Assignment 3

Gao Haochun A0194525Y Ge Siqi A0194550A Wang Pei A0194486M
Wei Yifei A0203451W

4/3/2021

Question 1

The sales department plans to run a large promotion. They provide you data obtained from a small trial ("hw3q1.csv"). They want to understand the effectiveness of the promotion. (Hint: the effectiveness can be different for different types of consumers.)

```
df1 <- read.csv("hw3q1.csv")
df1$age <- as.factor(df1$age)  ## Observe 3 age categories: 20-30, 30-40, 40+
df1$received_coupon <- as.factor(df1$received_coupon)
df1$after <- as.factor(df1$after)
summary(df1)
```

```
##     age       after    received_coupon    revenue
##  20-30:4000   0:6000   0:6000          Min.   : 1.54
##  30-40:4000   1:6000   1:6000          1st Qu.: 8.60
##  40+  :4000                           Median :10.19
##                                       Mean   :10.23
##                                       3rd Qu.:11.84
##                                       Max.   :19.40
```

```
df1_2030 <- df1 %>% filter(age == "20-30")
df1_3040 <- df1 %>% filter(age == "30-40")
df1_40 <- df1 %>% filter(age == "40+")

nrow(df1_2030[df1_2030$after==1,])/nrow(df1_2030)  # 50% each before and after
```

```
## [1] 0.5
```

```
nrow(df1_3040[df1_3040$after==1,])/nrow(df1_3040)  # 50% each before and after
```

```
## [1] 0.5
```

```
nrow(df1_40[df1_40$after==1,])/nrow(df1_40)  # 50% each for before and after
```

```
## [1] 0.5
```

```
nrow(df1_2030[df1_2030$received_coupon==1,])/nrow(df1_2030)  # 50% each treatment and control
```

```
## [1] 0.5
```

```
nrow(df1_3040[df1_3040$received_coupon==1,])/nrow(df1_3040)  # 50% each treatment and control
```

```
## [1] 0.5
```

```
nrow(df1_40[df1_40$received_coupon==1,])/nrow(df1_40)  # 50% each for treatment and control
```

```
## [1] 0.5
```

Part I

Calculate the treatment effects. Verify your results using felm. Are they the same as before-vs-after comparisons or treated-vs-non-treated comparisons?

Calculate the treatment effects using difference-in-differences (DID) estimator:

$$treatment \;\; effect = (Y_{treated,after} - Y_{nontreated,after}) - (Y_{treated,before} - Y_{nontreated,before})$$

```
# Treatment effect for all types of customers
overall_diff <- df1 %>% group_by(received_coupon, after) %>% summarise(rev_mean = mean(revenue))
```

```
## `summarise()` regrouping output by 'received_coupon' (override with `.groups` argument)
```

```
overall_diff
```

```
## # A tibble: 4 x 3
## # Groups:   received_coupon [2]
##   received_coupon after rev_mean
##   <fct>           <fct>    <dbl>
## 1 0               0         9.35
## 2 0               1        10.5
## 3 1               0         9.49
## 4 1               1        11.5
```

```
# Treatment effect for all types of customers
(11.549727 - 10.535157) - (9.493847 - 9.348597)   # 0.86932
```

```
## [1] 0.86932
```

```
# Treatment effect for difference types of customers (breakdown)
sep_diff <- df1 %>% group_by(age, received_coupon, after) %>% summarise(rev_mean = mean(revenue))
```

```
## `summarise()` regrouping output by 'age', 'received_coupon' (override with `.groups` argument)
```

```
sep_diff
```

```
## # A tibble: 12 x 4
## # Groups:   age, received_coupon [6]
##    age   received_coupon after rev_mean
##    <fct> <fct>           <fct>    <dbl>
##  1 20-30 0               0        10.3
##  2 20-30 0               1        11.8
##  3 20-30 1               0        10.4
##  4 20-30 1               1        12.3
##  5 30-40 0               0         8.46
##  6 30-40 0               1         9.11
##  7 30-40 1               0         8.77
##  8 30-40 1               1        10.3
##  9 40+   0               0         9.27
## 10 40+   0               1        10.7
## 11 40+   1               0         9.34
## 12 40+   1               1        12.0
```

```
# Treatment effect for age group 20-30 using DID:
(12.32244 - 11.78103) - (10.36813 - 10.31286)   # 0.48614
```

```
## [1] 0.48614
```

```
# Treatment effect for age group 30-40 using DID:
(10.32048 - 9.10888) - (8.77142 - 8.45867)   # 0.89885
```

```
## [1] 0.89885
```

```
# Treatment effect for age group 40+ using DID:
(12.00626 - 10.71556) - (9.34199 - 9.27426)   # 1.22297
```

```
## [1] 1.22297
```

Answer: - The average treatment effect for all types of customers i.e. across all age groups is 0.86932. - The treatment effects differ for customers from different age groups. - For customers aged from 20 to 30, the treatment effect is 0.48614. - For customers aged from 30 to 40, the treatment effect is 0.89885. For customers aged 40 and above, the treatment effect is 1.22297.

Verify the results using felm: felm can be used to verify the results of average treatment effect obtained using before-vs-after comparisons and treated-vs-nontreated comparisons:

1. felm to verify results of before-vs-after comparisons

```
lm1 <- felm(revenue ~ after | received_coupon, data = df1_2030)
lm2 <- felm(revenue ~ after | received_coupon, data = df1_3040)
lm3 <- felm(revenue ~ after | received_coupon, data = df1_40)
stargazer(lm1, lm2, lm3, type="text",omit.stat=c("LL","ser","f"), model.numbers=TRUE)
```

```
##
## ========================================
##                   Dependent variable:
##               ---------------------------
##                          revenue
##                  (1)       (2)       (3)
## ----------------------------------------
## after1       1.711***  1.100***  2.053***
##               (0.064)   (0.064)   (0.064)
##
## ----------------------------------------
## Observations   4,000     4,000     4,000
## R2             0.156     0.099     0.224
## Adjusted R2    0.156     0.099     0.223
## ========================================
## Note:          *p<0.1; **p<0.05; ***p<0.01
```

- For customers aged from 20 to 30, the coefficient of the binary variable after is 1.711.
- For customers aged from 30 to 40, the coefficient of the binary variable after is 1.100.
- For customers aged 40 and above, the coefficient of the binary variable after is 2.053.

Using before-vs-after comparisons: $mean(Y_{after}) - mean(Y_{before})$

```
# Age 20-30
# mean(df1_2030[df1_2030["after"]==1, "revenue"]) - mean(df1_2030[df1_2030["after"]==0, "revenue"])
(12.32244 + 11.78103) / 2 - (10.36813 + 10.31286) / 2   # 1.71124
```

```
## [1] 1.71124
```

```
# Age 30-40
# mean(df1_3040[df1_3040["after"]==1, "revenue"]) - mean(df1_3040[df1_3040["after"]==0, "revenue"])
(10.32048 + 9.10888) / 2 - (8.77142 + 8.45867) / 2   # 1.099635
```

```
## [1] 1.099635
```

```
# Age 40+
# mean(df1_40[df1_40["after"]==1, "revenue"]) - mean(df1_40[df1_40["after"]==0, "revenue"])
(12.00626 + 10.71556) / 2 - (9.34199 + 9.27426) / 2   # 2.052785
```

```
## [1] 2.052785
```

- For customers aged from 20 to 30, the before-vs-after comparison is 1.71124.
- For customers aged from 30 to 40, the before-vs-after comparison is 1.099635.
- For customers aged 40 and above, the before-vs-after comparison is 2.052785.

Thus, the results of felm which makes treatment (received_coupon) a fixed effect are consistent with the results of before-vs-after comparisons for different age groups.

2. felm to verify results of treated-vs-nontreated comparisons

```
lm4 <- felm(revenue ~ received_coupon | after, data = df1_2030)
lm5 <- felm(revenue ~ received_coupon | after, data = df1_3040)
lm6 <- felm(revenue ~ received_coupon | after, data = df1_40)
stargazer(lm4, lm5, lm6, type="text",omit.stat=c("LL","ser","f"), model.numbers=TRUE)
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                                revenue
##                         (1)        (2)        (3)
## ------------------------------------------------
## received_coupon1 0.298***  0.762***  0.679***
##                     (0.064)    (0.064)    (0.064)
##
## ------------------------------------------------
## Observations         4,000      4,000      4,000
## R2                   0.156      0.099      0.224
## Adjusted R2          0.156      0.099      0.223
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

- For customers aged from 20 to 30, the coefficient of the binary variable received_coupon is 0.298.
- For customers aged from 30 to 40, the coefficient of the binary variable received_coupon is 0.762.
- For customers aged 40 and above, the coefficient of the binary variable received_coupon is 0.679.

Using treated-vs-nontreated comparisons: avergae of Y_treated - average of Y_nontreated

```
# Age 20-30
# mean(df1_2030[df1_2030["received_coupon"]==1, "revenue"]) - mean(df1_2030[df1_2030["received_coupon"]
(10.36813 + 12.32244) / 2 - (10.31286 + 11.78103) / 2   # 0.29834
```

```
## [1] 0.29834
```

```
# Age 30-40
# mean(df1_3040[df1_3040["received_coupon"]==1, "revenue"]) - mean(df1_3040[df1_3040["received_coupon"]
(8.77142 + 10.32048) / 2 - (8.45867 + 9.10888) / 2   # 0.762175
```

```
## [1] 0.762175
```

```
# Age 40+
# mean(df1_40[df1_40["received_coupon"]==1, "revenue"]) - mean(df1_40[df1_40["received_coupon"]==0, "re
(9.34199 + 12.00626) / 2 - (9.27426 + 10.71556) / 2   # 0.679215
```

```
## [1] 0.679215
```

- For customers aged from 20 to 30, the treated-vs-nontreated comparison is 0.29834.
- For customers aged from 30 to 40, the treated-vs-nontreated comparison is 0.762175.
- For customers aged 40 and above, the treated-vs-nontreated comparison is 0.679215.

Thus, the results of felm which makes time (after) a fixed effect are consistent with the results of treated-vs-nontreated comparisons for different age groups.

Explanation: felm can verify before-vs-after comparisons and treated-vs-nontreated comparisons, but it cannot verify the treatement effects computed using difference-in-differences (DID) estimator. This is because felm fits a linear model by controlling for fixed effects but DID does not require to fix any variation.

- In the case of using felm to verify before-vs-after comparisons: felm makes treatment (received_coupon) a fixed effect. received_coupon controls for any unobserved treatment-specific heterogeneity that is is persistent over the time of experiment. Thus, there is no time trend in the model by setting received_coupon a fixed effect. This satisfies the assumption of before-vs-after comparisons. The felm results are consistent with before-vs-after comparisons.
- In the case of using felm to verify treated-vs-nontreated comparisons: felm makes time (after) a fixed effect. after controls for any unobserved time-specific trend that is common across all the customers. Thus, each customers have the same probability of being chosen as treatment and the experiment can be assumed to be random assignment of treatment. This satisfies the assumption of treated-vs-nontreated comparisons. The felm results are consistent with treated-vs-nontreated comparisons.

However, difference-in-differences does not require any of the above assumptions. The two assumption are violated in the sample data: there is a time trend and the assignment of treatment is not random. Thus, treatment effects measured using DID estimators are different from both before-vs-after comparisons and treated-vs-nontreated comparisons. felm is unable to verfiy the treatment effects computed using DID estimators.

Part II

The cost of the promotion is $0.8 per user. Provide specific advices to your colleague and estimate the gain (if any) from the promotion.

From Part I: - The average treatment effect for age group 20-30 is 0.48614, which is the revenue gain per user in the age group of 20-30. - The average treatment effect for age group 30-40 is 0.89885, which is the revenue gain per user in the age group of 30-40. - The average treatment effect for age group 40+ is 1.22297, which is the revenue gain per user in the age group of 40+.

Given that the cost of the promotion is $0.8 per user, below are the gain from the promotion for each age group of customers and some findings:

(1) It is profitable to run this promotion among age groups 30-40 and 40+, and 40+ is the most profitable age group. Profit gain per user for age group 30-40 = 0.89885 - 0.8 = 0.09885 Profit gain per user for age group 40+ = 1.22297 - 0.8 = 0.42297

(2) It is unprofitable to run this promotion among age group 20-30. Profit loss per user for age group 20-30 = 0.48614 - 0.8 = -0.31386

(3) Overall gain from the promotion based on the small trial of 12000 by combining 3 age groups is 4000 * 0.09885 + 4000 * 0.42297 + 4000 * (-0.31386) = 831.84

Here are some advice to our colleagues:

(1) It is favorable to run the promotion among age groups of 30-40 and 40+, and the company should promote the coverage of the promotion especially among those aged 40+ i.e. reach out more customers aged 30-40 and 40+, issue more vouchers to them, and encourage them to participate in the promotion.

(2) It is unfavorable to run the promotion for age group 20-30, and the company should reduce the coverage of the promotion among those aged 20-30 i.e. issue less vouchers to those aged 20-30.

(3) The company should issue vouchers based on customers' age. This can be achieved by showing ID to qualify for claiming the voucher or use internal data to filter the targeted customers and only issue vouchers to those aged 30-40 and 40+.

Question 2

Part I

Estimate consumer choice model using MLE. Discuss the meaning of your estimates.

```
data2 = read.csv("hw3q2.csv")
```

```
lik = function(beta0,beta1,beta2) {
  # likelihood function
  Pij =
  (data2$choices==0) * 0 +
    (data2$choices==1) * (beta1 + data2$p1 * beta0) +
    (data2$choices==2) * (beta2 + data2$p2 * beta0) -
    log(1+exp(beta1 + data2$p1*beta0)+exp(beta2 + data2$p2*beta0))

    return(-sum(Pij))
}

# estimation
mle1 = mle(lik,
    start = runif(3,-1,0),  # random initialization
    method = "L-BFGS-B",
    lower = c(-10, -10,-10),
  upper = c(-0.00001, 10, 10) # beta0, coeff of price, should be negative
)

summary(mle1)
```

```
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = lik, start = runif(3, -1, 0), method = "L-BFGS-B",
##     lower = c(-10, -10, -10), upper = c(-1e-05, 10, 10))
##
## Coefficients:
##         Estimate  Std. Error
## [1,] -0.07906081 0.002168384
## [2,]  0.66637973 0.018507126
## [3,] -1.66092588 0.025995260
##
## -2 log L: 166088.9
```

```
#  beta0       beta1        beta2
# -0.07906284  0.66639176  -1.66095162
```

Interpretation of coefficients:

- beta0 = -0.07906249 Price coefficient: One unit increase in price of a product, either JustGrab or GrabShare, will result in a 0.0791 units decrease in the utility level in average.
- beta1 = 0.66638432 The preference for JustGrab (j = 1) relative to not choosing any services is higher i.e. JustGrab is a more attractive option / generates higher utility than not choosing any (outside option).
- beta2 = -1.66094764 The preference for GrabShare (j = 2) relative to not choosing any services is lower i.e. GrabShare is a less attractive option / generates lower utility than not choosing any (outside option).

Part II

Discuss any potential endogeneity issue. Implement an estimation procedure that solves the endogeneity issue.

Endogeneity issue: Demand for GrabShare is unobserved, being an omitted variable which results in the endogeneity issue. During the morning peak hours when many people rush for school and work, the demand for GrabShare is high. This increase in demand leads to price surge, which results in higher price. Thus, demand for GrabShare is correlated with price2. Also, higher demand implies higher utility gained by customer once they obtain the GrabShare service, which affects customer choice. This suggests demand affects choice. Therefore, omitting the demand variable in the model will lead to omitted vairiable bias, resulting in endogeneity.

Proposed solution: Demand for GrabShare is reflected by surge2 which is demand-driven. Because the underlying demand is unobserved to us, we can include surge2 in the model instead. The solution is to use surge as an instrumental variable. Surge pricing only affects customer's utility level / choice only through the final price. Therefore, surge price is a good instrumental variable when running regression of utility on price.

```
data2$sales = 1

mkt_share=aggregate(sales~p1+p2+surge_1+surge_2+choices, data=data2, FUN=sum)
mkt_share$total=ave(mkt_share$sales,
                    by=list(mkt_share$p1,mkt_share$p2,mkt_share$surge_1,mkt_share$surge_2),FUN=sum)
mkt_share$share=mkt_share$sales/mkt_share$total

iop_dta=subset(mkt_share,choices!=0)#inside options
oop_dta=subset(mkt_share,choices==0)[,c("p1","p2","share")]#outside option
colnames(oop_dta)=c("p1","p2","share0")
lm_dta=merge(oop_dta,iop_dta)#merge inside options with outside option

lm_dta$log_odds=log(lm_dta$share/(lm_dta$share0))
lm_dta$price=(lm_dta$choices==1)*lm_dta$p1+
(lm_dta$choices==2)*lm_dta$p2   #the price for the focal product
lm_dta$product=as.factor(as.character(lm_dta$choices))
lm_dta[1:20,]
```

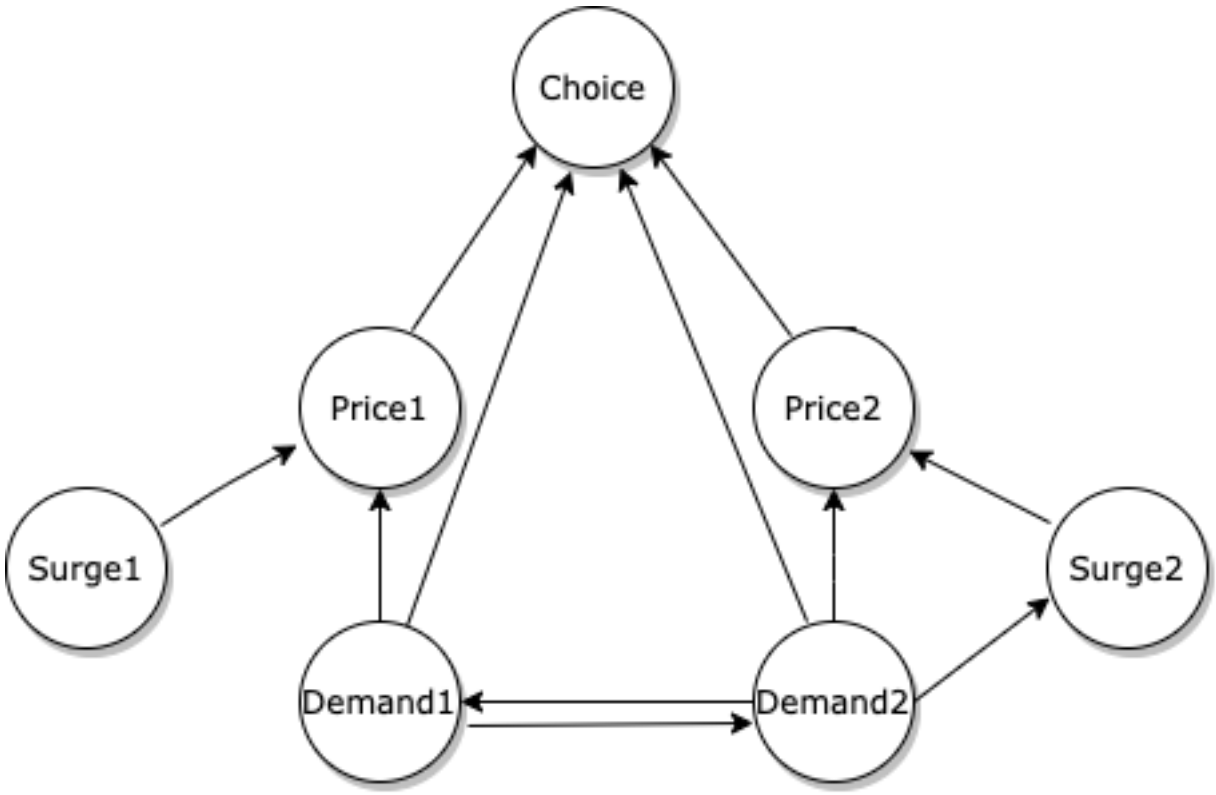


Figure 1: Proposed causal diagram

```
##       p1    p2    share0 surge_1 surge_2 choices sales total       share
## 1  10.99 10.99 0.6677937       2       1       2   206  1183 0.174133559
## 2  10.99 10.99 0.6677937       2       1       1   187  1183 0.158072697
## 3  10.99 12.99 0.8375421       2       2       1   180  1188 0.151515152
## 4  10.99 12.99 0.8375421       2       2       2    13  1188 0.010942761
## 5  10.99 13.99 0.7766990       2       2       1   226  1236 0.182847896
## 6  10.99 13.99 0.7766990       2       2       2    50  1236 0.040453074
## 7  10.99 14.99 0.6801988       2       2       2   207  1207 0.171499586
## 8  10.99 14.99 0.6801988       2       2       1   179  1207 0.148301574
## 9  10.99  4.99 0.7903861       2       0       1   256  1269 0.201733649
## 10 10.99  4.99 0.7903861       2       0       2    10  1269 0.007880221
## 11 10.99  5.99 0.7825397       2       0       1   230  1260 0.182539683
## 12 10.99  5.99 0.7825397       2       0       2    44  1260 0.034920635
## 13 10.99  6.99 0.6852459       2       0       2   200  1220 0.163934426
## 14 10.99  6.99 0.6852459       2       0       1   184  1220 0.150819672
## 15 10.99  8.99 0.7972973       2       1       2    12  1184 0.010135135
## 16 10.99  8.99 0.7972973       2       1       1   228  1184 0.192567568
## 17 10.99  9.99 0.7851562       2       1       2    38  1280 0.029687500
## 18 10.99  9.99 0.7851562       2       1       1   237  1280 0.185156250
## 19 11.99 10.99 0.4507730       2       1       2   127  1229 0.103336046
## 20 11.99 10.99 0.4507730       2       1       1   548  1229 0.445890968
##      log_odds price product
## 1  -1.3441568 10.99       2
## 2  -1.4409243 10.99       1
## 3  -1.7097859 10.99       1
## 4  -4.3377934 12.99       2
```

```
## 5  -1.4463983 10.99       1
## 6  -2.9549103 13.99       2
## 7  -1.3778043 14.99       2
## 8  -1.5231373 10.99       1
## 9  -1.3655733 10.99       1
## 10 -4.6081657  4.99       2
## 11 -1.4555770 10.99       1
## 12 -3.1094667  5.99       2
## 13 -1.4303112  6.99       2
## 14 -1.5136929 10.99       1
## 15 -4.3652195  8.99       2
## 16 -1.4207805 10.99       1
## 17 -3.2751567  9.99       2
## 18 -1.4446827 10.99       1
## 19 -1.4729776 10.99       2
## 20 -0.0108894 11.99       1
```

```r
lm_dta$surge = (lm_dta$choices==1)*(lm_dta$surge_1) + (lm_dta$choices==2)*(lm_dta$surge_2)

logistic_regression=lm(log_odds~price+product,lm_dta)
iv_reg=ivreg(log_odds ~ price+product|surge+product, data=lm_dta)  # surge as iv for price
stargazer(logistic_regression,iv_reg, type="text",omit.stat=c("f"))
```

```
##
## ===============================================================
##                                    Dependent variable:
##                                ---------------------------
##                                          log_odds
##                                     OLS       instrumental
##                                               variable
##                                     (1)            (2)
## ---------------------------------------------------------------
## price                               0.008        -0.213***
##                                    (0.047)        (0.054)
##
## product2                          -2.961***      -2.557***
##                                    (0.297)        (0.320)
##
## Constant                           -0.079        1.685***
##                                    (0.423)        (0.477)
##
## ---------------------------------------------------------------
## Observations                         157            157
## R2                                  0.410          0.324
## Adjusted R2                         0.402          0.316
## Residual Std. Error (df = 154)      1.784          1.909
## ===============================================================
## Note:                            *p<0.1; **p<0.05; ***p<0.01
```

The parameters are found to be:

- $beta_0 = -0.213$
- $beta_1 = 1.685$

- $beta_2 = 1.685 - 2.557 = -0.872$

Part III

Calculate optimal p_j if product j is the only product in the market, or if both products are in the market. Explain why the optimal prices are different.

```
beta0 = -0.213
beta1 = 1.685
beta2 = -0.872

# JustGrab only
profit_function1 = function(price1) {
  Profit1 = price1 * (exp(beta1 + price1 * beta0) / (1 + exp(beta1 + price1 * beta0)))

    return(-Profit1)
}

# GrabShare only
profit_function2 = function(price2) {
  Profit2 = price2 * (exp(beta2 + price2 * beta0) / (1 + exp(beta2 + price2 * beta0)))

    return(-Profit2)
}

p1 = optim(c(1), profit_function1, method = "BFGS")$par
p2 = optim(c(1), profit_function2, method = "BFGS")$par

# Prices
c(p1, p2)
```

```
## [1] 8.681130 5.326119
```

```
# Profits
c(-profit_function1(p1), -profit_function2(p2))
```

```
## [1] 3.9852603 0.6312835
```

- JustGrab is the only product: optimal price is 8.681130, with profit 3.9852603.
- GrabShare is the only product: optimal price is 5.326119. with profit 0.6312835.

```
beta0 = -0.213
beta1 = 1.685
beta2 = -0.872

profit_function3=function(prices){
  price1 = prices[1]
    price2 = prices[2]

    Profit1 = price1*(exp(beta1 + price1 * beta0) /
```

```
                        (1 + exp(beta1 + price1 * beta0) + exp(beta2 + price2 * beta0)))

    Profit2 = price2*(exp(beta2 + price2 * beta0) /
                          (1 + exp(beta1 + price1 * beta0) + exp(beta2 + price2 * beta0)))

    return(-(Profit1+Profit2))
}

res = optim(c(1, 1), profit_function3, method = "BFGS")$par
p1 = res[1]
p2 = res[2]

# Prices
c(p1,p2)
```

```
## [1] 8.842886 8.823795
```

```
# Profit
-profit_function3(res)
```

```
## [1] 4.147978
```

- Both products are in the market: optimal prices are 8.842886 and 8.823795 respectively, with total profit 4.147978.

Explanation for different prices: GrabShare and JustGrab are substitutes, so the company has incentive to increase price both products higher than single product to avoid canibalization effect.

Question 3

##Part I ### Assume that after the merger of firm 1 and firm 2, prices of product 1 and product 2 will be reoptimized as in a multiproduct pricing problem. But firm 3's price is held constant. Assume that lambda1 = lambda2 = 1. Solve the post-merger market outcome (prices, sales and costs, profits). Compare them with the pre-merger equilibrium.

Pre-merger:

```
set.seed(37)
#parameters
beta0=-0.03
beta1=2
beta2=1
beta3=2
N = 10
# global variables
price1=10
price2=10
price3=10

prob_1 = function(price1) {
```

```
  exp(beta1+price1*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))
}
prob_2 = function(price2) {
  exp(beta2+price2*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))
}
prob_3 = function(price3) {
  exp(beta3+price3*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))
}

cost = function(s) {
  (s-7)^2 + 50
}

# objective function of firm 1
profit_1=function(price1){
  produce1 = N*prob_1(price1)
    profit = produce1*price1 - cost(produce1)
    return(-profit)
}

# objective function of firm 2
profit_2=function(price2){
  produce2 = N*prob_2(price2)
    profit = produce2*price2 - cost(produce2)
    return(-profit)
}

# objective function of firm 3
profit_3=function(price3){
  produce3 = N*prob_3(price3)
    profit = produce3*price3 - cost(produce3)
    return(-profit)
}

# best response dynamic
i=0
error=1
Iter=1000
Tol=1e-10
prices = c(1,1,1)
while (i<Iter & error>Tol) {
    i=i+1
    price1=prices[1]
    price2=prices[2]
    price3=prices[3]

    prices[1] = optim(c(1), profit_1, method = "L-BFGS-B",
                lower=c(0),upper=c(Inf))$par

    prices[2] = optim(c(1), profit_2, method = "L-BFGS-B",
```

```
                lower=c(0),upper=c(Inf))$par

  prices[3] = optim(c(1), profit_3, method = "L-BFGS-B",
                lower=c(0),upper=c(Inf))$par

    error=abs(prices[1]-price1)+abs(prices[2]-price2)+abs(prices[3]-price3)
}

price1=prices[1]
price2=prices[2]
price3=prices[3]

c(i, error)
```

```
## [1] 1.550000e+02 7.053558e-11
```

```
# prices
prices
```

```
## [1] 42.47629 30.04197 42.47629
```

```
# sales
c(N*prob_1(price1), N*prob_2(price2), N*prob_3(price3))
```

```
## [1] 3.313246 1.769962 3.313246
```

```
# costs
sapply(c(N*prob_1(price1), N*prob_2(price2), N*prob_3(price3)), cost)
```

```
## [1] 63.59215 77.35330 63.59215
```

```
# profits
c(-profit_1(c(price1)),-profit_2(c(price2)),-profit_3(c(price3)))
```

```
## [1]  77.14225 -24.18015  77.14225
```

- The prices of the three products before merger are 42.47629, 30.04197, 42.47629.
- The sales of the three products before merger are 3.313246, 1.769962, 3.313246 (million).
- The cost of the three products before merger is 63.59215, 77.35330, 63.59215 (million).
- The profits of the three products before merger is 77.14225, -24.18015, 77.14225 (million).

Post-merger:

```
cost_merged = function(s1,s2,lambda1=1,lambda2=1) {
  return(lambda1*cost(s1) + lambda2*cost(s2))
}

profit_merged = function(param) {
  price1=param[1]
```

```
  price2=param[2]
  produce1=N*exp(beta1+price1*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))

  produce2=N*exp(beta2+price2*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))

  profit=produce1*price1 + produce2*price2 - cost_merged(produce1,produce2)

  return(-profit)
}

price3 = 42.47629
res = optim(c(1,1), profit_merged, method = "L-BFGS-B",
                lower=c(0,0),upper=c(Inf,Inf))$par

price1 = res[1]
price2 = res[2]
price3 = 42.47629

# prices
c(price1, price2, price3)
```

```
## [1] 50.38601 46.79862 42.47629
```

```
# sales
c(N*prob_1(price1), N*prob_2(price2),N*prob_3(price3))
```

```
## [1] 3.038511 1.244818 3.852256
```

```
# costs
c(cost_merged(N*prob_1(price1), N*prob_2(price2)), cost(N*prob_3(price3)))
```

```
## [1] 148.81552  59.90829
```

```
# profits
c(-profit_merged(c(price1, price1)), -profit_3(price3))
```

```
## [1]  62.29782 103.72126
```

If firm 3's price is held constant:

- The prices of the three products after merger are 50.38601 46.79862 42.47629.
- The sales of the three products after merger are 3.038511 1.244818 3.852256 (million).
- The cost of the post-merger firm and firm 3 after merger is 148.81552 59.90829 (million).
- The profits of the post-merger firm and firm 3 after merger is 52.9621 103.7213.

Compared with the pre-merger equilibrium: the prices of product 1 and 2 increase; sales of product 1 and 2 decreases, sales of product 3 increases; the sum of cost of firm 1 and 2 increases from 63.59215+77.35330 = 140.9454 to 148.81552 and the cost of firm 3 decrease. The profit of the merge increases from 77.14225-24.18015 = 52.9621 to 62.29782; the profit of firm 3 increases

Part II

In reality, firm 3 will respond as predicted by a simultaneous move pricing game.

a. Still assume that $\lambda_1 = \lambda_2 = 1$. Compute the post-merger market outcome and discuss how it differs from your answer in Part I.

```
i = 0
error=1
Iter=1000
Tol=1e-10
prices = c(1,1,1)

while (i < 150 & error > Tol) {
  price1 = prices[1]
  price2 = prices[2]
  price3 = prices[3]

  res1 = optim(c(1,1), profit_merged, method = "L-BFGS-B",
                lower=c(0,0),upper=c(Inf,Inf))$par
  prices[1] = res1[1]
  prices[2] = res1[2]
  prices[3] = optim(c(1), profit_3, method = "L-BFGS-B",
                lower=c(0),upper=c(Inf))$par

  i = i + 1
  error=abs(prices[1]-price1)+abs(prices[2]-price2)+abs(prices[3]-price3)
}

price1=prices[1]
price2=prices[2]
price3=prices[3]

c(i, error)
```

```
## [1] 1.500000e+02 7.991403e-10
```

```
# prices
prices
```

```
## [1] 51.45059 47.79469 45.91185
```

```
# sales
c(N*prob_1(price1),N*prob_2(price2),N*prob_3(price3))
```

```
## [1] 3.100972 1.273021 3.661517
```

```
# costs
c(cost_merged(N*prob_1(price1),N*prob_2(price2)), cost(N*prob_3(price3)))
```

```
## [1] 148.00072  61.14547
```

```r
# profits
c(-profit_merged(c(price1, price2)),-profit_3(c(price3)))
```

```
## [1]  72.38972 106.96156
```

If firm 3 responds as predicted by a simultaneous move pricing game:

- The prices of the three products after merger are 51.45059 47.79469 45.91185.
- The sales of the three products after merger are 3.100972 1.273021 3.661517(million).
- The cost of the merge and firm 3 after merger is 148.00072 61.14547 (million).
- The profits of the post-merger firm and firm 3 after merger is 72.38972 106.96156.

Compared with the answer in part I:

- The prices of product 1 and 2 increase and price of product 3 increases.
- The sales of product 1 and 2 increase and sales of price 3 decreases.
- The cost of the post-merger firm decreases and the cost of firm 3 increases.
- The profits of the post-merger firm increases and the profit of firm 3 decreases

b. Conduct a merger analysis for different values of λ_1 and λ_2.

```r
merger_analysis = function(lambda1, lambda2) {
  cost_merged = function(s1,s2) {
    return(lambda1*cost(s1) + lambda2*cost(s2))
  }

  profit_merged = function(param) {
    price1=param[1]
    price2=param[2]
    produce1=N*exp(beta1+price1*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))

    produce2=N*exp(beta2+price2*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))

    profit=produce1*price1 + produce2*price2 - cost_merged(produce1,produce2)

    return(-profit)
  }

  profit_3=function(price3){
    produce3 = N*exp(beta3+price3*beta0)/
    (1+exp(beta1+price1*beta0)+exp(beta2+price2*beta0)+exp(beta3+price3*beta0))

    profit = produce3*price3 - cost(produce3)
    return(-profit)
  }

  i = 0
  error=1
  Iter=1000
```

```
  Tol=1e-10
  prices = c(1,1,1)

  while (i < 150 & error > Tol) {
    price1 = prices[1]
    price2 = prices[2]
    price3 = prices[3]

    res1 = optim(c(1,1), profit_merged, method = "L-BFGS-B",
                 lower=c(0,0),upper=c(Inf,Inf))$par
    prices[1] = res1[1]
    prices[2] = res1[2]
    prices[3] = optim(c(1), profit_3, method = "L-BFGS-B",
                 lower=c(0),upper=c(Inf))$par

    i = i + 1
    error=abs(prices[1]-price1)+abs(prices[2]-price2)+abs(prices[3]-price3)
  }

  price1=prices[1]
  price2=prices[2]
  price3=prices[3]

  return (list(
    'prices'= prices,
    'sales' = c(N*prob_1(price1),N*prob_2(price2),N*prob_3(price3)),
    'costs' = c(cost_merged(N*prob_1(price1),N*prob_2(price2)), cost(N*prob_3(price3))),
    'profits'=c(-profit_merged(c(price1, price2)),-profit_3(c(price3)))
  ))
}

#check its consistency with 3.2.a
merger_analysis(1,1)$profits[1]
```

```
## [1] 72.38972
```

```
lambdas = seq(0.5,1.5, by = 0.1)

record = rep(c(0,0,0), 121)
record = array(record, dim = c(121,3))

row = 1
for (i in lambdas) {
  for (j in lambdas) {
    profit_aftermerge = merger_analysis(i,j)$profits[1]
    record[row,1] = i
    record[row,2] = j
    record[row,3] = profit_aftermerge
    row = row + 1
  }
}
#record
```

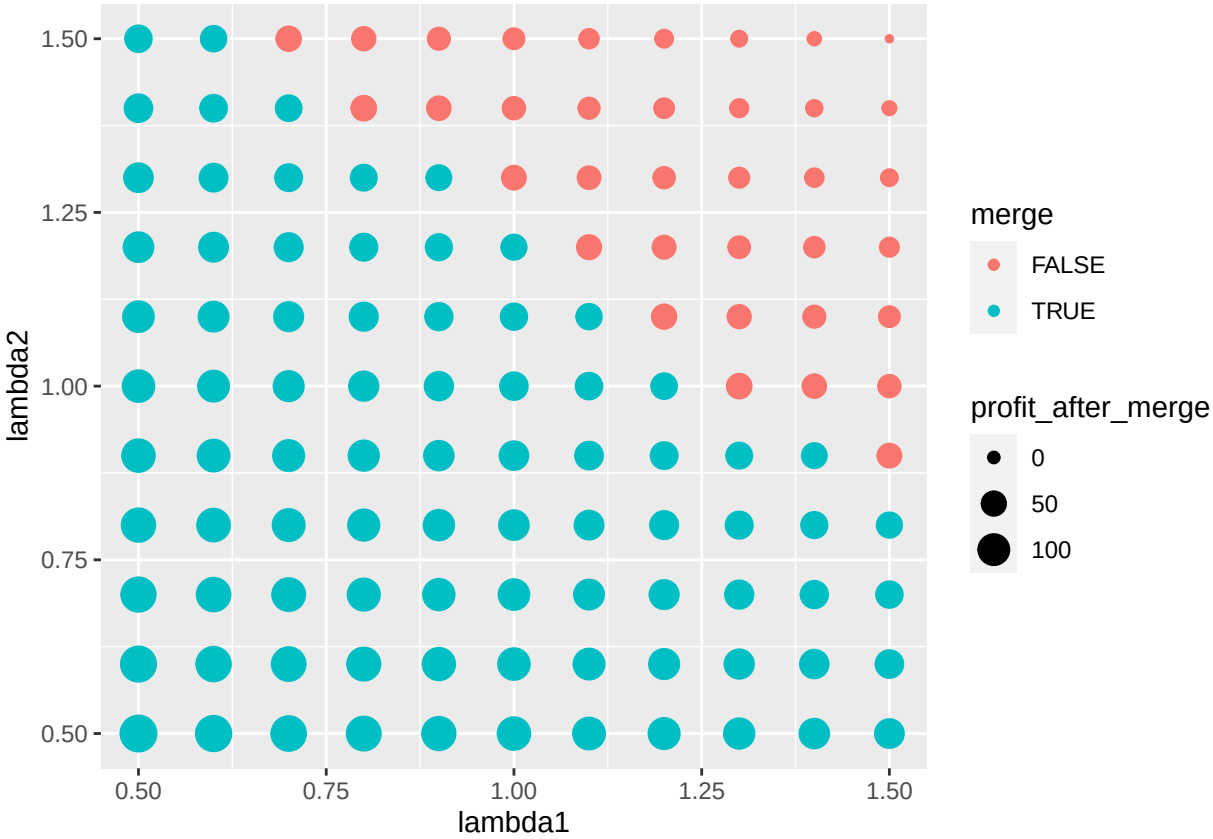
```
library("ggplot2")

record1 = as.data.frame(record)

names(record1) = c ("lambda1" , "lambda2", "profit_after_merge")
total_profit_before_merge = 77.14225-24.18015
record1$merge = record1$profit_after_merge > total_profit_before_merge
tail(record1, 10)
```

```
##     lambda1 lambda2 profit_after_merge merge
## 112     1.5     0.6          73.186456  TRUE
## 113     1.5     0.7          64.544783  TRUE
## 114     1.5     0.8          55.936216  TRUE
## 115     1.5     0.9          47.360992 FALSE
## 116     1.5     1.0          38.819318 FALSE
## 117     1.5     1.1          30.311371 FALSE
## 118     1.5     1.2          21.837307 FALSE
## 119     1.5     1.3          13.397253 FALSE
## 120     1.5     1.4           4.991308 FALSE
## 121     1.5     1.5          -3.380455 FALSE
```

```
ggplot(record1, aes(x=lambda1, y=lambda2)) + geom_point(aes(size=profit_after_merge, color = merge))
```



Profits for different values of λ_1 and λ_2 with the step of 0.1 from 0.5 to 1.5 are calculated. These profits are plotted in a scatter plot with x axis being the value of λ_1 and y axis being the value of λ_2.

The size of the point represents the quantity of the merger's profit and the color represents whether the profit of firm 1 and 2 increases after merger. Blue means increase which indicates that under the λ_1 and λ_2 of a blue point, the merge should be conducted while red means decrease which indicates that under the λ_1 and λ_2 of a red point, the merge should not be conducted.

Question 4

Part I

The effect of maximum capacity on stockpiling behaviour. Set the maximum inventory capacity as 5 and 20 (keep other parameters untouched), respectively. Describe and explain what you observe.

```
# original params
# set up
settings=list(K=3, # number of pack sizes
    n=c(2,6,0), # number of  units in size k, 0 is not buying
    Imax=20, # maximum inventory
    tol=1e-8, # error tolerance for convergence
    iter_max=10000  # maximum  number of iterations
    )

param=list(beta=0.99,
    alpha=4, # price sensitivity
    delta=10, # consumption utility
    c=0.05 # cost for holding inventory
    )

price=list(
    L=2, # number of price levels
    prices=data.frame(price_norm=c(2,5,0),price_prom=c(1.2,3,0)), #K-by-L, normal prices and promotion
    prob=c(0.84,0.16) # probability of different price levels.
    )

# fixed point iteration
value_function_iteration=function(settings,param,price){

    value_0=matrix(0,nrow=settings$Imax+1,ncol=price$L) # State vars are inventory and price
    error=settings$tol+1
    iteration=1
    start_time=Sys.time()

    while (error>=settings$tol & iteration<= settings$iter_max){
      # bellman_operator takes in curr_value and updates State
        value=Bellman_operator(value_0,settings,param,price)$value
        iteration=iteration+1
        error=max(abs(value-value_0))
        value_0=value
    }

    time_elapsed=Sys.time()-start_time
```

```
    solution=Bellman_operator(value_0,settings,param,price)

    output=list(time_elapsed,error,iteration,solution)
}

# fixed point operator that updates value function for any
Bellman_operator=function(value_0,settings,param,price){

  # 3 dims: num_Inventory x num_Price x num_Choice
    v_choice=array(0,dim=c(settings$Imax+1,price$L,settings$K))
    value=matrix(0,nrow=settings$Imax+1,ncol=price$L)
    choice=matrix(0,nrow=settings$Imax+1,ncol=price$L)
    inventory=c(0:settings$Imax)
    Ev=value_0%*%price$prob  # expected value

    for (k in 1:(settings$K)){
      # update inventory
        inventory_new0=inventory+settings$n[k]
        inventory_new=inventory_new0-1
        inventory_new[inventory_new<0]=0
        inventory_new[inventory_new>settings$Imax]=settings$Imax
        # inventory_new = min(settings$Imax, max(0, inventory_new0-1))

        # Calculate utility of curr inventory
        utility=param$delta*(inventory_new0>0)-param$c*inventory_new

        for (l in 1:price$L){
            v_choice[,l,k]=utility-param$alpha*price$prices[k,l]+param$beta*Ev[inventory_new+1]
        }
    }

    for (i in 1:(settings$Imax+1)){
        for (l in 1:price$L){
            value[i,l]=max(v_choice[i,l,])
            choice[i,l]=which.max(v_choice[i,l,])
        }
    }

    output=list(value=value,choice=choice)
    return(output)
}

results=value_function_iteration(settings,param,price)
results
```

```
## [[1]]
## Time difference of 0.565321 secs
##
## [[2]]
## [1] 9.995688e-09
##
## [[3]]
## [1] 2064
```

```
##
## [[4]]
## [[4]]$value
##           [,1]     [,2]
##  [1,] 717.8486 725.8486
##  [2,] 721.9373 728.3414
##  [3,] 725.6823 730.6638
##  [4,] 729.1146 732.8372
##  [5,] 732.2631 734.9388
##  [6,] 735.1643 736.9694
##  [7,] 737.8486 738.9297
##  [8,] 740.3414 740.8204
##  [9,] 742.6638 742.6638
## [10,] 744.8372 744.8372
## [11,] 746.9388 746.9388
## [12,] 748.9694 748.9694
## [13,] 750.9297 750.9297
## [14,] 752.8204 752.8204
## [15,] 754.6422 754.6422
## [16,] 756.3958 756.3958
## [17,] 758.0818 758.0818
## [18,] 759.7010 759.7010
## [19,] 761.2540 761.2540
## [20,] 762.7415 762.7415
## [21,] 764.1641 764.1641
##
## [[4]]$choice
##       [,1] [,2]
##  [1,]    2    2
##  [2,]    3    2
##  [3,]    3    2
##  [4,]    3    2
##  [5,]    3    2
##  [6,]    3    2
##  [7,]    3    2
##  [8,]    3    2
##  [9,]    3    3
## [10,]    3    3
## [11,]    3    3
## [12,]    3    3
## [13,]    3    3
## [14,]    3    3
## [15,]    3    3
## [16,]    3    3
## [17,]    3    3
## [18,]    3    3
## [19,]    3    3
## [20,]    3    3
## [21,]    3    3
```

Set maximum inventory capacity as 5

```
# Imax=5
new_settings=list(K=3, # number of pack sizes
    n=c(2,6,0), # number of  units in size k, 0 is not buying
    Imax=5, # maximum inventory
    tol=1e-8, # error tolerance for convergence
    iter_max=10000  # maximum  number of iterations
    )

results=value_function_iteration(new_settings, param, price)
results
```

```
## [[1]]
## Time difference of 0.3005459 secs
##
## [[2]]
## [1] 9.998416e-09
##
## [[3]]
## [1] 2027
##
## [[4]]
## [[4]]$value
##          [,1]     [,2]
## [1,] 685.8794 693.7952
## [2,] 690.2744 693.7952
## [3,] 693.8794 695.3422
## [4,] 697.0723 698.2759
## [5,] 700.1422 700.9952
## [6,] 703.0759 703.0759
##
## [[4]]$choice
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    2
## [3,]    3    1
## [4,]    3    1
## [5,]    3    1
## [6,]    3    3
```

Describe and explain the findings:

- When maximum inventory capacity = 5, all maximum expected sum of discounted utility (abbreviated as utility value below) at each current inventory levels and price promotions are lower than those when maximum inventory capacity = 20.
- From this output, we can conclude that higher inventory capacity gives higher utility values. This is because the utility value of choice 2 (large package) is higher than that of choice 1 (small package) i.e. larger package is more value-for-money. When max inventory = 20, the customer never opt for choice 1, but when max inventory = 5, the customer seldom have chances to opt for choice 2.

- For both cases of normal price and promotion, the purchase choices tend to be 3 (not buying) when current inventory is close to maximum. This is because individuals with smaller inventory capacity are more likely to replenish the stock more frequently than those with larger inventory capacity under same rate of consumption.

Part II

The effect of discount factors on stockpiling behaviour. Set discount factors as 0.5 and 0.99 (keep other parameters untouched), respectively. Describe and explain what you observe.

```
# discount=0.5
new_param_1=list(beta=0.5,
    alpha=4, # price sensitivity
    delta=10, # consumption utility
    c=0.05 # cost for holding inventory
    )
results=value_function_iteration(settings, new_param_1, price)
results
```

```
## [[1]]
## Time difference of 0.006103039 secs
##
## [[2]]
## [1] 9.313229e-09
##
## [[3]]
## [1] 32
##
## [[4]]
## [[4]]$value
##            [,1]     [,2]
##  [1,]  9.437333 12.63733
##  [2,] 14.974667 14.97467
##  [3,] 17.437333 17.43733
##  [4,] 18.618667 18.61867
##  [5,] 19.159333 19.15933
##  [6,] 19.379667 19.37967
##  [7,] 19.439833 19.43983
##  [8,] 19.419917 19.41992
##  [9,] 19.359958 19.35996
## [10,] 19.279979 19.27998
## [11,] 19.189990 19.18999
## [12,] 19.094995 19.09499
## [13,] 18.997497 18.99750
## [14,] 18.898749 18.89875
## [15,] 18.799374 18.79937
## [16,] 18.699687 18.69969
## [17,] 18.599844 18.59984
## [18,] 18.499922 18.49992
## [19,] 18.399961 18.39996
## [20,] 18.299980 18.29998
## [21,] 18.199990 18.19999
```

```
##
## [[4]]$choice
##       [,1] [,2]
##  [1,]    1    1
##  [2,]    3    3
##  [3,]    3    3
##  [4,]    3    3
##  [5,]    3    3
##  [6,]    3    3
##  [7,]    3    3
##  [8,]    3    3
##  [9,]    3    3
## [10,]    3    3
## [11,]    3    3
## [12,]    3    3
## [13,]    3    3
## [14,]    3    3
## [15,]    3    3
## [16,]    3    3
## [17,]    3    3
## [18,]    3    3
## [19,]    3    3
## [20,]    3    3
## [21,]    3    3
```

Observation:

- When discounting factor = 0.5, all utility values at each current inventory levels and price promotions are lower than those when discounting factor = 0.99.
- The customer only buys product when current stock is 0, and he only opt for choice 1 (small package) which is cheaper (thus larger utility for today).

Explanation:

- The lower the discounting factor, the more the customer cares about today's utility.
- Based on policy function and Bellman equation, when the discounting factor gets smaller, the maximum expected sum of discounted utility gets smaller.

Part III

The effect of price sensitivity on stockpiling behaviour. Change the value of price coefficient (keep other parameters untouched), respectively. Describe and explain what you observe.

Set price coefficient larger

```
# price coeff larger
new_param_2=list(beta=0.99,
    alpha=12, # price sensitivity: 4 -> 12
    delta=10, # consumption utility
    c=0.05 # cost for holding inventory
```

```
    )

new_param_3=list(beta=0.99,
    alpha=2, # price sensitivity: 4 -> 2
    delta=10, # consumption utility
    c=0.05 # cost for holding inventory
    )

results_1=value_function_iteration(settings, new_param_2, price)
results_2=value_function_iteration(settings, new_param_3, price)
c(results_1, results_2)
```

```
## [[1]]
## Time difference of 0.415323 secs
##
## [[2]]
## [1] 9.921223e-09
##
## [[3]]
## [1] 1933
##
## [[4]]
## [[4]]$value
##           [,1]     [,2]
##  [1,] 252.9897 268.9612
##  [2,] 262.9897 276.1022
##  [3,] 272.3868 282.9347
##  [4,] 281.2337 289.4932
##  [5,] 289.5797 295.8066
##  [6,] 297.4702 301.9885
##  [7,] 304.9612 308.0587
##  [8,] 312.1022 314.0181
##  [9,] 318.9347 319.8679
## [10,] 325.4932 325.6092
## [11,] 331.8066 331.8066
## [12,] 337.9885 337.9885
## [13,] 344.0587 344.0587
## [14,] 350.0181 350.0181
## [15,] 355.8679 355.8679
## [16,] 361.6092 361.6092
## [17,] 367.2431 367.2431
## [18,] 372.7707 372.7707
## [19,] 378.1930 378.1930
## [20,] 383.5111 383.5111
## [21,] 388.7259 388.7259
##
## [[4]]$choice
##       [,1] [,2]
##  [1,]    3    2
##  [2,]    3    2
##  [3,]    3    2
##  [4,]    3    2
##  [5,]    3    2
```

```
##  [6,]    3    2
##  [7,]    3    2
##  [8,]    3    2
##  [9,]    3    2
## [10,]    3    2
## [11,]    3    3
## [12,]    3    3
## [13,]    3    3
## [14,]    3    3
## [15,]    3    3
## [16,]    3    3
## [17,]    3    3
## [18,]    3    3
## [19,]    3    3
## [20,]    3    3
## [21,]    3    3
##
##
## [[5]]
## Time difference of 0.5218818 secs
##
## [[6]]
## [1] 9.995347e-09
##
## [[7]]
## [1] 2064
##
## [[8]]
## [[8]]$value
##           [,1]     [,2]
##  [1,] 848.2221 851.9881
##  [2,] 850.3365 853.1083
##  [3,] 852.2221 854.1672
##  [4,] 853.9080 855.1655
##  [5,] 855.4181 856.1038
##  [6,] 856.7726 856.9828
##  [7,] 857.9881 857.9881
##  [8,] 859.1083 859.1083
##  [9,] 860.1672 860.1672
## [10,] 861.1655 861.1655
## [11,] 862.1038 862.1038
## [12,] 862.9828 862.9828
## [13,] 863.8030 863.8030
## [14,] 864.5649 864.5649
## [15,] 865.2693 865.2693
## [16,] 865.9166 865.9166
## [17,] 866.5074 866.5074
## [18,] 867.0424 867.0424
## [19,] 867.5219 867.5219
## [20,] 867.9467 867.9467
## [21,] 868.3173 868.3173
##
## [[8]]$choice
##       [,1] [,2]
```

```
##  [1,]    1    2
##  [2,]    3    2
##  [3,]    3    2
##  [4,]    3    2
##  [5,]    3    2
##  [6,]    3    2
##  [7,]    3    3
##  [8,]    3    3
##  [9,]    3    3
## [10,]    3    3
## [11,]    3    3
## [12,]    3    3
## [13,]    3    3
## [14,]    3    3
## [15,]    3    3
## [16,]    3    3
## [17,]    3    3
## [18,]    3    3
## [19,]    3    3
## [20,]    3    3
## [21,]    3    3
```

Observation: - When price sensitivity is larger, all utility values at each current inventory levels and price promotions decrease. - The customer only opt for choice 2 (buy large package) when current inventory is not close to full, and he always opt for choice 3 (not buying) when there is no promotions even though the current inventory is 0.

Explanation:

- When the price sensitivity is lower, the customer cares less about the price, thus the price dis-utility is lower. Based on the Bellman Operator, as a result, when the price dis-utility becomes less negative, the maximum expected sum of discounted utility increases.

Part IV

Note that in the slides, we assume that there are two price levels: normal price and promotion price, which is 60% discount. In this exercise, we assume that there are three price levels: normal price, 70% discount and 40% discount. The probabilities of each price levels are 0.8, 0.1 and 0.1 respectively. Please change the code correspondingly, keeping other parameters untouched. Report and briefly discuss your results.

```
new_price=list(
    L=3, # number of price levels
    prices=data.frame(price_norm=c(2,5,0),discount_1=c(1.4,3.5,0),discount_2=c(0.8, 2, 0)),
    # discount_1: 70%, discount_2: 40%
    #K-by-L, normal prices and promotion prices
    prob=c(0.8, 0.1, 0.1) # probability of different price levels.
    )

results=value_function_iteration(settings, param, new_price)
results
```

```
## [[1]]
## Time difference of 1.118968 secs
##
## [[2]]
## [1] 9.994437e-09
##
## [[3]]
## [1] 2064
##
## [[4]]
## [[4]]$value
##           [,1]     [,2]     [,3]
##  [1,] 751.3735 757.1542 763.1542
##  [2,] 755.5983 759.5737 765.5737
##  [3,] 759.3735 761.8390 767.8390
##  [4,] 762.7619 763.9561 769.9561
##  [5,] 765.8147 765.9344 771.9344
##  [6,] 768.5743 768.5743 773.7826
##  [7,] 771.1542 771.1542 775.5087
##  [8,] 773.5737 773.5737 777.1196
##  [9,] 775.8390 775.8390 778.6218
## [10,] 777.9561 777.9561 780.0556
## [11,] 779.9344 779.9344 781.4250
## [12,] 781.7826 781.7826 782.7308
## [13,] 783.5087 783.5087 783.9735
## [14,] 785.1196 785.1196 785.1537
## [15,] 786.6218 786.6218 786.6218
## [16,] 788.0556 788.0556 788.0556
## [17,] 789.4250 789.4250 789.4250
## [18,] 790.7308 790.7308 790.7308
## [19,] 791.9735 791.9735 791.9735
## [20,] 793.1537 793.1537 793.1537
## [21,] 794.2722 794.2722 794.2722
##
## [[4]]$choice
##       [,1] [,2] [,3]
##  [1,]    1    2    2
##  [2,]    3    2    2
##  [3,]    3    2    2
##  [4,]    3    2    2
##  [5,]    3    2    2
##  [6,]    3    3    2
##  [7,]    3    3    2
##  [8,]    3    3    2
##  [9,]    3    3    2
## [10,]    3    3    2
## [11,]    3    3    2
## [12,]    3    3    2
## [13,]    3    3    2
## [14,]    3    3    2
## [15,]    3    3    3
## [16,]    3    3    3
## [17,]    3    3    3
## [18,]    3    3    3
```

```
## [19,]    3    3    3
## [20,]    3    3    3
## [21,]    3    3    3
```

```
# Expected price for current settings
# Choice 1, small package
2*0.8 + 1.4*0.1 + 0.8*0.1
```

```
## [1] 1.82
```

```
# Choice 2, large package
5*0.8 + 3.5*0.1 + 2*0.1
```

```
## [1] 4.55
```

```
# Expected price for original settings
# Choice 1, small package
2*0.84 + 1.2*0.16
```

```
## [1] 1.872
```

```
# Choice 2, large package
5*0.84 + 3*0.16
```

```
## [1] 4.68
```

- After the price settings change, all utility values at each current inventory levels increase as compared to original settings.
- The expected prices for both choice 1 and choice 2 become lower than previous price settings: expected price for small package: $1.82 < 1.872$, expected price for large package: $4.55 < 4.68$. Thus, the lower the expected price, the less the price dis-utility, and the maximum expected sum of discounted utility is larger.
- When there is no promotion, the customer only opts for choice 1 (small package) when there is no stock; when there is a moderate promotion, the customer opts for choice 2 (large package) when the stock is low; when there is a great promotion, the customer opts for choice 2 (large package) as much as possible.