Data Analysis Homework 3

Jimmy Hickey

10/21/2020

1

a

Notice that

$$\frac{\partial \mu}{\partial \alpha} = \frac{\partial}{\partial \alpha}\alpha^T \begin{bmatrix} 1 \\ \eta \\ \eta^2 \end{bmatrix} = \begin{bmatrix} 1 \\ \eta \\ \eta^2 \end{bmatrix}.$$

Further, the first term of our estimating equation is constant in α so take

$$A_{\eta_j,i} = \frac{\mathcal{C}_{d_{\eta_j},i}}{\prod_{k=2}^{K}[\pi_{\eta_j,k}(\overline{X}_{ki}, \widehat{\gamma}_k)]\pi_{\eta_j,1}(X_1; \hat{\gamma}_1)}$$

Then multiplying our derivative vector gives us the estimating equations

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{m} 1 \cdot \left[A_{\eta_j,i}(Y_1 - \alpha_1 - \alpha_2\eta_j - \alpha_3\eta_j^2)\right] \sum_{i=1}^{n}\sum_{j=1}^{m} 1 \cdot \left[A_{\eta_j,i}(Y_1 - \alpha_1 - \alpha_2\eta_j - \alpha_3\eta_j^2)\right] &= 0 \\ &= 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} \eta \cdot \left[A_{\eta_j,i}(Y_1 - \alpha_1 - \alpha_2\eta_j - \alpha_3\eta_j^2)\right] &= 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} \eta^2 \cdot \left[A_{\eta_j,i}(Y_1 - \alpha_1 - \alpha_2\eta_j - \alpha_3\eta_j^2)\right] &= 0 \end{aligned}$$

We will take our range to be $\eta \in (50, 400)$. With a step size of 10, this gives $m = 36$.

```r
ldl = read.table("LDL.dat.txt", header=FALSE)
# remove ID column
ldl = ldl[,-1]
names(ldl) = c("L1", "A1", "L2", "S2", "A2", "L3",
"S3", "A3", "L4", "S4", "A4", "Y", "S5")



logistic_func = function(x){
  return( exp(x) / (1 + exp(x)) )
}

# Propensity model from equation 7 of homework
calc_gamma = function(data){
  out = matrix(0, nrow=4, ncol=3)

  gamma1_mod = glm(A1 ~ L1, data, family = "binomial")
  # add extra 0 because other terms has an S factor
  out[1,] = c(gamma1_mod$coefficients, 0)

  gamma2_mod = glm(A2 ~ L2 + S2, data, family = "binomial")
  out[2,] = gamma2_mod$coefficients

  gamma3_mod = glm(A3 ~ L3 + S3, data, family = "binomial")
  out[3,] = gamma3_mod$coefficients

  gamma4_mod = glm(A4 ~ L4 + S4, data, family = "binomial")
  out[4,] = gamma4_mod$coefficients

  return(out)
}


# Cd vector for equation 5.27 on slide 304
calc_cd = function(data, regime, K){
  n = dim(data)[1]


  # need for AIPW
  if(K==0){
    return(rep(1, n))
  }


  L = cbind(data$L1, data$L2, data$L3, data$L4, data$Y)
  # again need 0s because there are no side effects at the beginning
  S = cbind(rep(0, n), data$S2, data$S3, data$S4, data$S5)
  A = cbind(data$A1, data$A2, data$A3, data$A4)


  cd_vec = rep(1, n)

  for(i in 1:n){

    for(k in 1:K){
        decision = regime(L[i,k], S[i,k], A[i,k], k)
        cd_vec[i] = cd_vec[i] * ( A[i,k] == decision )
    }
  }
  return(cd_vec)
}
```

```
# calculate the product of propensities used in the denominator
propen_denom = function(data, regime, K){
    n = dim(data)[1]

  # need for AIPW
  if(K==0){
    return(rep(1, n))
  }

  L = cbind(data$L1, data$L2, data$L3, data$L4, data$Y)
  # again need 0s because there are no side effects at the beginning
  S = cbind(rep(0, n), data$S2, data$S3, data$S4, data$S5)
  A = cbind(data$A1, data$A2, data$A3, data$A4)

  gamma = calc_gamma(data)


  # initialize vector of length n
  prod = rep(1, n)

  for(i in 1:n){

    for(k in 1:K){
      val = gamma[k, 1] + gamma[k, 2] * L[i,k] + gamma[k, 3] * S[i,k]
      p = logistic_func(val)
      dk = regime(L[i,], S[i,], A[i,], k)

      pi_k = p * dk + (1-p)*(1-dk)

      prod[i] = prod[i] * pi_k
    }
  }

  return(prod)
}


MSM = function(data, K, etas){
  m = length(etas)
  n = dim(data)[1]

  # First we will build the A matrix
  A_mat = matrix(NA, nrow = n, ncol = m)

  for(j in 1:m){
    eta_j = etas[j]

    # define new regime for each eta value
    regime_eta = function(L, S, A, dk){
      return(S == 0 && L > eta_j)
    }

    cd_j = calc_cd(data, regime_eta, K)
    propen = propen_denom(data, regime_eta, K)

    A_mat[1:n, j] = cd_j / propen
  }

  # vector of left parameters that is constant in alpha
  const_vec = c(
    data$Y %*% A_mat %*% rep(1, m),
    data$Y %*% A_mat %*% etas,
    data$Y %*% A_mat %*% (etas^2)
  )


  # matrix of alpha coefficients for 3 equations we need to solve
  alpha_mat = matrix(c(
    rep(1, n) %*% A_mat %*% rep(1, m),
      rep(1, n) %*% A_mat %*% etas,
      rep(1, n) %*% A_mat %*% (etas^2),
    rep(1, n) %*% A_mat %*% etas,
      rep(1, n) %*% A_mat %*% (etas^2),
      rep(1, n) %*% A_mat %*% (etas^3),
    rep(1, n) %*% A_mat %*% (etas^2),
      rep(1, n) %*% A_mat %*% (etas^3),
      rep(1, n) %*% A_mat %*% (etas^4)
  ), nrow=3, ncol=3, byrow = TRUE)


  return(solve(alpha_mat) %*% const_vec)
}


etas = seq(50, 200, length.out=50)
K = 4
MSM(ldl, K, etas)
```

```
##                [,1]
## [1,] 117.85292039
## [2,]  -0.39425908
## [3,]   0.00276527
```

b

```
MSM_value = function(data, K, eta_vec, eta){
  fit = MSM_fit(data, K, eta_vec)
  return(fit[1] + fit[2] * eta + fit[3] * eta^2)
}


MSM_bootstrap = function(data, K, eta_vec, eta, rep){
  data_val = MSM_value(data, K, eta_vec, eta)
  boot_val = rep(NA, rep)

  for(i in 1:rep){
    boot_data = data[sample( dim(data)[1], replace = TRUE ), ]
    boot_val[i] = MSM_value(boot_data, K, eta_vec, eta)
  }

  se = sd(boot_val)
  return(c(data_val, se))
}

# See results below
# eta_vec = seq(90, 200, 10)
# for(i in 1:length(eta_vec)){
#   boot_est = MSM_bootstrap(ldl, K=4, eta_vec, eta_vec[i], 5)
#
#   cat("=======\nFor eta=", eta_vec[i], " the value is ", boot_est[1],
#       " with a standard deviation of ", boot_est[2])
# }
```

C

The bootstrap takes a long time to run, so here is the code from a previous run and the output.

```
# eta_vec = seq(90, 200, 10)
# for(i in 1:length(eta_vec)){
#   boot_est = MSM_bootstrap(ldl, K=4, eta_vec, eta_vec[i], 5)
#
#   cat("=======\nFor eta=", eta_vec[i], " the value is ", boot_est[1],
#       " with a standard deviation of ", boot_est[2])
# }

# =======
# For eta= 90  the value is  106.1375  with a standard deviation of  1.046301=======
# For eta= 100  the value is  107.5367  with a standard deviation of  1.671561=======
# For eta= 110  the value is  105.5903  with a standard deviation of  1.610017=======
# For eta= 120  the value is  100.2983  with a standard deviation of  11.07936=======
# For eta= 130  the value is  91.66065  with a standard deviation of  27.16922=======
# For eta= 140  the value is  79.67742  with a standard deviation of  71.67292=======
# For eta= 150  the value is  64.34858  with a standard deviation of  88.07134=======
# For eta= 160  the value is  45.67413  with a standard deviation of  49.85914=======
# For eta= 170  the value is  23.65407  with a standard deviation of  101.2761=======
# For eta= 180  the value is  -1.711588  with a standard deviation of  181.0727=======
# For eta= 190  the value is  -30.42286  with a standard deviation of  80.02617=======
# For eta= 200  the value is  -62.47973  with a standard deviation of  291.0023
```

Clearly by those standard deviations (and negative values!) that something needs to be address and perhaps 5 reptitions of the bootstrap is not enough. These results are very different than the results from homework 2, where we saw the optimal η around 150 (and standard errors less than 15!).

2

a

```r
# adapting code from Dr. Halloway's slide 21
# start at decision 4 with data S4 (side effect at 4)
fSet4 = function(S4){

  # can be (0,1) or (0)
  # label them option S42 and S41
  # thse are A_k,2 and A_k,1 in problem statement
  subsets = list( list("S42", c(0,1)),
                  list("S41", c(0)))

  txOpts = rep(x = NA, times = length(x = S4))

  txOpts[ S4 == 0] = "S42"
  txOpts[ S4 == 1] = "S41"

  # need named list
  return( list("subsets" = subsets, "txOpts" = txOpts))

}


# set up models and contrasts
# similar to Halloway slide 32

# models for decision S41
# include data up to decision 4
moMain_S41 = buildModelObjSubset(model = ~ L1 + L2 + L3 + L4 + S2 + S3,
                                 solver.method = "lm",
                                 subset= "S41",
                                 dp = 2L)

# only want L4 here
moCont_S41 = buildModelObjSubset(model = ~ L4,
                                 solver.method = "lm",
                                 subset= "S41",
                                 dp = 2L)

# models for decision S42
# include data up to decision 4
moMain_S42 = buildModelObjSubset(model = ~ L1 + L2 + L3 + L4 + S2 + S3,
                                 solver.method = "lm",
                                 subset= "S42",
                                 dp = 2L)

# only want L4 here
moCont_S42 = buildModelObjSubset(model = ~ L4,
                                 solver.method = "lm",
                                 subset= "S42",
                                 dp = 2L)
```

```r
moMain4_list = list(moMain_S41, moMain_S42)
moCont4_list = list(moCont_S41, moCont_S42)

# qlearn
# note we take response -1 to minimize instead of maximize
q4 = qLearn(moMain = moMain4_list,
            moCont = moCont4_list,
            iter = 0L,
            data = ldl,
            response = -1 * ldl$Y,
            txName = "A4",
            fSet = fSet4
            )
```

```
## First step of the Q-Learning Algorithm.
##
## Subsets of treatment identified as:
## $S41
## [1] 0
##
## $S42
## [1] 0 1
##
## Number of patients in data for each subset:
##  S41  S42
##  428 4572
##
## Outcome regression.

## NOTE: subset(s) S41 received tx not in accordance with specified feasible tx sets

## Fitting models for  S41 using 428 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + L3 + L4 + S2 + S3 + A4 +
##     L4:A4, data = data)
##
## Coefficients:
## (Intercept)           L1           L2           L3           L4           S2
##   15.291669     0.037129    -0.147751     0.093294    -1.019478    -1.519143
##          S3           A4        L4:A4
##    0.063996    -5.438549     0.009569
##
## Fitting models for  S42 using 4572 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + L3 + L4 + S2 + S3 + A4 +
##     L4:A4, data = data)
##
## Coefficients:
```

```
## (Intercept)          L1          L2          L3          L4          S2
##    4.371803   -0.007653    0.031302   -0.032936   -0.978421    0.814785
##          S3          A4       L4:A4
##    1.442055   14.445602   -0.025825
##
##
## Recommended Treatments:
##    0    1
##  428 4572
##
## Estimated value: -119.1496
```

```
# getting at coefficients is ugly
cat("Stage 4 decision:\nGive the patient the standard dose if they are experiencing side effects.\nIf t
```

```
## Stage 4 decision:
## Give the patient the standard dose if they are experiencing side effects.
## If the patient is not experiencing side effects, provide a high dose only if LDL < 559.369 = -14.445
```

```
# Decision point 3

fSet3 = function(S3){

  # can be (0,1) or (0)
  # label them option S32 and S31
  # thse are A_k,2 and A_k,1 in problem statement
  subsets = list( list("S32", c(0,1)),
                  list("S31", c(0)))

  txOpts = rep(x = NA, times = length(x = S3))

  txOpts[ S3 == 0] = "S32"
  txOpts[ S3 == 1] = "S31"

  # need named list
  return( list("subsets" = subsets, "txOpts" = txOpts))

}


# set up models and contrasts
# similar to Halloway slide 32

# models for decision S31
# include data up to decision 3
moMain_S31 = buildModelObjSubset(model = ~ L1 + L2 + L3 + S2,
                                 solver.method = "lm",
                                 subset= "S31",
                                 dp = 2L)

# only want L3 here
moCont_S31 = buildModelObjSubset(model = ~ L3,
                                 solver.method = "lm",
```

```r
                                  subset= "S31",
                                  dp = 2L)

# models for decision S32
# include data up to decision 3
moMain_S32 = buildModelObjSubset(model = ~ L1 + L2 + L3 + S2,
                                  solver.method = "lm",
                                  subset= "S32",
                                  dp = 2L)

# only want L3 here
moCont_S32 = buildModelObjSubset(model = ~ L3,
                                  solver.method = "lm",
                                  subset= "S32",
                                  dp = 2L)

moMain3_list = list(moMain_S31, moMain_S32)
moCont3_list = list(moCont_S31, moCont_S32)

# qlearn
# response is the output from qlearning at decision 4!
q3 = qLearn(moMain = moMain3_list,
            moCont = moCont3_list,
            iter = 0L,
            data = ldl,
            response = q4,
            txName = "A3",
            fSet = fSet3
            )
```

```
## Step 2 of the Q-Learning Algorithm.
##
## Subsets of treatment identified as:
## $S31
## [1] 0
##
## $S32
## [1] 0 1
##
## Number of patients in data for each subset:
##  S31  S32
##  385 4615
##
## Outcome regression.

## NOTE: subset(s) S31 received tx not in accordance with specified feasible tx sets

## Fitting models for  S31 using 385 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + L3 + S2 + A3 + L3:A3, data = data)
##
```

```
## Coefficients:
## (Intercept)           L1           L2           L3           S2           A3
##    31.92297     -0.03372      0.05725     -1.05302      1.40321    -14.62403
##       L3:A3
##     0.04108
##
## Fitting models for  S32 using 4615 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + L3 + S2 + A3 + L3:A3, data = data)
##
## Coefficients:
## (Intercept)           L1           L2           L3           S2           A3
##   22.879561    -0.005727     0.021170    -1.013018    -0.951402    15.489884
##       L3:A3
##   -0.037653
##
##
## Recommended Treatments:
##    0    1
##  385 4615
##
## Estimated value: -114.0321
```

```r
# getting at coefficients is ugly
cat("Stage 3 decision:\nGive the patient the standard dose if they are experiencing side effects.\nIf tl
```

```
## Stage 3 decision:
## Give the patient the standard dose if they are experiencing side effects.
## If the patient is not experiencing side effects, provide a high dose only if LDL < 411.3873 = -15.489
```

```r
# Decision point 2

fSet2 = function(S2){

  # can be (0,1) or (0)
  # label them option S22 and S21
  # these are A_k,2 and A_k,1 in problem statement
  subsets = list( list("S22", c(0,1)),
                  list("S21", c(0)))

  txOpts = rep(x = NA, times = length(x = S2))

  txOpts[ S2 == 0] = "S22"
  txOpts[ S2 == 1] = "S21"

  # need named list
  return( list("subsets" = subsets, "txOpts" = txOpts))

}
```

```
# set up models and contrasts
# similar to Halloway slide 22

# models for decision S21
# include data up to decision 2
moMain_S21 = buildModelObjSubset(model = ~ L1 + L2,
                                 solver.method = "lm",
                                 subset= "S21",
                                 dp = 2L)

# only want L2 here
moCont_S21 = buildModelObjSubset(model = ~ L2,
                                 solver.method = "lm",
                                 subset= "S21",
                                 dp = 2L)

# models for decision S22
# include data up to decision 2
moMain_S22 = buildModelObjSubset(model = ~ L1 + L2,
                                 solver.method = "lm",
                                 subset= "S22",
                                 dp = 2L)

# only want L2 here
moCont_S22 = buildModelObjSubset(model = ~ L2,
                                 solver.method = "lm",
                                 subset= "S22",
                                 dp = 2L)

moMain2_list = list(moMain_S21, moMain_S22)
moCont2_list = list(moCont_S21, moCont_S22)

# qlearn
# response is the output from qlearning at decision 4!
q2 = qLearn(moMain = moMain2_list,
            moCont = moCont2_list,
            iter = 0L,
            data = ldl,
            response = q3,
            txName = "A2",
            fSet = fSet2
)
```

```
## Step 3 of the Q-Learning Algorithm.
##
## Subsets of treatment identified as:
## $S21
## [1] 0
##
## $S22
## [1] 0 1
##
## Number of patients in data for each subset:
```

```
##  S21  S22
##  297 4703
##
## Outcome regression.

## NOTE: subset(s) S21 received tx not in accordance with specified feasible tx sets

## Fitting models for  S21 using 297 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + A2 + L2:A2, data = data)
##
## Coefficients:
## (Intercept)           L1           L2           A2        L2:A2
##    62.35559     -0.06540     -1.07636     -7.67885      0.02724
##
## Fitting models for  S22 using 4703 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + L2 + A2 + L2:A2, data = data)
##
## Coefficients:
## (Intercept)           L1           L2           A2        L2:A2
##    40.95186      0.02639     -1.03888     12.37753     -0.01005
##
##
## Recommended Treatments:
##    0    1
##  297 4703
##
## Estimated value: -108.9012
```

```
# getting at coefficients is ugly
cat("Stage 2 decision:\nGive the patient the standard dose if they are experiencing side effects.\nIf tl
```

```
## Stage 2 decision:
## Give the patient the standard dose if they are experiencing side effects.
## If the patient is not experiencing side effects, provide a high dose only if LDL < 1231.31 = -12.3775
```

```
# Decision point 1

fSet1 = function(data){

  # no side effects at 1
  # so only make decision based on LDL
  subsets = list( list("S12", c(0,1)))

  txOpts = rep(x = "S12", times = dim(data)[1])

  # need named list
  return( list("subsets" = subsets, "txOpts" = txOpts))
```

```r
}


# models for decision S12
# include data up to decision 1
moMain_S12 = buildModelObjSubset(model = ~ L1,
                                 solver.method = "lm",
                                 subset= "S12",
                                 dp = 2L)

# only want L1 here
moCont_S12 = buildModelObjSubset(model = ~ L1,
                                 solver.method = "lm",
                                 subset= "S12",
                                 dp = 2L)

moMain1_list = list(moMain_S12)
moCont1_list = list(moCont_S12)

# qlearn
# response is the output from qlearning at decision 4!
q1 = qLearn(moMain = moMain1_list,
            moCont = moCont1_list,
            iter = 0L,
            data = ldl,
            response = q2,
            txName = "A1",
            fSet = fSet1
)
```

```
## Step 4 of the Q-Learning Algorithm.
##
## Subsets of treatment identified as:
## $S12
## [1] 0 1
##
## Number of patients in data for each subset:
##  S12
## 5000
##
## Outcome regression.
## Fitting models for  S12 using 5000 patient records.
## Regression analysis for Combined:
##
## Call:
## lm(formula = YinternalY ~ L1 + A1 + L1:A1, data = data)
##
## Coefficients:
## (Intercept)           L1           A1        L1:A1
##    61.27611     -1.03266     18.17817     -0.04398
##
##
## Recommended Treatments:
```

```
##    1
## 5000
##
## Estimated value: -103.6736
```

```
# getting at coefficients is ugly
cat("Stage 1 decision:\nGive the patient the standard dose if they are experiencing side effects.\nIf th
```

```
## Stage 1 decision:
## Give the patient the standard dose if they are experiencing side effects.
## If the patient is not experiencing side effects, provide a high dose only if LDL < 413.3304 = -18.178
```

All of these LDL cut offs seem very high, so perhaps something is wrong.

b

```
cat("The value of the regime is decided at the last step of our backward iterative process. That is, it
```

```
## The value of the regime is decided at the last step of our backward iterative process. That is, it is
```

```
q1
```

```
## Q-Learning: step 4
## Outcome Regression Analysis
## $Subset=S12
## Combined
##
## Call:
## lm(formula = YinternalY ~ L1 + A1 + L1:A1, data = data)
##
## Coefficients:
## (Intercept)           L1           A1        L1:A1
##    61.27611     -1.03266     18.17817     -0.04398
##
## Recommended Treatments:
##    1
## 5000
##
## Estimated value: -103.6736
```

3

```
# same as last homework but now using propensity product function
calc_ipw = function(data, regime, K){


  cd = calc_cd(data, regime, K)
  gamma = calc_gamma(data)

  propen_prod = propen_denom(data, regime, K)

  numerator = data$Y * cd

  est = numerator / propen_prod

  return(mean(est))
}


etas = seq(min(ldl$Y), max(ldl$Y), length.out = 100)
ipw_list = rep(NA, length(etas))
eta_opt_ind = 1

for(i in 1:length(etas)){
    eta_i = etas[i]

    # define new regime for each eta value
    regime_eta = function(L, S, A, dk){
      return(S == 0 && L > eta_i)
    }

    ipw_list[i] = calc_ipw(ldl, regime_eta, K)

    # less than because we are minimizing
    if(ipw_list[i] < ipw_list[eta_opt_ind]){
      eta_opt_ind = i
    }

}

print(ipw_list)
```

```
##   [1] 103.2618 103.5675 103.8635 104.1626 104.1626 104.4514 104.7663 105.0819
##   [9] 105.0819 105.0819 105.4096 105.2108 105.1566 105.4074 106.5677 106.8606
##  [17] 106.2633 106.6128 105.9832 106.4691 106.9998 107.5996 109.9855 108.9414
##  [25] 110.0942 110.9918 111.6006 113.7153 117.1844 117.7439 119.9620 121.3681
##  [33] 123.3645 129.8544 131.7266 134.3165 136.1269 135.7932 138.2186 144.7866
##  [41] 149.3071 155.4781 154.8681 151.1307 147.3079 147.5461 150.9764 157.9237
##  [49] 158.0361 161.6400 165.3670 164.6654 166.8271 163.7152 166.9793 176.1237
##  [57] 170.3609 165.6695 162.3722 162.7982 163.4683 162.8151 160.5350 160.5523
##  [65] 164.5591 164.5198 169.2285 163.4470 154.8867 153.5614 148.7635 143.6050
##  [73] 139.6045 132.9776 127.3258 125.9615 130.5406 131.5616 134.4396 131.6680
##  [81] 129.9467 130.4705 134.6330 136.9419 138.5811 138.4425 139.3084 138.2369
##  [89] 137.8617 137.3485 135.2239 135.8608 136.5110 136.3096 137.0625 137.0625
##  [97] 137.0625 137.0625 137.0625 136.3096
```

```
cat("The optimal choice in eta is ", etas[eta_opt_ind])
```

```
## The optimal choice in eta is  51
```

b

```r
cat("This regime has a value of ", ipw_list[eta_opt_ind])
```

```
## This regime has a value of  103.2618
```

c

Here the minimum value of 103.2618 was achieved at $\eta = 51$. While this is consistent when looking through the list of IPW values, it does raise suspicions as it is very low. In fact, it is the first value tried (the minimum value in our range).

4

```r
# Notice that for the AIPW estimator we will need our Q functions
# that we previously estimated in 2

Q_val = function(data, regime, K, Q_coeff){
  n = dim(data)[1]

  L = cbind(data$L1, data$L2, data$L3, data$L4, data$Y)
  # again need 0s because there are no side effects at the beginning
  S = cbind(rep(0, n), data$S2, data$S3, data$S4, data$S5)
  A = cbind(data$A1, data$A2, data$A3, data$A4)


  decisions = rep(NA, n)

  for(i in 1:n){
    decisions[i] = regime(L, S, A, K)
  }


  # need to decide which q function to use
  if(K == 1){
    Q_val = cbind(rep(1, n), data$L1, decisions, decisions * data$L1) %*%
      unlist(Q_coeff[1])
  } else if(K == 2){
    Q_val = cbind(rep(1,n), data$L1, data$L2, decisions, decisions * data$L2) %*%
      unlist(Q_coeff[2])
  } else if(K == 3){
     Q_val = cbind(rep(1,n), data$L1, data$L2, data$L3, data$S2, decisions, decisions * data$L3) %*%
       unlist(Q_coeff[3])
  } else if(K == 4){
      Q_val = cbind(rep(1,n), data$L1, data$L2, data$L3, data$L4, data$S2, data$S3, decisions, decisio
        unlist(Q_coeff[4])
  }

  # negative for miminzation again
  return( -1 * Q_val)
}



# modify function from before
# eqn 5.37
calc_aipw = function(data, regime, K, Q_coeff){
  n = dim(data)[1]


  cd = calc_cd(data, regime, K)
  gamma = calc_gamma(data)

  propen_prod = propen_denom(data, regime, K)

  numerator = data$Y * cd

  ipw = numerator / propen_prod

  augmentation = rep(0, n)

  for(k in 1:K){
    # cd_k-1
    cd_km1 = calc_cd(data, regime, k-1)
    # cd_k
    cd_k = calc_cd(data, regime, k)


    # propensity_k-1
    prop_km1 = propen_denom(data, regime, k-1)
    # propensity_k
    prop_k = propen_denom(data, regime, k)
    Q = Q_val(data, regime, k, Q_coeff)

    augmentation = augmentation + ( cd_km1 / prop_km1 - cd_k / prop_k ) * Q
  }

  return(mean(ipw + augmentation))
}



Q_coeff= list(coef(q1)$outcome$`Subset=S12`$Combined,
           coef(q2)$outcome$`Subset=S22`$Combined,
           coef(q3)$outcome$`Subset=S32`$Combined,
           coef(q4)$outcome$`Subset=S42`$Combined)

etas = seq(min(ldl$Y), max(ldl$Y), length.out = 100)
aipw_list = rep(NA, length(etas))
eta_opt_ind = 1

for(i in 1:length(etas)){
    eta_i = etas[i]

    # define new regime for each eta value
    regime_eta = function(L, S, A, dk){
      return(S == 0 && L > eta_i)
    }

    aipw_list[i] = calc_aipw(ldl, regime_eta, K, Q_coeff)

    if(aipw_list[i] < aipw_list[eta_opt_ind]){
      eta_opt_ind = i
    }

}
print(aipw_list)
```

```
##   [1] 104.1116 104.2654 104.4058 104.5594 104.5594 104.6730 104.7649 104.8520
##   [9] 104.8520 104.8520 104.9399 104.9767 105.1606 105.1721 105.5040 105.5852
##  [17] 105.4889 105.5710 105.4904 105.5935 105.8215 105.9179 106.3997 106.4141
##  [25] 106.9559 107.1152 107.5190 108.3679 108.9176 109.0226 109.7263 109.9118
##  [33] 110.4739 111.0104 111.3848 112.3159 113.2555 114.2143 114.4714 116.0535
##  [41] 116.7182 117.3503 117.8656 118.1700 118.2054 119.4857 120.6117 121.9622
##  [49] 122.8833 123.8343 124.5236 125.5784 126.2821 126.3311 127.2094 129.2218
##  [57] 129.1552 128.7320 128.9108 129.5767 130.5850 131.3172 131.8060 132.9511
##  [65] 134.0242 134.6255 136.4522 136.1812 136.3042 136.7686 137.3811 137.5494
##  [73] 138.2982 139.2292 139.2462 139.4365 140.8141 141.8123 142.3038 142.3816
##  [81] 142.4374 142.9560 143.3912 143.5237 143.4161 143.2406 143.5896 143.6967
##  [89] 143.7245 143.9027 143.9756 143.9342 143.8902 143.7182 143.7401 143.7401
##  [97] 143.7401 143.7266 143.7266 143.7266
```

```
cat("The optimal choice in eta is ", etas[eta_opt_ind])
```

```
## The optimal choice in eta is  51
```

b

```
cat("This regime has a value of ", aipw_list[eta_opt_ind])
```

```
## This regime has a value of  104.1116
```

c

Here the minimum value of 104.1116 was achieved at $\eta = 51$. This is consistent with the values in the output vector, but is this again occurs at the minimum η tested. This behavior between both the IPW and AIPW estimators likely indicates a bug in the IPW part of the calcualtion.

5

a

```r
# specify propensity model for each feasible set
# Like Halloway side 56 but longer

# decision 4
moPropen_S42 = buildModelObjSubset(model = ~ L4,
                                  solver.method = "glm",
                                  solver.args = list("family" = "binomial"),
                                  predict.args = list("type" = "response"),
                                  subset = "S42",
                                  dp = 4L )

moPropen_S41 = buildModelObjSubset(model = ~ L4,
                                  solver.method = "glm",
                                  solver.args = list("family" = "binomial"),
                                  predict.args = list("type" = "response"),
                                  subset = "S41",
                                  dp = 4L )

# decision 3
moPropen_S32 = buildModelObjSubset(model = ~ L3,
                                   solver.method = "glm",
                                   solver.args = list("family" = "binomial"),
                                   predict.args = list("type" = "response"),
                                   subset = "S32",
                                   dp = 3L )

moPropen_S31 = buildModelObjSubset(model = ~ L3,
                                   solver.method = "glm",
                                   solver.args = list("family" = "binomial"),
                                   predict.args = list("type" = "response"),
                                   subset = "S31",
                                   dp = 3L)

# decision 2
moPropen_S22 = buildModelObjSubset(model = ~ L2,
                                   solver.method = "glm",
                                   solver.args = list("family" = "binomial"),
                                   predict.args = list("type" = "response"),
                                   subset = "S22",
                                   dp =2L )

moPropen_S21 = buildModelObjSubset(model = ~ L2,
                                   solver.method = "glm",
                                   solver.args = list("family" = "binomial"),
                                   predict.args = list("type" = "response"),
                                   subset = "S21",
                                   dp = 2L )
```

```
# decision 1
moPropen_S12 = buildModelObjSubset(model = ~ L1,
                                   solver.method = "glm",
                                   solver.args = list("family" = "binomial"),
                                   predict.args = list("type" = "response"),
                                   subset = "S12",
                                   dp = 1L )

moPropen_list = list(moPropen_S12,
                   moPropen_S22, moPropen_S21,
                   moPropen_S32, moPropen_S31,
                   moPropen_S42, moPropen_S41)

# redefine regimes to make DynTx happy
# see Halloway slide 59
# note that our classes are 0,1 rather than "A" and "B"

# regime at decision 1
regime_d1 = function(eta1, data){
  # cast to integer because idk it it can handle booleans
  return(as.integer(data$L1>eta1))
}

regime_d2 = function(eta2, data){
  # check S2 first for hopeful lazy eval
  return(as.integer(data$S2==0 & data$L2>eta2))
}

regime_d3 = function(eta3, data){
  return(as.integer(data$S3==0 & data$L3>eta3))
}

regime_d4 = function(eta4, data){
  return(as.integer(data$S4==0 & data$L4>eta4))
}

# Halloway slide 62
# don't need to worry about na.rm for our case since we are always
# recording the same covariates


starting.values = c(c(mean(ldl$L1)), c(mean(ldl$L2)), c(mean(ldl$L3)), c(mean(ldl$L4)))

Domains = matrix(data =
  c(c(min(ldl$L1)), c(min(ldl$L2)), c(min(ldl$L3)), c(min(ldl$L4)),
  c(max(ldl$L1)), c(max(ldl$L2)), c(max(ldl$L3)), c(max(ldl$L4))),
  ncol = 2L)


pop.size = 500 # Too Small


vsObj = optimalSeq(moPropen = list(moPropen_S12,moPropen_S22,moPropen_S21,moPropen_S32,moPropen_S31,moP
                    data = ldl, response = -1*ldl$Y, txName = c('A1','A2','A3','A4'),
                    regimes = list(regime_d1, regime_d2,regime_d3,regime_d4),
                    fSet = list(fSet1, fSet2,fSet3,fSet4),
                    Domains = as.numeric(Domains),
                    starting.values = starting.values,
                    pop.size = pop.size,
                    verbose = TRUE)
```

```
## IPW estimator will be used
## Value Search - Coarsened Data Perspective 4 Decision Points
## Decision point 1
##
## Subsets of treatment identified as:
## $S12
## [1] 0 1
##
## Number of patients in data for each subset:
##  S12
## 5000
## Decision point 2
##
## Subsets of treatment identified as:
## $S21
## [1] 0
##
## $S22
## [1] 0 1
##
## Number of patients in data for each subset:
##  S21  S22
##  297 4703
## Decision point 3
##
## Subsets of treatment identified as:
## $S31
## [1] 0
##
## $S32
## [1] 0 1
##
## Number of patients in data for each subset:
##  S31  S32
##  385 4615
## Decision point 4
##
## Subsets of treatment identified as:
## $S41
## [1] 0
##
## $S42
## [1] 0 1
##
## Number of patients in data for each subset:
```

```
##  S41  S42
##  428 4572
##
## Propensity for treatment regression.
## Decision point 1
## Fitting models for  S12 using 5000 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L1, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L1
##   -0.991801     0.006118
##
## Degrees of Freedom: 4999 Total (i.e. Null);  4998 Residual
## Null Deviance:       6928
## Residual Deviance: 6922  AIC: 6926
## Decision point 2

## NOTE: subset(s) S21 received tx not in accordance with specified feasible tx sets

## Fitting models for  S22 using 4703 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L2, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L2
##   -0.595683     0.003867
##
## Degrees of Freedom: 4702 Total (i.e. Null);  4701 Residual
## Null Deviance:       6519
## Residual Deviance: 6514  AIC: 6518
## Fitting models for  S21 using 297 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L2, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L2
##    1.438085    -0.009561
##
## Degrees of Freedom: 296 Total (i.e. Null);  295 Residual
## Null Deviance:       411.3
## Residual Deviance: 409.2     AIC: 413.2
## Decision point 3

## NOTE: subset(s) S31 received tx not in accordance with specified feasible tx sets

## Fitting models for  S32 using 4615 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L3, family = "binomial", data = data)
```

```
##
## Coefficients:
## (Intercept)           L3
##   -0.527456     0.002925
##
## Degrees of Freedom: 4614 Total (i.e. Null);  4613 Residual
## Null Deviance:       6387
## Residual Deviance: 6382  AIC: 6386
## Fitting models for  S31 using 385 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L3, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L3
##     0.21811     -0.00347
##
## Degrees of Freedom: 384 Total (i.e. Null);  383 Residual
## Null Deviance:       525.8
## Residual Deviance: 525.3     AIC: 529.3
## Decision point 4

## NOTE: subset(s) S41 received tx not in accordance with specified feasible tx sets

## Fitting models for  S42 using 4572 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L4, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L4
##    -0.62507      0.00275
##
## Degrees of Freedom: 4571 Total (i.e. Null);  4570 Residual
## Null Deviance:       6266
## Residual Deviance: 6261  AIC: 6265
## Fitting models for  S41 using 428 patient records.
## Regression analysis for moPropen:
##
## Call:  glm(formula = YinternalY ~ L4, family = "binomial", data = data)
##
## Coefficients:
## (Intercept)           L4
##    0.425239    -0.005023
##
## Degrees of Freedom: 427 Total (i.e. Null);  426 Residual
## Null Deviance:       586.5
## Residual Deviance: 584.8     AIC: 588.8
##
## Outcome regression.
## No outcome regression performed.
##
##
## Thu Oct 22 13:02:57 2020
## Domains:
##  -1.000000e+01   <=  X1   <=    1.000000e+01
##  -1.000000e+01   <=  X2   <=    1.000000e+01
##  -1.000000e+01   <=  X3   <=    1.000000e+01
##  -1.000000e+01   <=  X4   <=    1.000000e+01
##
## Data Type: Floating Point
## Operators (code number, name, population)
##  (1) Cloning.........................  65
##  (2) Uniform Mutation.................  62
##  (3) Boundary Mutation................  62
##  (4) Non-Uniform Mutation.............  62
##  (5) Polytope Crossover...............  62
##  (6) Simple Crossover.................  62
##  (7) Whole Non-Uniform Mutation........  62
##  (8) Heuristic Crossover..............  62
##  (9) Local-Minimum Crossover..........  0
##
## HARD Maximum Number of Generations: 100
## Maximum Nonchanging Generations: 10
## Population size       : 500
## Convergence Tolerance: 1.000000e-03
##
## Not Using the BFGS Derivative Based Optimizer on the Best Individual Each Generation.
## Not Checking Gradients before Stopping.
## Using Out of Bounds Individuals.
##
## Maximization Problem.
##
##
## Generation#      Solution Value
##
##       0  -1.002247e+02
##
## 'wait.generations' limit reached.
## No significant improvement in 10 generations.
##
## Solution Fitness Value: -1.002247e+02
##
## Parameters at the Solution:
##
##  X[ 1] : -5.010370e+00
##  X[ 2] : -6.276860e+00
##  X[ 3] : -5.920860e+00
##  X[ 4] : 3.234605e+00
##
## Solution Found Generation 1
## Number of Generations Run 11
##
## Thu Oct 22 13:10:14 2020
## Total run time : 0 hours 7 minutes and 17 seconds
## Genetic Algorithm
## $value
## [1] -100.2247
```

```
##
## $par
## [1] -5.010370 -6.276860 -5.920860  3.234605
##
## $gradients
## [1] NA NA NA NA
##
## $generations
## [1] 11
##
## $peakgeneration
## [1] 1
##
## $popsize
## [1] 500
##
## $operators
## [1] 65 62 62 62 62 62 62 62  0
##
##
## $dp=1
## Recommended Treatments:
##    1
## 5000
## $dp=2
## Recommended Treatments:
##    0    1
##  297 4703
## $dp=3
## Recommended Treatments:
##    0    1
##  385 4615
## $dp=4
## Recommended Treatments:
##    0    1
##  428 4572
##
## Estimated Value: -100.2247
```

```
cat("Here we get a value of: ", estimator(vsObj), "\n and coefficients:\n")
```

```
## Here we get a value of:  -100.2247
##  and coefficients:
```

```
print(coef(vsObj))
```

```
## $propensity
## $propensity$`dp=1`
## $propensity$`dp=1`$`Subset=S12`
##  (Intercept)           L1
## -0.991800509  0.006118285
##
##
```

```
## $propensity$‘dp=2‘
## $propensity$‘dp=2‘$‘Subset=S22‘
##  (Intercept)           L2
## -0.595683184  0.003867191
##
## $propensity$‘dp=2‘$‘Subset=S21‘
##  (Intercept)           L2
##  1.438085356 -0.009560588
##
##
## $propensity$‘dp=3‘
## $propensity$‘dp=3‘$‘Subset=S32‘
##  (Intercept)           L3
## -0.527456219  0.002924772
##
## $propensity$‘dp=3‘$‘Subset=S31‘
##  (Intercept)           L3
##  0.218114607 -0.003470146
##
##
## $propensity$‘dp=4‘
## $propensity$‘dp=4‘$‘Subset=S42‘
##  (Intercept)           L4
## -0.625071351  0.002750375
##
## $propensity$‘dp=4‘$‘Subset=S41‘
##  (Intercept)           L4
##  0.425239492 -0.005022964
```

```
cat("\nUsing output from a run with a small population size (500), we get the optimal rule of giving a l
```

```
##
## Using output from a run with a small population size (500), we get the optimal rule of giving a high
```

b

This cutoff seems more reasonable since η is not directly on the boundary as in 3 and 4; however, it does come with a lower value.

6

a

In problem 1 we got the coefficients $\alpha_1 = 117.85292039$, $\alpha_2 = -0.39425908$ and $\alpha_3 = 0.00276527$. We use this to maximize eta.

$$\begin{aligned}
\widehat{V}(\eta) &= 117.85292039 + -0.39425908\eta + 0.00276527\eta^2 \\
\frac{dV}{d\eta} &= -0.39425908 + 0.00276527 \cdot 2\eta \overset{\text{set}}{=} 0 \\
\widehat{\eta} &= 71.2876
\end{aligned}$$

So our rule would be to give the low dose if they patient is currently experiencing a side effect of if their LDL is above 71.2876.

b

Notice that this estimate is higher than those given by the IPW and AIPW estimator. Also notice that this η value is outside of the range of [90,200]. This may explain the decreasing values that we saw from the bootstrap in question 1, perhaps we need to test a wider range.