercept, coef2, coef3, price.coef)
```

Build the utility functions here

```
# Function to compute the exponent of the linear form:
compute_exp_lin_form <- function(variables_vector, coefficients_vector){
  # compute the linear form
  ln.form <- variables_vector*coefficients_vector
  return(exp(-sum(ln.form)))
}

# Unit test
print(compute_exp_lin_form(variables_vector, coefficients_vector))
```

```
## [1] 0.785714
```

```
# Function to compute the estimated predicted probabilities:
compute_sigma_hat <- function(variables_vector, coefficients_vector){
  # compute the linear form
  ln.form <- variables_vector*coefficients_vector
  return <- (1 + compute_exp_lin_form(variables_vector, coefficients_vector) )^(-1)
}
# Unit test
print(compute_sigma_hat(variables_vector, coefficients_vector))
```

```
## [1] 0.5600001
```

```r
# Function to compute the 1st partial derivative of f w.r.t. the price: (15)
compute_first_pd_of_f <- function(booking_cost, variables_vector, coefficients_vector){
  # first compute sigma of z-k using (1)
  price_coefficient  = coefficients_vector[4] # it's assumed this is in the third place
  lin_form <- compute_exp_lin_form(variables_vector, coefficients_vector)
  sigma_value <- compute_sigma_hat(variables_vector, coefficients_vector)
  # Implement the equation corresponding to (15)
  return_value <- (price_coefficient*lin_form*(sigma_value^2)*
                     (variables_vector[4] - booking_cost)) + sigma_value
  #print(variables_vector)
  #print(paste("New First PD of f in CFPDF: ", return_value))
  return(return_value)
}
compute_first_pd_of_f(booking_cost, variables_vector, coefficients_vector)
```

```
## [1] 0.5604995
```

```r
# Function to compute the 2nd partial derivative of f w.r.t. the price: (16)
compute_second_pd_of_f <- function(booking_cost, variables_vector, coefficients_vector){
  # first compute sigma of z-k using (1)
  price_coefficient  = coefficients_vector[4] # it's assumed this is in the third place
  lin_form <- compute_exp_lin_form(variables_vector, coefficients_vector)
  sigma_value <- compute_sigma_hat(variables_vector, coefficients_vector)
  # Implement the equation corresponding to (16)
  first_half <- ( ((-price_coefficient^2)*lin_form*sigma_value^2) +
                    (2*(price_coefficient^2)*(lin_form^2)*(sigma_value^3))
                  )*(variables_vector[4] - booking_cost)
  second_half <- (price_coefficient*lin_form*sigma_value^2
                  )+(price_coefficient*lin_form*(sigma_value^2))

  return_value <- first_half + second_half
  #print(paste("New Second PD of f in CSPDF: ", return_value))
  return(return_value)
}

# Function to compute the 1st partial derivative of c w.r.t. the price: (17)
compute_first_pd_of_c <- function(loan_amount_vector, variables_vector,
                                  coefficients_vector){
  # first compute sigma of z-k using (1)
  price_coefficient  = coefficients_vector[4] # it's assumed this is in the third place
  # first do for segment 1
  var_vec1 = variables_vector
  # set the two variables after the intercept to zero
  var_vec1[2] = 0; var_vec1[3] = 0
  lin_f1 <- compute_exp_lin_form(var_vec1, coefficients_vector)
  sigma_v1 <- compute_sigma_hat(var_vec1, coefficients_vector)
  seg1 <- (price_coefficient*lin_f1*(sigma_v1)^2)*loan_amount_vector[1]
  # first do for segment 2
  var_vec2 = variables_vector
  # set the third to zero
  var_vec2[3] = 0
  lin_f2 <- compute_exp_lin_form(var_vec2, coefficients_vector)
  sigma_v2 <- compute_sigma_hat(var_vec2, coefficients_vector)
```

```r
  seg2 <- (price_coefficient*lin_f2*(sigma_v2)^2)*loan_amount_vector[2]
  # first do for segment 1
  var_vec3 = variables_vector
  # set the second to zero
  var_vec3[2] = 0
  lin_f3 <- compute_exp_lin_form(var_vec3, coefficients_vector)
  sigma_v3 <- compute_sigma_hat(var_vec3, coefficients_vector)
  seg3 <- (price_coefficient*lin_f1*(sigma_v3)^2)*loan_amount_vector[3]

  # Implement the equation corresponding to (15)
  return_value <- seg1 + seg2 + seg3
  #print(paste("New PD of C in CFPDC: ", return_value))
  return(return_value)
}

# Function to compute the 1st partial derivative of c w.r.t. the price: (17)
compute_second_pd_of_c <- function(loan_amount_vector, variables_vector,
                                   coefficients_vector){
  # first compute sigma of z-k using (1)
  price_coefficient  = coefficients_vector[4] # it's assumed this is in the third place
  # first do for segment 1
  var_vec1 = variables_vector
  # set the two variables after the intercept to zero
  var_vec1[2] = 0; var_vec1[3] = 0
  lin_f1 <- compute_exp_lin_form(var_vec1, coefficients_vector)
  sigma_v1 <- compute_sigma_hat(var_vec1, coefficients_vector)
  seg1 <- (
    (-price_coefficient^2)*lin_f1*(sigma_v1)^2)*loan_amount_vector[1] +
    ( 2*(price_coefficient^2)*(lin_f1^2)*(sigma_v1^2)*loan_amount_vector[1]  )
  # first do for segment 2
  var_vec2 = variables_vector
  # set the third to zero
  var_vec2[3] = 0
  lin_f2 <- compute_exp_lin_form(var_vec2, coefficients_vector)
  sigma_v2 <- compute_sigma_hat(var_vec2, coefficients_vector)
  seg2 <-  (
    (-price_coefficient^2)*lin_f2*(sigma_v2)^2)*loan_amount_vector[2] +
    ( 2*(price_coefficient^2)*(lin_f2^2)*(sigma_v2^2)*loan_amount_vector[2]  )
  # first do for segment 1
  var_vec3 = variables_vector
  # set the second to zero
  var_vec3[2] = 0
  lin_f3 <- compute_exp_lin_form(var_vec3, coefficients_vector)
  sigma_v3 <- compute_sigma_hat(var_vec3, coefficients_vector)
  seg3 <-  (
    (-price_coefficient^2)*lin_f3*(sigma_v3)^2)*loan_amount_vector[3] +
    ( 2*(price_coefficient^2)*(lin_f3^2)*(sigma_v3^2)*loan_amount_vector[3]  )

  # Implement the equation corresponding to (15)
  return_value <- seg1 + seg2 + seg3
  #print(paste("New PD of C in CFPDC: ", return_value))
  return(return_value)
}
```

```
compute_c <- function(loan_amount_vector, variables_vector, coefficients_vector, B){
  # first compute sigma of z-k using (1)
  price_coefficient  = coefficients_vector[4] # it's assumed this is in the third place
  # first do for segment 1
  var_vec1 = variables_vector
  # set the two variables after the intercept to zero
  var_vec1[2] = 0; var_vec1[3] = 0
  sigma_v1 <- compute_sigma_hat(var_vec1, coefficients_vector)
  seg1 <- (sigma_v1)*loan_amount_vector[1]
  # first do for segment 2
  var_vec2 = variables_vector
  # set the two variables after the intercept to zero
  var_vec2[2] = 0; var_vec2[3] = 0
  sigma_v2 <- compute_sigma_hat(var_vec2, coefficients_vector)
  seg2 <- sigma_v2*loan_amount_vector[2]
  # first do for segment 1
  var_vec3 = variables_vector
  # set the two variables after the intercept to zero
  var_vec3[2] = 0; var_vec3[3] = 0
  sigma_v3 <- compute_sigma_hat(var_vec3, coefficients_vector)
  seg3 <- (sigma_v3)*loan_amount_vector[3]

  # Implement the equation corresponding to (15)
  return_value <- seg1 + seg2 + seg3 - B
  #print(variables_vector)
  #print(paste("Computed C in CC: ", return_value))
  return(return_value)
}
```

```
# Function to compute the new lambda
compute_new_lambda <- function(loan_amount_vector, booking_cost,
                               variables_vector, coefficients_vector, B){
  price_var = variables_vector[4]
  first_half = (
    compute_first_pd_of_f(
    booking_cost, variables_vector,coefficients_vector
    ) )/compute_first_pd_of_c(loan_amount_vector, variables_vector, coefficients_vector)
  second_half = (
    (
      compute_second_pd_of_f(
      booking_cost, variables_vector, coefficients_vector
      ) - compute_second_pd_of_c(loan_amount_vector, variables_vector, coefficients_vector)
    )*compute_c(
        loan_amount_vector, variables_vector, coefficients_vector, B
        ) )/(
          compute_first_pd_of_c(loan_amount_vector, variables_vector, coefficients_vector
                                )^2)
  return_value = first_half - second_half
  #print(paste("New Lambda in CNL: ", return_value))
  return(return_value)
}
# Function to compute the new price
compute_new_price <- function(
```

```
  lambda, loan_amount_vector, booking_cost, variables_vector, coefficients_vector
  ){
  price_var = variables_vector[4]
  first_half=(
  lambda*compute_first_pd_of_c(loan_amount_vector, variables_vector, coefficients_vector)
    ) - compute_first_pd_of_f(booking_cost, variables_vector, coefficients_vector)
  #print(paste("First Half in CNP: ", first_half))
  second_half=compute_second_pd_of_f(booking_cost, variables_vector, coefficients_vector) -
    (lambda * compute_second_pd_of_c(loan_amount_vector, variables_vector,
                                     coefficients_vector) )
  #print(paste("Second Half in CNP: ", second_half))
  return_value = (first_half/second_half) #+ price_var
  #print(variables_vector)
  #print(paste("New Price in CNP: ", return_value))
  return(return_value)
}

# Function to check if the minimum has been achieved: (11)
check_convergence <- function(new_price, lambda, loan_amount_vector, booking_cost, variables_vector, coe
  price_var = variables_vector[4]
  first_half = - (
  lambda*compute_first_pd_of_c(loan_amount_vector, variables_vector, coefficients_vector)
    ) + compute_first_pd_of_f(booking_cost, variables_vector, coefficients_vector)
  #print(paste("First Half in CNP: ", first_half))
  second_half=compute_second_pd_of_f(booking_cost, variables_vector, coefficients_vector) -
    (lambda * compute_second_pd_of_c(loan_amount_vector, variables_vector,
                                     coefficients_vector) )
  #print(paste("Second Half in CNP: ", second_half)
  return_value = (first_half) + (new_price-price_var)*second_half
  return(return_value)
}
```

```
# Loop to obtain the new values
data_values_s1 <- data.frame(index = c(1), lambda = c(1), price = c(1), conv = c(1), secp = c(1))
data_values_s2 <- data.frame(index = c(1), lambda = c(1), price = c(1), conv = c(1), secp = c(1))
data_values_s3 <- data.frame(index = c(1), lambda = c(1), price = c(1), conv = c(1), secp = c(1))

for (k in 1:K){
  # Run for segment 1
  if(variables_vector[2]==0 & variables_vector[3]==0){

    new_lambda <- compute_new_lambda(loan_amount_vector, booking_cost,
                                     variables_vector, coefficients_vector, B)
    new_x_value <- compute_new_price(new_lambda, loan_amount_vector, booking_cost, variables_vector, coe
    conv_value <- check_convergence(new_x_value, lambda, loan_amount_vector, booking_cost, variables_vec
    sec_p_of_f <- compute_second_pd_of_f(booking_cost, variables_vector, coefficients_vector)
    # update the vectors
    variables_vector[4] <- new_x_value;
    data_values_s1[k, ] <- c(k, new_lambda, new_x_value, abs(conv_value), sec_p_of_f)
  }else{
    print("Wrong configuration of segment 1 variables")
    break;
  }
```

```r
  # Run for segment 2 by updating the variable vector
  variables_vector[2] = 1
  ## check the conditions before running
  if(variables_vector[2]==1 & variables_vector[3]==0){

    new_lambda <- compute_new_lambda(loan_amount_vector, booking_cost,
                                     variables_vector, coefficients_vector, B)
    new_x_value <- compute_new_price(new_lambda, loan_amount_vector, booking_cost, variables_vector, coe
    conv_value <- check_convergence(new_x_value, lambda, loan_amount_vector, booking_cost, variables_vec
    sec_p_of_f <- compute_second_pd_of_f(booking_cost, variables_vector, coefficients_vector)
    # update the vectors
    variables_vector[4] <- new_x_value;
    data_values_s2[k, ] <- c(k, new_lambda, new_x_value, abs(conv_value), sec_p_of_f)
  }else{
    print("Wrong configuration of segment 2 variables")
    break;
  }

  # Run for segment 3 by updating the variable vector
  variables_vector[3] = 1
  variables_vector[2] = 0
  ## check the conditions before running
  if(variables_vector[2]==0 & variables_vector[3]==1){

    new_lambda <- compute_new_lambda(loan_amount_vector, booking_cost,
                                     variables_vector, coefficients_vector, B)
    new_x_value <- compute_new_price(new_lambda, loan_amount_vector, booking_cost, variables_vector, coe
    conv_value <- check_convergence(new_x_value, lambda, loan_amount_vector, booking_cost, variables_vec
    sec_p_of_f <- compute_second_pd_of_f(booking_cost, variables_vector, coefficients_vector)
    # update the vectors
    variables_vector[4] <- new_x_value;
    data_values_s3[k, ] <- c(k, new_lambda, new_x_value, abs(conv_value), sec_p_of_f)
  }else{
    print("Wrong configuration of segment 3 variables")
    break;
  }
  # Reset to segment 1
  variables_vector[3] = 0
  variables_vector[2] = 0

}
```

```r
data_values_s1 <- data_values_s1 %>% mutate(segment = factor(rep(1, K)))
data_values_s2 <- data_values_s2 %>% mutate(segment = factor(rep(2, K)))
data_values_s3 <- data_values_s3 %>% mutate(segment = factor(rep(3, K)))
head(data_values_s1)
```

```
##   index lambda price conv          secp segment
## 1     1    NaN   NaN  NaN -0.0001877969       1
## 2     2    NaN   NaN  NaN           NaN       1
## 3     3    NaN   NaN  NaN           NaN       1
## 4     4    NaN   NaN  NaN           NaN       1
## 5     5    NaN   NaN  NaN           NaN       1
```

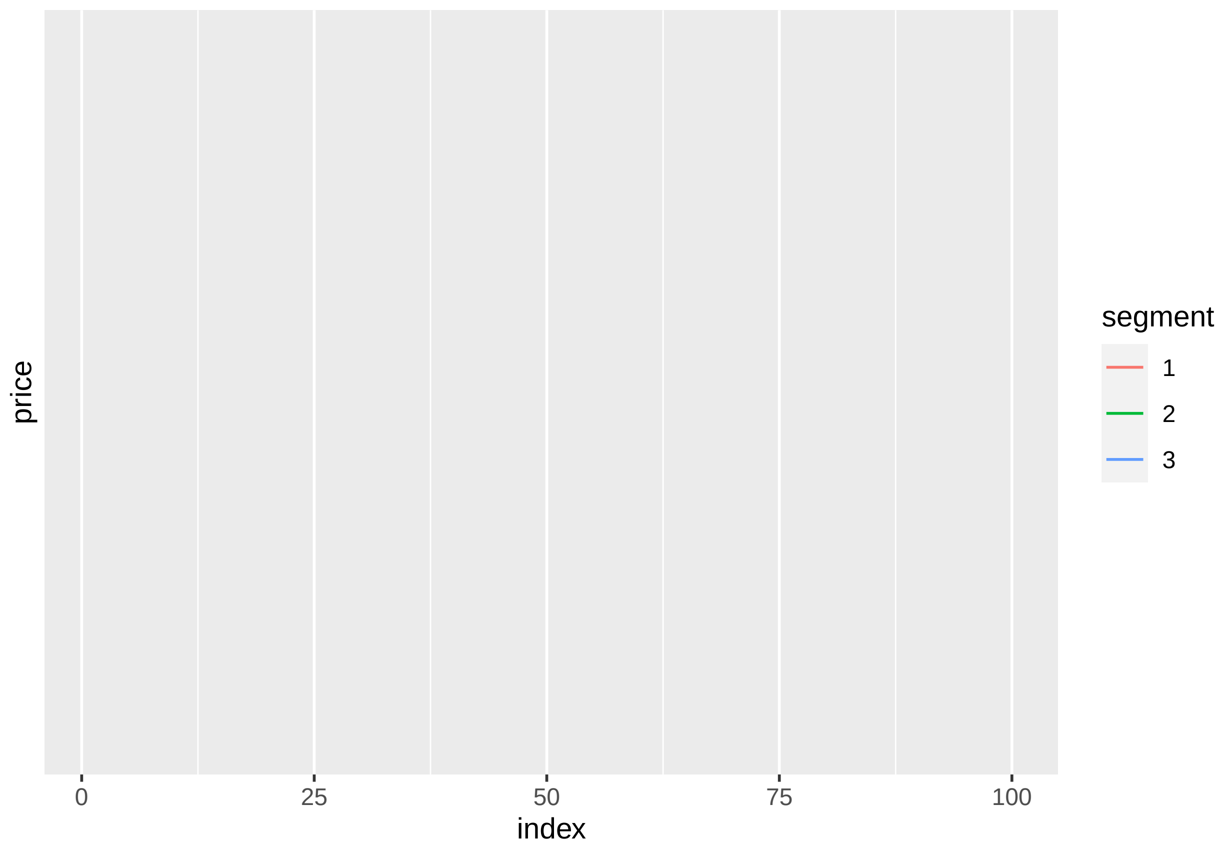
```
## 6     6    NaN   NaN  NaN           NaN       1
```

```r
merged_data_values <- rbind(data_values_s1, data_values_s2, data_values_s3)
head(merged_data_values)
```

```
##   index lambda price conv          secp segment
## 1     1    NaN   NaN  NaN -0.0001877969       1
## 2     2    NaN   NaN  NaN           NaN       1
## 3     3    NaN   NaN  NaN           NaN       1
## 4     4    NaN   NaN  NaN           NaN       1
## 5     5    NaN   NaN  NaN           NaN       1
## 6     6    NaN   NaN  NaN           NaN       1
```

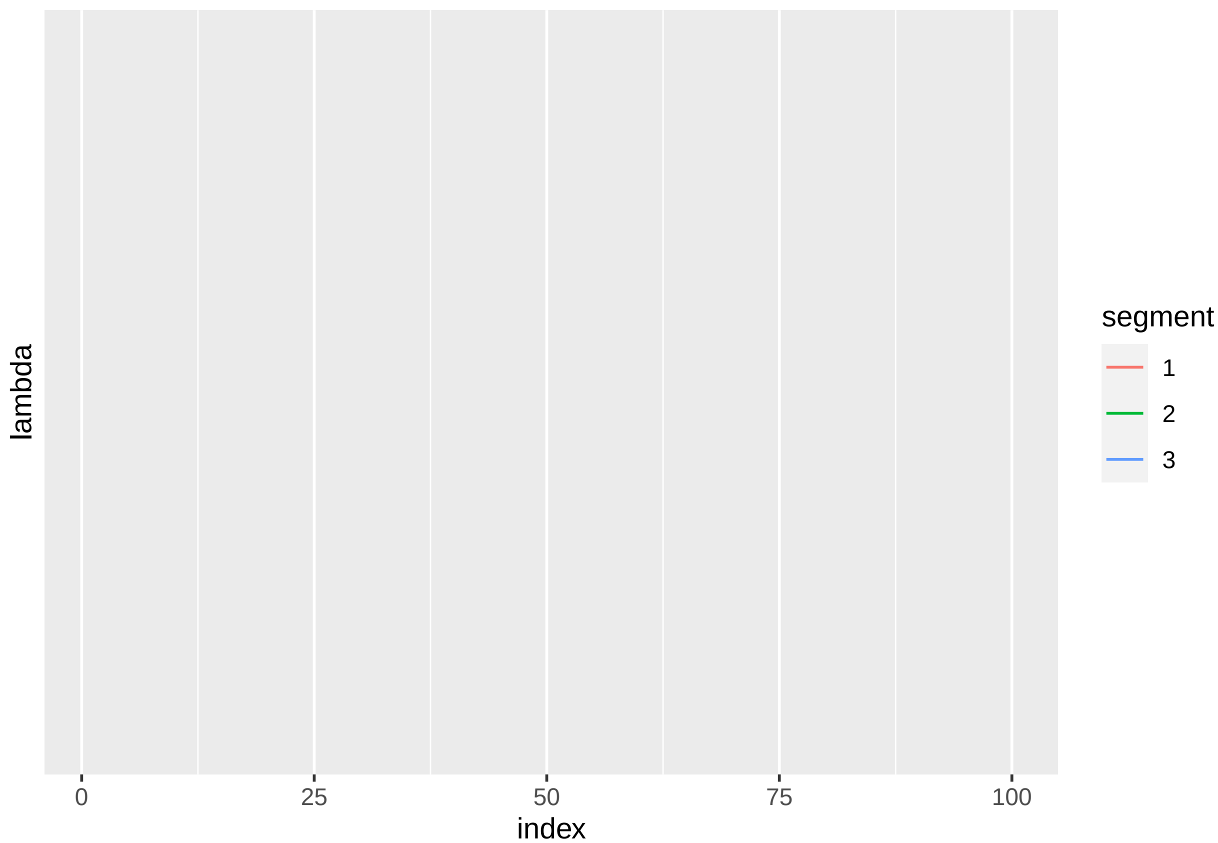
```r
library(ggplot2)
ggplot(data = merged_data_values) + geom_line(aes(x = index, y = price, color = segment))
```

```
## Warning: Removed 300 row(s) containing missing values (geom_path).
```




```r
ggplot(data = merged_data_values) + geom_line(aes(x = index, y = lambda, color = segment))
```

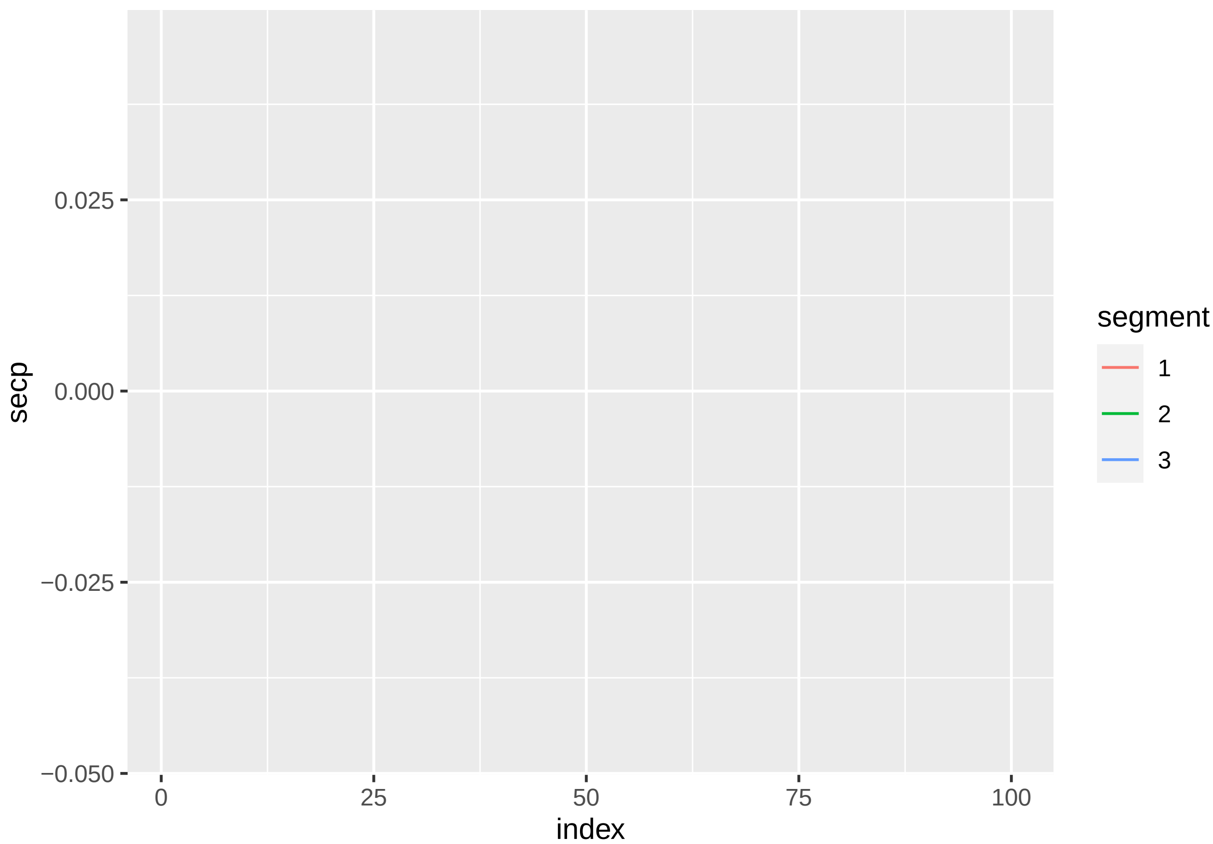
```
## Warning: Removed 300 row(s) containing missing values (geom_path).
```




```
ggplot(data = merged_data_values) + geom_line(aes(x = index, y = secp, color = segment))
```

```
## Warning: Removed 299 row(s) containing missing values (geom_path).
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```


0.025
0.000
−0.025
−0.050
secp
0
25
50
75
100
index
segment
1
2
3