**Turing models**

**Regression model using MvNormal specifications in Turing model – WORKS**

@model md(X, y) = begin

D = size(X, 2)

sigma ~ TruncatedNormal(0, 1, 0, Inf)

alpha ~ Normal(0, 5)

beta ~ MvNormal(zeros(D), ones(D))

lp = alpha .+ X \* beta

y ~ MvNormal(lp, sigma)

end

# WORKS WITH NUTS

samples = 2000

#X = rand(10, 52)

#y = rand(10)

model = md(X, y)

alg = NUTS(samples, 0.65)

chain = sample(model, alg)

show(chain)

plot(chain)

**Two equation SUR that works**

@model SUR(y1, X1, y2, X2, n, k1, k2) = begin

# Set variance prior.

S = zeros(2,2) + I.\*40.00

Sigma ~ Wishart(60, S)

iSigma = Matrix(Hermitian(inv(Sigma)))

# Set the priors on our coefficients.

b1 = Array{Real}(undef, k1)

b2 = Array{Real}(undef, k2)

for i in 1:k1

b1[i] ~ Normal(0, 100)

end

for i in 1:k2

b2[i] ~ Normal(0, 100)

end

# Calculate all the mu terms.

mu1 = X1 \* b1

mu2 = X2 \* b2

# likelihood

for i = 1:n

yy = [y1[i], y2[i]]

mu = [mu1[i], mu2[i]]

yy ~ MvNormal(mu, iSigma)

end

end

**WORKS BUT – estimates a yy[1] and yy[2] as parameters?!**

**Heteroskedastic error model:**

@model md(X, y) = begin #, ::Type{TV}=Vector{Float64}) where {TV} = begin

n, k = size(X)

# sigma ~ TruncatedNormal(0, 1, 0, Inf)

alpha ~ Normal(0, 5)

λ = Array{Real}(undef,n)

for i in 1:n

λ[i] ~ Chisq(10)

end

beta ~ MvNormal(zeros(k), ones(k))

S = λ #.\*ones(n)

# S = Symmetric(Matrix{TV}((λ.\*ones(n))

# S = Vector{Real}(diagm(λ))

lp = alpha .+ X \* beta

y ~ MvNormal(lp, S)

end

**\*\*\*\* Following works!**

@model mdhet(X, y) = begin #, ::Type{TV}=Vector{Float64}) where {TV} = begin

n, k = size(X)

# sigma ~ TruncatedNormal(0, 1, 0, Inf)

alpha ~ Normal(0, 5)

λ = Array{Real}(undef,n)

for i in 1:n

λ[i] ~ Chisq(6)

end

beta ~ MvNormal(zeros(k), ones(k))

S = λ #.\*ones(n)

# S = Symmetric(Matrix{TV}((λ.\*ones(n))

# S = Vector{Real}(diagm(λ))

lp = alpha .+ X \* beta

y ~ MvNormal(lp, S)

end

samples = 2000

# alg = HMC(samples, 0.005, 5)

alg = NUTS(samples, 0.65)

model = mdhet(X,y)

chain = sample(model, alg)

@show(show(chain))

**FOLLOWING WORKS WITH LAMBDAS scales to one (including sigma)**

**Code in: reg\_with\_MvNormal\_heterosked\_working\_11\_2019.jl**

#### NEED TO SCALE LAMBDAS TO MEAN OF ONE

@model mdhet(X, y) = begin #, ::Type{TV}=Vector{Float64}) where {TV} = begin

n, k = size(X)

# sigma ~ TruncatedNormal(0, 1, 0, Inf)

alpha ~ Normal(0, 5)

tau = Array{Real}(undef,n)

sigma ~ Uniform(0.0001, 20.0)

for i in 1:n

tau[i] ~ Chisq(4)

end

beta ~ MvNormal(zeros(k), ones(k).\*10)

S = sigma.\*(4.0 ./tau) #.\*ones(n)

# S = Symmetric(Matrix{TV}((λ.\*ones(n))

# S = Vector{Real}(diagm(λ))

lp = alpha .+ X \* beta

y ~ MvNormal(lp, S)

end

samples = 2000

# alg = HMC(samples, 0.005, 5)

alg = NUTS(samples, 0.65)

model = mdhet(X,y)

chain = sample(model, alg)

@show(show(chain))

lambda1 = 4.0 ./Array(chain["tau[1]"])

mean(lambda1)

lambda2 = 4.0 ./Array(chain["tau[2]"])

mean(lambda2)

lambda6 = 4.0 ./Array(chain["tau[6]"])

mean(lambda6)

lambda175 = 4.0 ./Array(chain["tau[175]"])

mean(lambda175)

lambda15 = 4.0 ./Array(chain["tau[15]"])

mean(lambda15)

plot(lambda6, st=:density)

plot!(lambda1, st=:density)

plot!(lambda2, st=:density)

plot!(lambda175, st=:density)

plot!(lambda15, st=:density)

sigmas = Array(chain["sigma"])

plot(sigmas, st=:density, fill=true)

**results for heterosked dgp**

Summary Statistics

│ Row │ parameters │ mean │ std │ naive\_se │ mcse │ ess │ r\_hat │

│ │ Symbol │ Float64 │ Float64 │ Float64 │ Float64 │ Any │ Any │

├─────┼────────────┼──────────┼───────────┼─────────────┼─────────────┼─────────┼──────────┤

│ 1 │ alpha │ 0.944608 │ 0.0226089 │ 0.000714957 │ 0.000860407 │ 754.083 │ 1.00066 │

│ 2 │ beta[1] │ 0.949515 │ 0.0196704 │ 0.000622032 │ 0.00041435 │ 1066.8 │ 0.999257 │

│ 3 │ beta[2] │ 1.04335 │ 0.0237556 │ 0.000751219 │ 0.000645351 │ 872.102 │ 0.999016 │

│ 4 │ sigma │ 0.226908 │ 0.0184154 │ 0.000582346 │ 0.000648016 │ 568.737 │ 1.00016 │

│ 5 │ tau[1] │ 4.16611 │ 2.15787 │ 0.0682379 │ 0.059186 │ 1665.46 │ 0.999201 │

│ 6 │ tau[2] │ 5.66485 │ 3.3417 │ 0.105674 │ 0.0594127 │ 1609.89 │ 0.99974 │

│ 7 │ tau[3] │ 3.12536 │ 1.52982 │ 0.0483773 │ 0.0288954 │ 2070.29 │ 1.00114 │

│ 8 │ tau[4] │ 3.86332 │ 1.96587 │ 0.0621662 │ 0.0373085 │ 1456.53 │ 0.99902 │

│ 9 │ tau[5] │ 5.01121 │ 2.8465 │ 0.0900142 │ 0.0505424 │ 1727.05 │ 0.999024 │

│ 10 │ tau[6] │ 1.72461 │ 0.758999 │ 0.0240017 │ 0.0115499 │ 1477.29 │ 0.999 │

│ 11 │ tau[7] │ 5.57482 │ 3.23592 │ 0.102329 │ 0.0401273 │ 1816.32 │ 0.999007 │

│ 12 │ tau[8] │ 2.8505 │ 1.37186 │ 0.0433821 │ 0.0143385 │ 2029.79 │ 0.999297 │

│ 13 │ tau[9] │ 5.71679 │ 3.1965 │ 0.101082 │ 0.0926802 │ 1598.77 │ 0.999039 │

│ 14 │ tau[10] │ 2.51457 │ 1.19224 │ 0.0377021 │ 0.0261078 │ 1585.2 │ 0.999535 │

│ 15 │ tau[11] │ 4.86317 │ 2.62648 │ 0.0830566 │ 0.0492687 │ 1840.17 │ 0.999171 │

│ 16 │ tau[12] │ 3.14445 │ 1.48515 │ 0.0469645 │ 0.0352249 │ 1537.76 │ 0.999306 │

│ 17 │ tau[13] │ 5.48432 │ 3.08467 │ 0.0975457 │ 0.051731 │ 1937.86 │ 0.999173 │

│ 18 │ tau[14] │ 4.91364 │ 2.7732 │ 0.0876964 │ 0.0322959 │ 1792.25 │ 0.999282 │

│ 19 │ tau[15] │ 4.17812 │ 2.17379 │ 0.0687414 │ 0.041618 │ 1658.9 │ 1.00061 │

│ 20 │ tau[16] │ 2.04496 │ 0.957602 │ 0.030282 │ 0.0204841 │ 1571.51 │ 0.999412 │

│ 21 │ tau[17] │ 5.36849 │ 2.98732 │ 0.0944675 │ 0.0468763 │ 1786.36 │ 0.999161 │

│ 22 │ tau[18] │ 2.43889 │ 1.14718 │ 0.0362772 │ 0.021877 │ 1726.37 │ 0.99969 │

│ 23 │ tau[19] │ 5.58848 │ 3.07823 │ 0.0973421 │ 0.0540047 │ 1874.5 │ 0.99906 │

│ 24 │ tau[20] │ 4.04773 │ 2.1775 │ 0.0688588 │ 0.0524088 │ 1447.13 │ 1.00001 │

Etc.