Problem Set 4

Felix Nguyen

December 16, 2019

Question 1

Read the data

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
state <- read.csv('PS2_state_space.csv')
F1 <- read.csv('PS2_transition_a1.csv')
F0 <- read.csv('PS2_transition_a0.csv')
sim <- read.csv('PS2_simdata.csv')
C <- as.numeric(state$C)
I <- as.numeric(state$I)
P <- as.numeric(state$P)

F1 <- as.matrix(F1[,3:70])
F0 <- as.matrix(F0[,3:70])
```

Setting up the global parameters

```
n <- nrow(state)
alpha <- 2
lambda <- -3
beta <- 0.99
ibar <- 4
ps <- 0.5
pr <- 2
c <- 0.25
gamma <- 0.5
euler <- -digamma(1)
```

Choice specific value function:

```
choice_val <- function(EV){
  U1 <- alpha*C - P
  U0 <- rep(0, n)
  for(i in 1:n){
    indi <- as.numeric(C[i] > 0)
    U0[i] <- ifelse(I[i] == 0, lambda*indi, alpha*C[i])
```

```r
  }
  V0 <- U0 + beta*F0%*%EV
  V1 <- U1 + beta*F1%*%EV
  V <- cbind(V1, V0)
  return(V)
}
```

The expected value function:

```r
emax <- function(EV0){
  val <- choice_val(EV0)
  EV1 <- log(rowSums(exp(val))) + euler
  return(EV1)
}
```

Finally, the contraction mapping:

```r
contraction <- function(threshold){
  k <- 0
  EV <- matrix(0,n,1)
  EV_new <- emax(EV)

  while(max(abs(EV_new-EV)) > threshold){
    EV <- EV_new
    EV_new <- emax(EV) #Vbar(s) = E(V(s,epsilon))
    k <- k +1
  }
  cat('Converged after ',k, 'iterations')
  return(EV_new)
}
```

Getting the expected value function and tabulate:

```r
EV_true <- contraction(1e-6)
```

```
## Converged after  1378 iterations
```

```r
tab1 <- cbind(state,EV_true)
print(tab1)
```

```
##       X id    I    C   P  EV_true
## 1   1-1  0 0.00 0.00 2.0 100.5981
## 2   2-1  1 0.25 0.00 2.0 101.1151
## 3   3-1  2 0.50 0.00 2.0 101.4264
## 4   4-1  3 0.75 0.00 2.0 101.6338
## 5   5-1  4 1.00 0.00 2.0 101.7796
## 6   6-1  5 1.25 0.00 2.0 101.8856
## 7   7-1  6 1.50 0.00 2.0 101.9641
## 8   8-1  7 1.75 0.00 2.0 102.0231
## 9   9-1  8 2.00 0.00 2.0 102.0677
## 10 10-1  9 2.25 0.00 2.0 102.1015
## 11 11-1 10 2.50 0.00 2.0 102.1269
## 12 12-1 11 2.75 0.00 2.0 102.1460
## 13 13-1 12 3.00 0.00 2.0 102.1602
## 14 14-1 13 3.25 0.00 2.0 102.1701
## 15 15-1 14 3.50 0.00 2.0 102.1769
## 16 16-1 15 3.75 0.00 2.0 102.1816
## 17 17-1 16 4.00 0.00 2.0 102.1848
## 18 18-1 17 0.00 0.25 2.0 100.0023
## 19 19-1 18 0.25 0.25 2.0 101.0981
## 20 20-1 19 0.50 0.25 2.0 101.6151
## 21 21-1 20 0.75 0.25 2.0 101.9264
## 22 22-1 21 1.00 0.25 2.0 102.1338
## 23 23-1 22 1.25 0.25 2.0 102.2796
## 24 24-1 23 1.50 0.25 2.0 102.3856
## 25 25-1 24 1.75 0.25 2.0 102.4641
## 26 26-1 25 2.00 0.25 2.0 102.5231
## 27 27-1 26 2.25 0.25 2.0 102.5677
## 28 28-1 27 2.50 0.25 2.0 102.6015
## 29 29-1 28 2.75 0.25 2.0 102.6269
## 30 30-1 29 3.00 0.25 2.0 102.6460
## 31 31-1 30 3.25 0.25 2.0 102.6602
## 32 32-1 31 3.50 0.25 2.0 102.6701
## 33 33-1 32 3.75 0.25 2.0 102.6769
## 34 34-1 33 4.00 0.25 2.0 102.6816
## 35 35-1 34 0.00 0.00 0.5 101.3188
## 36 36-1 35 0.25 0.00 0.5 101.6487
## 37 37-1 36 0.50 0.00 0.5 101.8726
## 38 38-1 37 0.75 0.00 0.5 102.0309
## 39 39-1 38 1.00 0.00 0.5 102.1460
## 40 40-1 39 1.25 0.00 0.5 102.2313
## 41 41-1 40 1.50 0.00 0.5 102.2954
## 42 42-1 41 1.75 0.00 0.5 102.3438
## 43 43-1 42 2.00 0.00 0.5 102.3806
## 44 44-1 43 2.25 0.00 0.5 102.4084
## 45 45-1 44 2.50 0.00 0.5 102.4293
## 46 46-1 45 2.75 0.00 0.5 102.4448
## 47 47-1 46 3.00 0.00 0.5 102.4563
## 48 48-1 47 3.25 0.00 0.5 102.4636
## 49 49-1 48 3.50 0.00 0.5 102.4686
## 50 50-1 49 3.75 0.00 0.5 102.4721
## 51 51-1 50 4.00 0.00 0.5 102.4745
## 52 52-1 51 0.00 0.25 0.5 101.3722
## 53 53-1 52 0.25 0.25 0.5 101.8188
## 54 54-1 53 0.50 0.25 0.5 102.1487
## 55 55-1 54 0.75 0.25 0.5 102.3726
## 56 56-1 55 1.00 0.25 0.5 102.5309
## 57 57-1 56 1.25 0.25 0.5 102.6460
## 58 58-1 57 1.50 0.25 0.5 102.7313
## 59 59-1 58 1.75 0.25 0.5 102.7954
## 60 60-1 59 2.00 0.25 0.5 102.8438
## 61 61-1 60 2.25 0.25 0.5 102.8806
## 62 62-1 61 2.50 0.25 0.5 102.9084
## 63 63-1 62 2.75 0.25 0.5 102.9293
## 64 64-1 63 3.00 0.25 0.5 102.9448
## 65 65-1 64 3.25 0.25 0.5 102.9563
## 66 66-1 65 3.50 0.25 0.5 102.9636
## 67 67-1 66 3.75 0.25 0.5 102.9686
## 68 68-1 67 4.00 0.25 0.5 102.9721
```

Problem 2

Calculating the CCP vector ($\hat{P}(s)$):

```
Phat <- sim %>% group_by(state_id) %>%
  summarise(Pr = sum(choice)/n()) %>% .$Pr
```

```
## Warning: The `printer` argument is deprecated as of rlang 0.3.0.
## This warning is displayed once per session.
```

```
Phat[Phat<0.001] <- 0.001 #Putting constraint 0.001 <= P <= .999

Phat[Phat>0.999] <- 0.999
```

Next, the CCP Mapping function:

```
CCP <- function(PR){
  U1 <- alpha*C - P
  U0 <- rep(0, n)
  for(i in 1:n){
    indi <- as.numeric(C[i] > 0)
    U0[i] <- ifelse(I[i] == 0, lambda*indi, alpha*C[i])
  }
  E1 <- euler - log(PR)
  E0 <- euler - log(1-PR)
  F_b <- F0*(1-PR) + F1*PR
  EU <- (1-PR)*(U0 + E0) + PR*(U1 + E1)
  EVP <- solve(as.matrix(diag(n)-beta*F_b))%*%EU
  val <- choice_val(EVP)
  Prb <- exp(val[,1])/rowSums(exp(val))
  return(list(EVP, Prb))
}
```

Calculate the value function and tabulate:

```
EV_P <- CCP(Phat)[[1]]
tab2 <- cbind(tab1, EV_P)
print(tab2)
```

```
##         X id    I    C   P  EV_true      EV_P
## S0    1-1  0 0.00 0.00 2.0 100.5981 100.07249
## S1    2-1  1 0.25 0.00 2.0 101.1151 100.54612
## S2    3-1  2 0.50 0.00 2.0 101.4264 100.87757
## S3    4-1  3 0.75 0.00 2.0 101.6338 101.18278
## S4    5-1  4 1.00 0.00 2.0 101.7796 101.42023
## S5    6-1  5 1.25 0.00 2.0 101.8856 101.58309
## S6    7-1  6 1.50 0.00 2.0 101.9641 101.69026
## S7    8-1  7 1.75 0.00 2.0 102.0231 101.78258
## S8    9-1  8 2.00 0.00 2.0 102.0677 101.84773
## S9   10-1  9 2.25 0.00 2.0 102.1015 101.89765
## S10 11-1 10 2.50 0.00 2.0 102.1269 101.92257
## S11 12-1 11 2.75 0.00 2.0 102.1460 101.95723
## S12 13-1 12 3.00 0.00 2.0 102.1602 101.97935
## S13 14-1 13 3.25 0.00 2.0 102.1701 101.99328
## S14 15-1 14 3.50 0.00 2.0 102.1769 102.00601
## S15 16-1 15 3.75 0.00 2.0 102.1816 102.01484
## S16 17-1 16 4.00 0.00 2.0 102.1848 102.01895
## S17 18-1 17 0.00 0.25 2.0 100.0023  99.43358
## S18 19-1 18 0.25 0.25 2.0 101.0981 100.55482
## S19 20-1 19 0.50 0.25 2.0 101.6151 100.94279
## S20 21-1 20 0.75 0.25 2.0 101.9264 101.40974
## S21 22-1 21 1.00 0.25 2.0 102.1338 101.69654
## S22 23-1 22 1.25 0.25 2.0 102.2796 101.92301
## S23 24-1 23 1.50 0.25 2.0 102.3856 102.08240
## S24 25-1 24 1.75 0.25 2.0 102.4641 102.19635
## S25 26-1 25 2.00 0.25 2.0 102.5231 102.28214
## S26 27-1 26 2.25 0.25 2.0 102.5677 102.34775
## S27 28-1 27 2.50 0.25 2.0 102.6015 102.39714
## S28 29-1 28 2.75 0.25 2.0 102.6269 102.42848
## S29 30-1 29 3.00 0.25 2.0 102.6460 102.45778
## S30 31-1 30 3.25 0.25 2.0 102.6602 102.47900
## S31 32-1 31 3.50 0.25 2.0 102.6701 102.49346
## S32 33-1 32 3.75 0.25 2.0 102.6769 102.50601
## S33 34-1 33 4.00 0.25 2.0 102.6816 102.51456
## S34 35-1 34 0.00 0.00 0.5 101.3188 100.64809
## S35 36-1 35 0.25 0.00 0.5 101.6487 100.13838
## S36 37-1 36 0.50 0.00 0.5 101.8726 101.45458
## S37 38-1 37 0.75 0.00 0.5 102.0309 101.50261
## S38 39-1 38 1.00 0.00 0.5 102.1460 101.81296
## S39 40-1 39 1.25 0.00 0.5 102.2313 101.94042
## S40 41-1 40 1.50 0.00 0.5 102.2954 102.04544
## S41 42-1 41 1.75 0.00 0.5 102.3438 102.11368
## S42 43-1 42 2.00 0.00 0.5 102.3806 102.16758
## S43 44-1 43 2.25 0.00 0.5 102.4084 102.21131
## S44 45-1 44 2.50 0.00 0.5 102.4293 102.22928
## S45 46-1 45 2.75 0.00 0.5 102.4448 102.26083
## S46 47-1 46 3.00 0.00 0.5 102.4563 102.26845
## S47 48-1 47 3.25 0.00 0.5 102.4636 102.28852
## S48 49-1 48 3.50 0.00 0.5 102.4686 102.29932
## S49 50-1 49 3.75 0.00 0.5 102.4721 102.30590
## S50 51-1 50 4.00 0.00 0.5 102.4745 102.30967
## S51 52-1 51 0.00 0.25 0.5 101.3722 100.88441
## S52 53-1 52 0.25 0.25 0.5 101.8188 101.26904
## S53 54-1 53 0.50 0.25 0.5 102.1487 101.63782
## S54 55-1 54 0.75 0.25 0.5 102.3726 101.91750
## S55 56-1 55 1.00 0.25 0.5 102.5309 102.16960
## S56 57-1 56 1.25 0.25 0.5 102.6460 102.33088
## S57 58-1 57 1.50 0.25 0.5 102.7313 102.45799
## S58 59-1 58 1.75 0.25 0.5 102.7954 102.54648
## S59 60-1 59 2.00 0.25 0.5 102.8438 102.61693
## S60 61-1 60 2.25 0.25 0.5 102.8806 102.67095
## S61 62-1 61 2.50 0.25 0.5 102.9084 102.70827
## S62 63-1 62 2.75 0.25 0.5 102.9293 102.72941
## S63 64-1 63 3.00 0.25 0.5 102.9448 102.76257
## S64 65-1 64 3.25 0.25 0.5 102.9563 102.77004
## S65 66-1 65 3.50 0.25 0.5 102.9636 102.78777
## S66 67-1 66 3.75 0.25 0.5 102.9686 102.79924
## S67 68-1 67 4.00 0.25 0.5 102.9721 102.80493
```

Problem 4

NXFP likelihood function

```r
Y <- sim$choice #Choice vector

loglike <- function(params){
  alpha <<- params[1]
  lambda <<- params[2]
  it <- 0
  eps <- 1e-10
  vCCP <- Phat
  vCCP0 <- rep(0,n)
  while(max(abs(vCCP0 - vCCP)) > eps){
    vCCP0 <- vCCP
    vCCP <- CCP(vCCP0)[[2]]
    it <- it + 1
  }
  vCCP1 <- rep(0, length(sim$state_id))
  for(j in 1:length(sim$state_id)){
    vCCP1[j] <- vCCP[sim$state_id[j]+1]
  }
  L <- vCCP1*Y + (1-vCCP1)*(1-Y)
  LLF <- sum(log(L))/1000
  return(LLF)
}
```

Maximizing the log-likelihood:

```r
bounds = c(-10, 10) # Reasonable bound for faster optimization
fit <- optim(par = rep(0.1,2), fn = loglike, method=c("L-BFGS-B"),
             lower=bounds[1],upper=bounds[2],control=list(fnscale=-1)) #-1 to Maximize
fit
```

```
## $par
## [1]  0.7310996 -2.4243890
##
## $value
## [1] -8.661219
##
## $counts
## function gradient
##       14       14
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```r
cat('alpha is', fit$par[1])
```

```
## alpha is 0.7310996
```

```r
cat(' lambda is', fit$par[2])
```

```
##  lambda is -2.424389
```