```
      name:  <unnamed>
       log:  /Users/erinmarkiewitz/Dropbox/Phd_Coursework/Econ675/hw2\results\
> pset2_stata.smcl
  log type:  smcl
 opened on:  12 Oct 2018, 18:29:35

1 .
2 .
3 . ************************************
4 . ************ Question 1 ************
5 . ************************************
6 . /*
>    * Some values
>    global M = 1000 //number of iterations
>    global n = 1000
>    global hvalues .5 .6 .7 .8 0.8199 .9 1 1.1 1.2 1.3 1.4 1.5
>    mat hvalues = (0.8199, .5, .6, .7, .8, .9, 1, 1.1, 1.2, 1.3, 1.4, 1.5)
>
>    *DGP Values
>    global mu1 = -1.5
>    global mu2 = 1
>    global sd1 = sqrt(1.5)
>    global sd2 = 1
>
>    mata:
>    //*********FUNCTIONS***********
>    // function for calculating kernel
>    real scalar function kern(real scalar u){
>    return(.75*(1-u^2)*(abs(u)<=1))
>    }
>
>    // function for calculating true density
>    real scalar function f_true(real scalar u){
>    return(.5*normalden(u,-1.5,sqrt(1.5)) + .5*normalden(u,1,1))
>    }
>
>    // function for calculating MSE (LI & LO)
>    real vector function mse(real vector xdata, real scalar hvalue){
>    //Construct two matrices of xdata
>    M1 = J($n,$n,.) // n x n matrix with one column for each observation
>    M2 = J($n,$n,.) // n x n matrix with one row for each observation
>    for (i=1; i<= $n; i++) {
>    v = J($n,1,xdata[i])
>    M1[,i] = v
>    M2[i,] = v'
>    }
>
>    M3 = (M1-M2)/hvalue //object to be evaluated by kernel
>    M4 = J($n,$n,.)
>    M5 = J($n,$n,.)
>    fx = J($n,1,.)
>
>    for (i=1; i<=$n; i++){
>    for (j=1; j<=$n; j++){
>    M4[i,j] = kern(M3[i,j])
>    }
>    M5[i,] = M4[i,]
>    M5[i,i]=0
>
>    fx[i,1] = f_true(xdata[i])
>    }
>
>    fhat_LI = rowsum(M4)/($n*hvalue)
>    fhat_LO = rowsum(M5)/(($n-1)*hvalue)
>
>    sqe_LI = (fhat_LI-fx):^2
>    sqe_LO = (fhat_LO-fx):^2
>
>    mse_LI = mean(sqe_LI)
>    mse_LO = mean(sqe_LO)
>
>    return((mse_LI,mse_LO))
>    }
>
>    // function for importing/exporting to mata for mse calculation
>    void iteration(real scalar m){
>    x= st_data(.,.)
>    hvalues = st_matrix("hvalues")
>
>    mse = J(12,2,.)
>    for (h=1; h<=12; h++){
>    mse[h,] = mse(x,hvalues[1,h])
>    }
>    st_matrix("msetemp",mse)
>    }
>    end
>
>
>    *Empty matrix to be filled
>    mat msesum = J(12,2,0)
>
>    *Loop through iterations
>    timer on 1
>    forval m = 1/$M{
>    disp `m'
>    set obs $n
>
```

```
>    *equally weight two normal distributions
>    gen comps = uniform() >= .5
>
>    *generate sample
>    gen x = comps*rnormal($mu1,$sd1) + (1-comps)*rnormal($mu2,$sd2)
>    drop comps
>
>    *call mata function to calculate mse
>    mata iteration(`m')
>    drop x
>    mat msesum = msesum + msetemp
>    }
>    timer off 1
>    timer list
>
>    mat imse = msesum*1000
>    svmat imse
>    rename imse1 imse_li
>    rename imse2 imse_lo
>
>    egen h = fill(.5, .6, .7, .8, 0.8199, .9, 1, 1.1, 1.2, 1.3, 1.4, 1.5)
>
>    twoway(line imse_li h)(line imse_lo h), ytitle("IMSE (Thousands)") ///
>    xtitle("h") xline(0.8199) caption("Note: Vertical line is at h_AMSE")
>    graph export $resdir/pset2q1.png, replace
>
>    *?
>
>    **************
>    **** Problem 2
>    **************
>
>    **************
>    **** Problem 2a-b
>    ***************
>    set obs 1000
>
>
>    * Define cross validation function: CV(list, i): vars=variable list, i =
> max polynomial
>    mata
>    void CV(vars, i) {
>    st_view(y=., ., "y")
>    st_view(X=., ., tokens(vars))
>    XpX  = cross(X, X)
>    XpXinv  = invsym(XpX)
>    b  = XpXinv*cross(X, y)
>    w = diagonal(X*XpXinv*X')
>    muhat = X*b
>    num = (y - muhat):*(y - muhat)
>    den= (J(1000,1,1) - w):*(J(1000,1,1) - w)
>    div = num:/den
>    CV = mean(div)
>    CV
>    st_numscalar("mCV"+strofreal(i), CV)
>    }
>    end
>
>
>    * Program which runs the monte-carlo experiment
>    program CVsim, rclass
>    drop _all
>    set obs 1000
>    forvalues i = 0/20 {
>    gen CV`i' = 0
>    }
>    gen x = runiform(-1,1)
>    gen e = x^2*(rchi2(5)-5)
>    gen y = exp(-0.1*(4*x-1)^2)*sin(5*x)+e
>    forvalues i = 0/20 {
>    gen x`i' = x^`i'
>    }
>    forvalues i = 0/20 {
>    global xlist = "x0-x`i'"
>    di "$xlist"
>    mata CV("$xlist", `i')
>    replace CV`i' = mCV`i'
>    }
>    end
>
>
>
>    * Run the experiment
>    set seed 12345
>    simulate CV0=CV0 CV1=CV1 CV2=CV2 CV3=CV3 CV4=CV4 CV5=CV5 CV6=CV6 CV7=CV7
> CV8=CV8 ///
>    CV9=CV9 CV10=CV10 CV11=CV11 CV12=CV12 CV13=CV13 CV14=CV14 CV15=CV15 ///
>    CV16=CV16 CV17=CV17 CV18=CV18 CV19=CV19 CV20=CV20, reps(100) nodots: CVsi
> m
>    collapse *
>    gen i = 1
>    reshape long CV, i(i) j(k)
>    sort CV
>    local min = k[1]
>    twoway scatter CV k, ytitle("Mean CV") xtitle("K") xlabel(0(2)20) xmtick(
> 0(1)20) xline(`min') title("Average CV(K), across 1000 simulations")
>    graph export $resdir\pset2q2b.png, replace
>
```

```
>
>    **************
>    ***Problem 2c
>    ***************
>
>    * Program which runs the monte-carlo experiment for mu_0
>    program muhatsim, rclass
>    drop _all
>    set obs 1000
>    gen x = runiform(-1,1)
>    gen e = x^2*(rchi2(5)-5)
>    gen y = exp(-0.1*(4*x-1)^2)*sin(5*x)+e
>    forvalues p = 0/7 {
>    gen x`p' = x^`p'
>    }
>    reg y x0-x7, nocons
>    clear
>    set obs 11
>    gen n = _n
>    gen foo = 1
>    gen x = -1+(_n-1)/5
>    forvalues p = 0/7 {
>    gen x`p' = x^`p'
>    }
>    predict muhat
>    predict se, stdp
>    generate lb = muhat - invnormal(0.975)*se
>    generate ub = muhat + invnormal(0.975)*se
>
>
>
>    keep n muhat foo lb ub
>    reshape wide muhat lb ub, i(foo) j(n)
>    end
>
>
>
>    set seed 12345
>    simulate muhat1=muhat1 muhat2=muhat2 muhat3=muhat3 muhat4=muhat4 muhat5=m
> uhat5 ///
>    muhat6=muhat6 muhat7=muhat7 muhat8=muhat8 muhat9=muhat9 muhat10=muhat10 m
> uhat11=muhat11 ///
>    ub1=ub1 ub2=ub2 ub3=ub3 ub4=ub4 ub5=ub5 ub6=ub6 ub7=ub7 ub8=ub8 ub9=ub9 u
> b10=ub10 ub11=ub11 ///
>    lb1=lb1 lb2=lb2 lb3=lb3 lb4=lb4 lb5=lb5 lb6=lb6 lb7=lb7 lb8=lb8 lb9=lb9 l
> b10=lb10 lb11=lb11, reps(1000) nodots: muhatsim
>    gen i = _n
>    reshape long muhat ub lb, i(i) j(grid)
>    collapse muhat ub lb, by(grid)
>    gen x = -1+ (grid-1)/5
>    twoway (function y = exp(-0.1*(4*x-1)^2)*sin(5*x), range(-1 1) lcolor(red
> )) ///
>    (line muhat x, lcolor(gs6)) (line lb x, lcolor(gs6) lpattern(dash)) (line
>  ub x, lcolor(gs6) lpattern(dash)), ///
>    legend(order(1 "DGP" 2 "Prediction" 3 "Confidence Interval") rows(1)) yti
> tle(Y) xtitle(X) title("Mu_hat(x) across 1000 simulations")
>    graph export $resdir\pset2q2c.png, replace
>
>
>    **************
>    ***Problem 2d
>    ***************
>
>    * Program which runs the monte-carlo experiment for mu_1
>    program dmuhatsim, rclass
>    drop _all
>    set obs 1000
>    gen x = runiform(-1,1)
>    gen e = x^2*(rchi2(5)-5)
>    gen y = exp(-0.1*(4*x-1)^2)*((0.8-3.2*x)*sin(5*x)+5*cos(5*x)) + e
>    forvalues p = 0/7 {
>    gen x`p' = x^`p'
>    }
>    reg y x0-x7, nocons
>    clear
>    set obs 11
>    gen n = _n
>    gen foo = 1
>    gen x = -1+(_n-1)/5
>    forvalues p = 0/7 {
>    gen x`p' = x^`p'
>    }
>    predict dmuhat
>    predict se, stdp
>    generate lb = dmuhat - invnormal(0.975)*se
>    generate ub = dmuhat + invnormal(0.975)*se
>
>
>
>    keep n dmuhat foo lb ub
>    reshape wide dmuhat lb ub, i(foo) j(n)
>    end
>
>
>    set seed 12345
>    simulate dmuhat1=dmuhat1 dmuhat2=dmuhat2 dmuhat3=dmuhat3 dmuhat4=dmuhat4
> dmuhat5=dmuhat5 ///
>    dmuhat6=dmuhat6 dmuhat7=dmuhat7 dmuhat8=dmuhat8 dmuhat9=dmuhat9 dmuhat10=
   > dmuhat10 dmuhat11=dmuhat11 ///
   >    ub1=ub1 ub2=ub2 ub3=ub3 ub4=ub4 ub5=ub5 ub6=ub6 ub7=ub7 ub8=ub8 ub9=ub9 u
   > b10=ub10 ub11=ub11 ///
   >    lb1=lb1 lb2=lb2 lb3=lb3 lb4=lb4 lb5=lb5 lb6=lb6 lb7=lb7 lb8=lb8 lb9=lb9 l
   > b10=lb10 lb11=lb11, reps(1000) nodots: dmuhatsim
   >    gen i = _n
   >    reshape long dmuhat ub lb, i(i) j(grid)
   >    collapse dmuhat ub lb, by(grid)
   >    gen x = -1+ (grid-1)/5
   >    twoway (function y = exp(-0.1*(4*x-1)^2)*((0.8-3.2*x)*sin(5*x)+5*cos(5*x)
   > ), range(-1 1) lcolor(red)) ///
   >    (line dmuhat x, lcolor(gs6)) (line lb x, lcolor(gs6) lpattern(dash)) (lin
   > e ub x, lcolor(gs6) lpattern(dash)), ///
   >    legend(order(1 "DGP" 2 "Prediction" 3 "Confidence Interval") rows(1)) yti
   > tle(Y) xtitle(X) title("(d/dx)*Mu_hat(x) across 1000 simulations")
   >    graph export $resdir\pset2q2d.png, replace
   >
   >
   >
   >
   > */ */
 7 .
 8 . *******
 9 . *** Problem 3
10 . *******
11 .
12 . drop _all

13 . set obs 1000
   number of observations (_N) was 0, now 1,000

14 . local theta = 1

15 . local d = 5

16 . local n = 500
```

```
17 .
18 . forvalues p = 1/14 {
     2.         gen v_hat`p' = .
     3.         gen theta_hat`p' = .
     4.
19 . }
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)
   (1,000 missing values generated)

20 .
21 .
```

```
22 . mata:
   ───────────────────────────────────────────────── mata (type end to exit) ───
   :         void polyloop(i) {
   >                 X       = uniform(`n',`d'):*2 :-1
   >                 ep      = invnormal(uniform(`n',1)):*0.3637899:*(1 :+ rowsum
   > (X:^2))
   >                 gx      = exp(rowsum(X:^2))
   >                 T       = invnormal(uniform(`n',1)) + rowsum(X:^2):^.5 :>= 0
   >                 Y   = T + gx + ep
   >                 cons= J(500,1,1)
   >                 /*Raising to single powers */
   >                 X2      = X:^2
   >                 X3      = X:^3
   >                 X4      = X:^4
   >                 X5      = X:^5
   >                 X6      = X:^6
   >                 X7      = X:^7
   >                 X8      = X:^8
   >                 X9      = X:^9
   >                 X10 = X:^10
   >                 /*Kronekering, but this creates some duplicates*/
   >                 X1k = X#X
   >                 X2k = X2#X2
   >                 X3k = X3#X3
   >                 X4k = X4#X4
   >                 /* Manually removing duplicates...might be a better way to d
   > o this */
   >                 X1k = X1k[1::`n',2::5], X1k[1::`n', 8::10], X1k[1::`n',14::1
   > 5], X1k[1::`n', 20]
   >                 X2k = X2k[1::`n',2::5], X2k[1::`n', 8::10], X2k[1::`n',14::1
   > 5], X2k[1::`n', 20]
   >                 X3k = X3k[1::`n',2::5], X3k[1::`n', 8::10], X3k[1::`n',14::1
   > 5], X3k[1::`n', 20]
   >                 X4k = X4k[1::`n',2::5], X4k[1::`n', 8::10], X4k[1::`n',14::1
   > 5], X4k[1::`n', 20]
   >                 A = asarray_create("real",1)
   >                 asarray(A,1,X)
   >                 asarray(A,2,(asarray(A,1),X2))
   >                 asarray(A,3,(asarray(A,2),X1k))
   >                 asarray(A,4,(asarray(A,3),X3))
   >                 asarray(A,5,(asarray(A,4),X2k))
   >                 asarray(A,6,(asarray(A,5),X4))
   >                 asarray(A,7,(asarray(A,6),X3k))
   >                 asarray(A,8,(asarray(A,7),X5))
   >                 asarray(A,9,(asarray(A,8),X4k))
   >                 asarray(A,10,(asarray(A,9),X6))
   >                 asarray(A,11,(asarray(A,10),X7))
   >                 asarray(A,12,(asarray(A,11),X8))
   >                 asarray(A,13,(asarray(A,12),X9))
   >                 asarray(A,14,(asarray(A,13),X10))
   >                 theta_hat = I(1,14):*0
   >                 se_hat = I(1,14):*0
   >                 k_hat = I(1,14):*0
   >
   >                 for (j=1; j<=14; j++) {
   >                         Z = qrsolve(cons,(T,asarray(A,j)))
   >                         ZZ  = Z*Z'
   >                         Yhat = ZZ*Y
   >                         W = diag(ZZ)
   >                         ///CV_out = diag(mean_vec*(Y - Yhat) / (1 - W)^2)//
   >
   >                         ZQ = (cons,asarray(A,j))*invsym((cons,asarray(A,j))'
   > *(cons,asarray(A,j)))*(cons,asarray(A,j))'
   >                         M = I(`n') - ZQ
   >                         YM = M*Y
   >                         TM = M*T
   >                         theta_hat[1,j] = (TM'*YM) / (TM'*TM)
   >                         sigma = diag(ZQ*(Y-T*theta_hat[1,j]))
   >                         se_hat[1,j] = sqrt(invsym(T'*ZQ*T)*(T'*ZQ*sigma*ZQ*T
   > )*invsym(T'*ZQ*T))
   >                         st_store(i, "v_hat"+strofreal(j), se_hat[1,j])
   >                         st_store(i, "theta_hat"+strofreal(j), theta_hat[1,j]
   > )
   >
   >                         }
   >                         }
   note: variable k_hat set but not used
   note: variable Yhat set but not used
   note: variable W set but not used

   :                         end
   ──────────────────────────────────────────────────────────────────────────

23 . forvalues i = 1/10 {
  2.          mata polyloop(`i')
  3. }
```

```
24 .
25 . save output_q3_temp.dta, replace
   file output_q3_temp.dta saved

26 .
27 . use output_q3,clear

28 . gen obs = _n

29 . reshape long v_hat theta_hat, i(obs) j(k)
   (note: j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14)

   Data                               wide   ->   long
   ─────────────────────────────────────────────────────────────────────────────

   Number of obs.                     1000   ->   14000
   Number of variables                  29   ->       4
   j variable (14 values)                    ->   k
   xij variables:
                 v_hat1 v_hat2 ... v_hat14   ->   v_hat
     theta_hat1 theta_hat2 ... theta_hat14   ->   theta_hat
   ─────────────────────────────────────────────────────────────────────────────

30 .
31 . gen abs_theta_hat = abs(theta_hat)
   (13,860 missing values generated)

32 . egen sd_theta_hat = sd(theta_hat) ,by(k)

33 . gen coverage_rate = 1 if abs_theta_hat <= 1.96 * sd_theta_hat/sqrt(1000)
   (13,990 missing values generated)

34 . replace coverage_rate = 0 if coverage_rate == .
   (13,990 real changes made)

35 .
36 . collapse (mean) coverage_rate mean_v_hat= v_hat  mean_theta_hat=theta_hat (s
   > d) sd_theta_hat = theta_hat, by(k)
```

```
37 . gen mean_bias = mean_theta_hat  - 1

38 . gen v_theta_hat = sd_theta_hat^2

39 . 
40 . *coverage rate test
41 . log close
        name:  <unnamed>
         log:  /Users/erinmarkiewitz/Dropbox/Phd_Coursework/Econ675/hw2\results\
  > pset2_stata.smcl
    log type:  smcl
   closed on:  12 Oct 2018, 18:29:40
  ──────────────────────────────────────────────────────────────────────────────
```