

```

clear
set scheme sj
set seed 569231
program drop _all
*-----
* Program to generate an artificial MCMC analysis
* from a multivariate normal mixture
*-----
program mixconvSim
    syntax newvarlist , Mean(name) Covariance(name) Rho(real) ///
        Initial(name) Updates(integer) clear [ step Proportions(numlist) Component(name) ]

*-----
* number of components
*-----
    local p = rowsof(`covariance')
    local q = colsof(`covariance')
    local d = `q' / `p'
    local N = `updates'
    clear
    set obs `N'
    local i = 0
    foreach v of local varlist {
        local ++i
    }
    qui gen `v' = `initial'[`i',1]
    }
    if `i' != `p' {
        di as err "`i' variables but `p' dimensions"
        exit(499)
    }
    if `d' > 1 & "`component'" != "" qui gen `component' = 1
    forvalues i=1/`d' {
        tempvar m`i' v`i' cS`i' L`i' iv`i'
        matrix `m`i'' = `mean'[1..`p',`i']
        local i1 = (`i'-1)*`p'+1
        local i2 = `i'*`p'
        matrix `v`i'' = `covariance'[1..`p',`i1'..'i2']
        matrix `iv`i'' = invsym(`v`i'')
        matrix `cS`i'' = (1-`rho'*`rho')*`v`i''
        matrix `L`i'' = cholesky(`cS`i'')
        local ld`i' = log(det(`v`i''))
    }
    if `d' > 1 {
        local i = 0
        local s = 0
        foreach pr of numlist `proportions' {
            local ++i
            local pr`i' = `pr'
            local s = `s' + `pr'
        }
        if `i' != `d' {
            di as err "`i' proportions but `d' components"
            exit(499)
        }
        forvalues i=1/`d' {
            local pr`i' = `pr`i' / `s'
        }
        if abs(`s'-1) > .0001 {
            di as err "Warning: proportions scaled to sum to one"
        }
    }
}

*-----
* temporary matrices
*-----
    tempname x0 z y T cM
    matrix `x0' = `initial'
    matrix `z' = J(`p',1,0)

*-----
* Generate the chain for iteration k
*-----
    forvalues k=2/`N' {

*-----
* Select Component
*-----
        local c = 1
        if `d' > 1 {
            forvalues i=1/`d' {

```

```

matrix `T' = (`x0' - `m`i') * `iv`i' * (`x0' - `m`i')
local LL`i' = -0.5 * `T'[1,1] - 0.5 * `ld`i'
if `i' > 1 local mLL = max(`mLL', `LL`i')
else local mLL = `LL`i'
}
local s = 0
forvalues i=1/`d' {
    local p`i' = `pr`i' * exp(`LL`i' - `mLL')
    local s = `s' + `p`i'
}
local u = runiform() * `s'
local c = 1
local t = `pl'
while `u' > `t' {
    local ++c
    local t = `t' + `p`c'
}
}
forvalues i=1/`p' {
    matrix `z'[`i',1] = rnormal()
}
matrix `cM' = `m`c' + `rho' * (`x0' - `m`c')
matrix `y' = `cM' + `L`c' * `z'
local i = 0
foreach v of local varlist {
    local ++i
    qui replace `v' = `y'[`i',1] in `k'
}
if `d' > 1 & "&component" != "" qui replace `component' = `c' in `k'
matrix `x0' = `y'
}
end
*-----
* program to create a matrix plot that distinguishes
* one section of the chain from the rest
*-----
program convPlot
    syntax varlist [if] [in], from(integer) to(integer)

    marksample touse
    local p : word count `varlist'
    local i = 0
    local k = 0
    local plotlist ""
    local hole ""
    foreach v1 of varlist `varlist' {
        local ++i
        local j = 0
        foreach v2 of varlist `varlist' {
            local ++j
            if `i' < `j' {
                local ++k
                twoway (scatter `v1' `v2' if (_n < `from' | _n > `to') & `touse' , ///
                    mcol(blue) msy(circle) ///
                    (scatter `v1' `v2' if _n >= `from' & _n <= `to' & `touse' , ///
                    c(1) mcol(red) msy(circle) lpat(dash)) , ///
                    legend(off) nodraw
                qui graph save plot`k'.gph , replace
                local plotlist "`plotlist' plot`k'.gph"
            }
            else if `i' < `p' & `j' > 1 {
                local ++k
                local hole "`hole' `k'"
            }
        }
    }
    graph combine `plotlist' , hole(`hole')
end
*-----
* Generate the chain
*-----
matrix S = (2, 2, 2 \ 2, 3, 3 \ 2, 3, 4)
matrix M = (5 \ 5 \ 5)
matrix Init = (4 \ 10 \ 0)
mixconvSim theta1 theta2 theta3 , mean(M) init(Init) rho(0.95) ///
    updates(1000) cov(S) clear

```

```
*-----  
* Show burn-in  
*-----  
convPlot theta1 theta2 theta3 , from(1) to(100)  
*-----  
* show a portion of the converged chain  
*-----  
convPlot theta1 theta2 theta3 , from(201) to(300)  
*-----  
* Marginal trace plots  
*-----  
mcmctrace theta1 theta2 theta3  
*-----  
* Marginal cusum plots  
*-----  
mcmccusum theta1 theta2 theta3 , ref(2)  
*-----  
* Mahal plot using all points to estimate the means etc  
*-----  
mcmcmahal theta1 theta2 theta3  
*-----  
* Mahal plot using second half to estimate the means etc  
*-----  
mcmcmahal theta1 theta2 theta3, p(0.5)
```