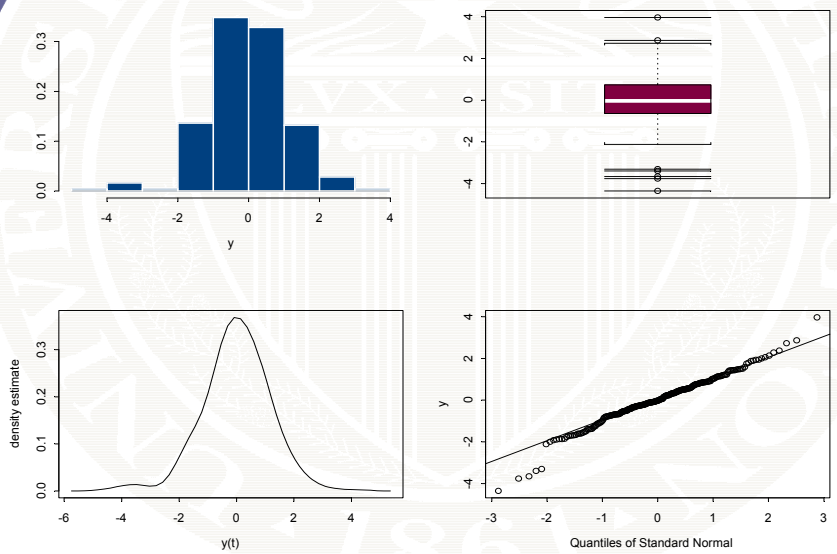
Nonlinear GMM Examples

Econ 583
Eric Zivot
Winter 2013
Updated: November 8, 2010

Simulated Student's t data with 10 degrees of freedom; T=250



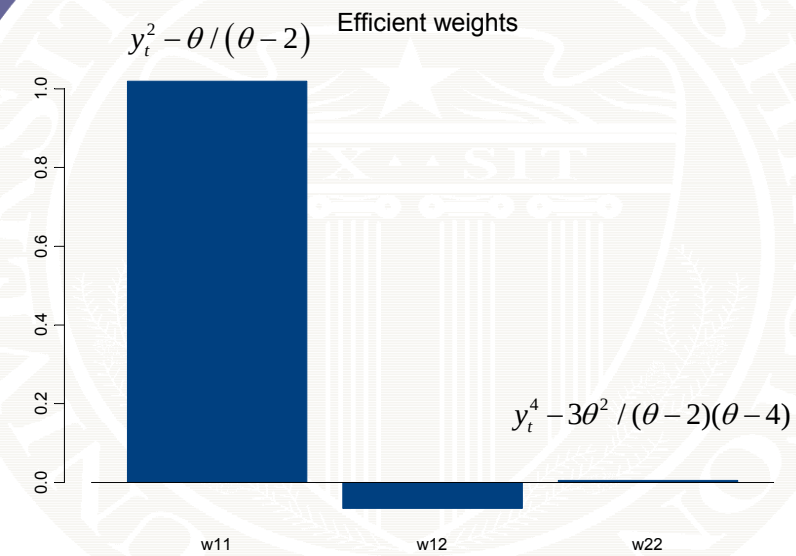
```
# function to compute moment conditions for
# estimating df parameter from Student-t distribution
t.moments <- function(parm,data=NULL) {
# parm = df parameter
# data = [y^2, y^4]
m1 = parm/(parm - 2)
m2 = 3*parm*parm/((parm - 2)*(parm - 4))
t(t(data) - c(m1,m2))
}

> start.vals = 15
> names(start.vals) = c("theta")
> t.gmm.iter = GMM(start.vals,t.moments,
+      method="iterative",max.steps=100,
+      data=data)
```

Iterated Efficient GMM Estimation

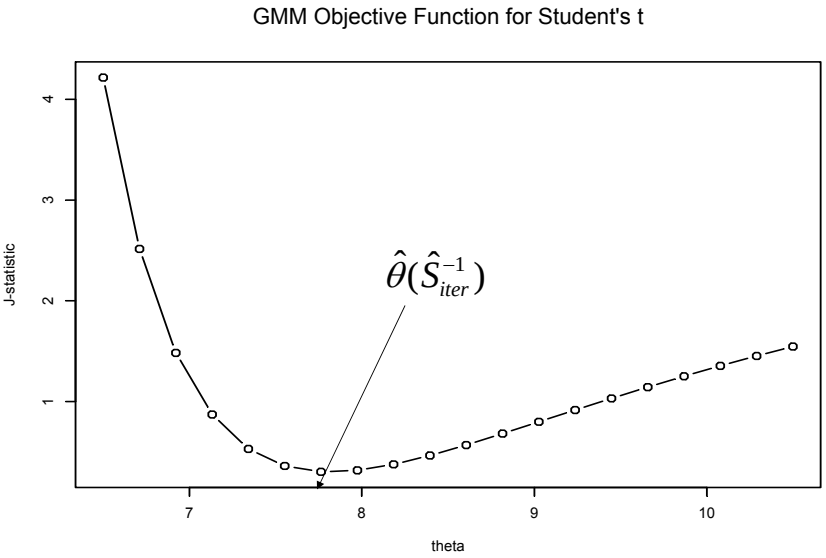
```
Coefficients:
      Value Std.Error t value Pr(>|t|)
theta 7.815 1.123     6.959   0.000

Optimization Info:
Number of Iterative Steps: 3

Test of Overidentification:
 J-stat Df P.value
 0.3025 1  0.5823
```

Do not reject specification of model

Efficient weights



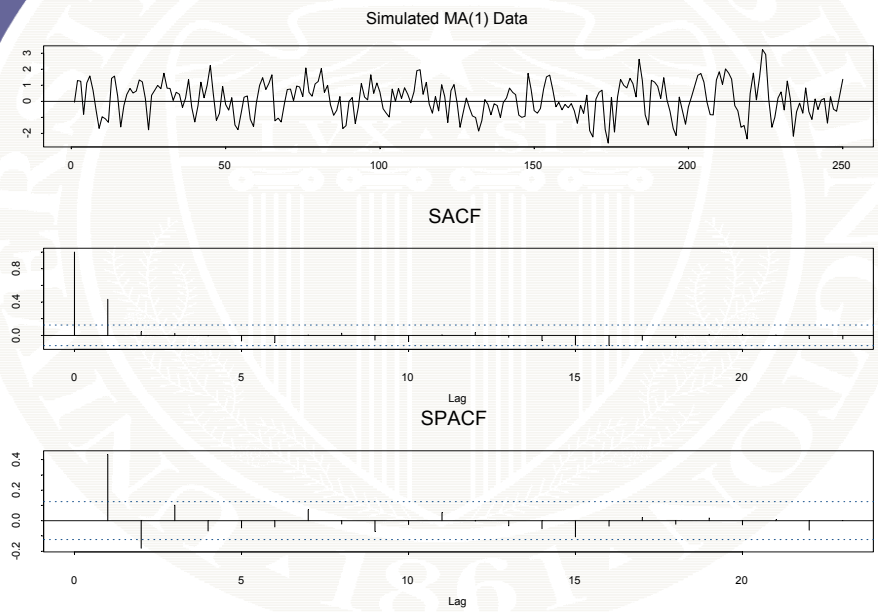

```
> t.gmm.iter$weight.matrix
```



Tracing Out the GMM Objective Function

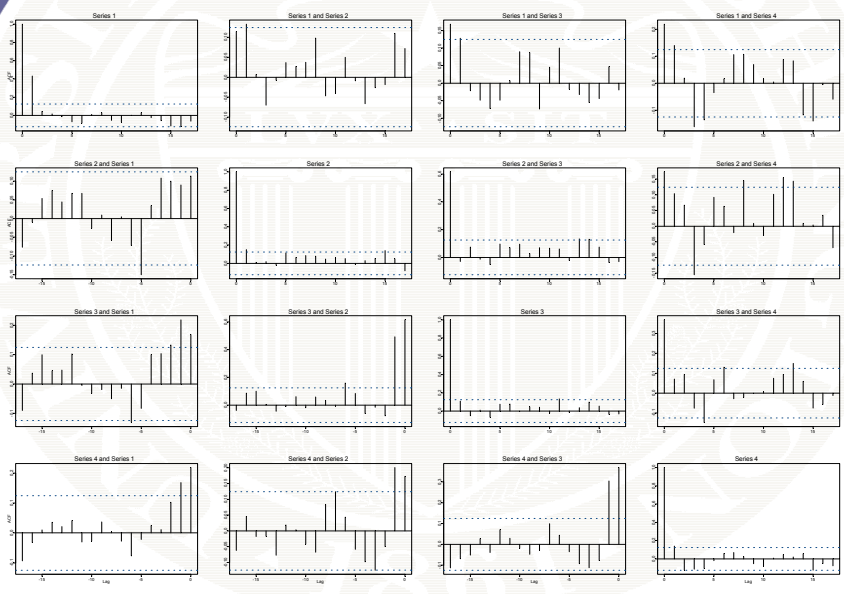
```
> s.hat.inv = t.gmm.iter$weight.matrix
> s.hat = solve(s.hat.inv)
> s.hat = 0.5*(s.hat + t(s.hat))

> n.theta=20
> theta.vals = seq(6.5, 10.5, length=n.theta)
> t.gmm.obj = rep(0,n.theta)
> for (i in 1:20) {
+ t.gmm.obj[i] = GMM(theta.vals[i],t.moments,
+                    w = s.hat,w0.efficient=T,
+                    method="iterative",max.steps=0,
+                    control=nlregb.control(iter.max=0),
+                    trace=F, data=data)$objective
}
```

MA(1) process: θ = 0.5, σ = 1, T=250



```
# function to compute four moments from MA(1) model
# y(t) = mu + e(t) + psi*e(t-1)
# e(t) ~ iid (0,sig2)
ma1.moments <- function(parm,data=NULL) {
# parm = (mu,psi,sig2)'
# data = (y(t),y(t)^2,y(t)*y(t-1),y(t)*y(t-2))
# is assumed to be a matrix
m1 = parm[1]
m2 = parm[1]^2 + parm[3]*(1 + parm[2]^2)
m3 = parm[1]^2 + parm[3]*parm[2]
m4 = parm[1]^2
t(t(data) - c(m1,m2,m3,m4))
}
```

Cross lag correlations of sample moments



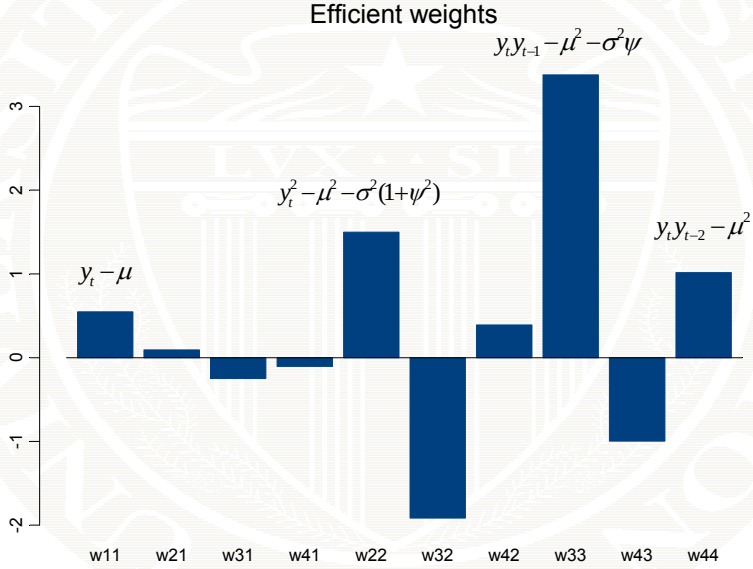
```
start.vals = c(0,0.5,1)
names(start.vals) = c("mu","psi","sig2")

# estimate using truncated kernel with
# bandwith = 1
ma1.gmm.trunc = GMM(start.vals,
                    ma1.moments,
                    data=ma1.data,
                    method="iterative",
                    ts=T,
                    var.hac.control=
                    var.hac.control(bandwidth=1,
                    window="truncated"))
```

Iterated Efficient GMM Estimation

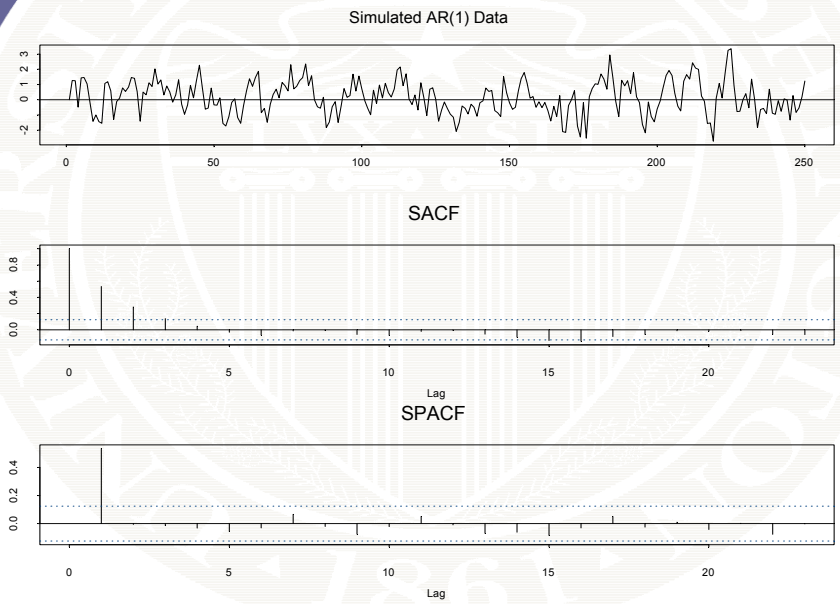
```
Coefficients:
      Value Std.Error t value Pr(>|t|)
  mu  0.1018  0.0927    1.0980  0.2733
 psi  0.5471  0.0807    6.7788  0.0000
sig2  0.8671  0.0763   11.3581  0.0000

Optimization Info:
Number of Iterative Steps: 3

 Test of Overidentification:
  J-stat Df P.value
  0.3549 1  0.5513
```

Do not reject specification of model

Efficient weights



Simulated AR(1) data

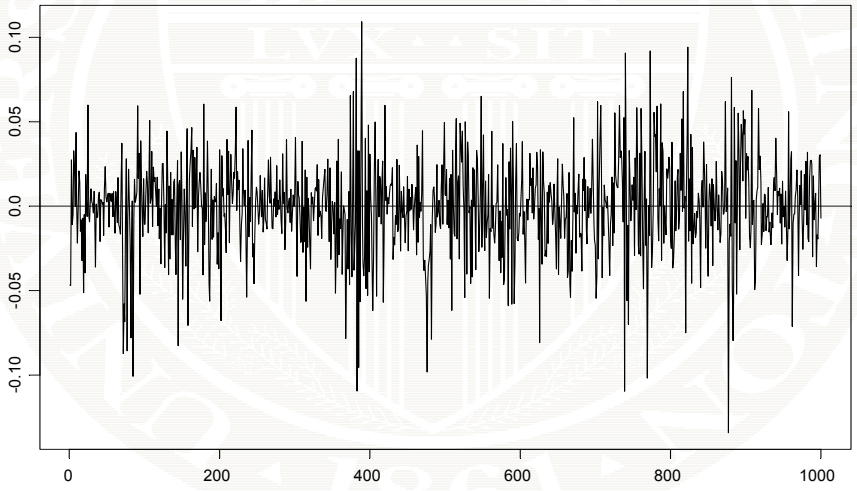


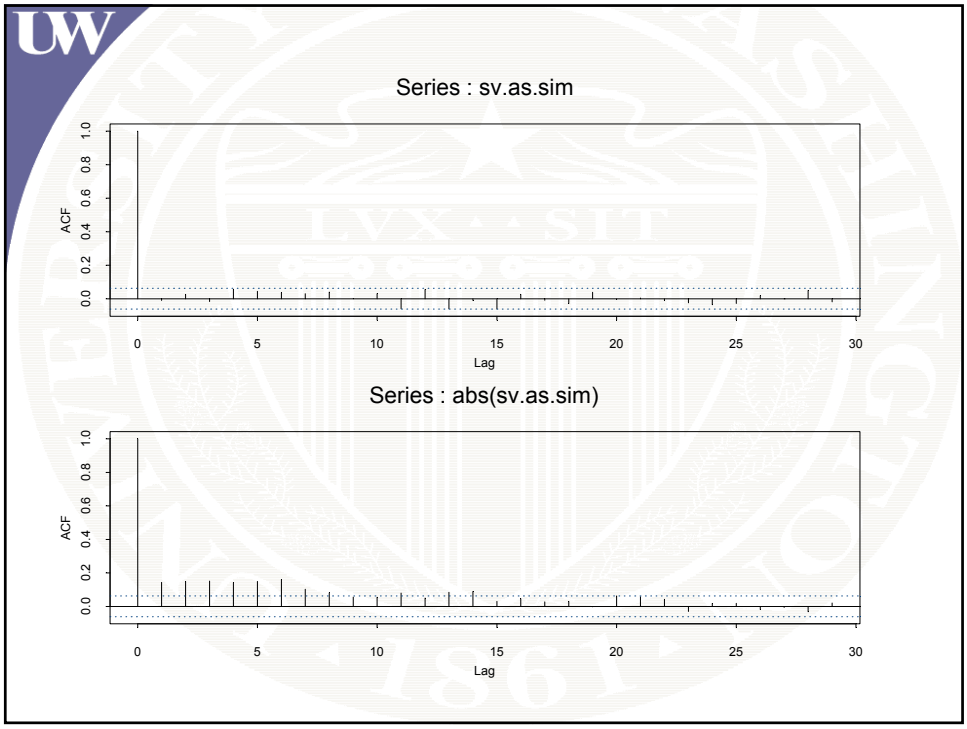
GMM estimation of misspecified MA(1) model

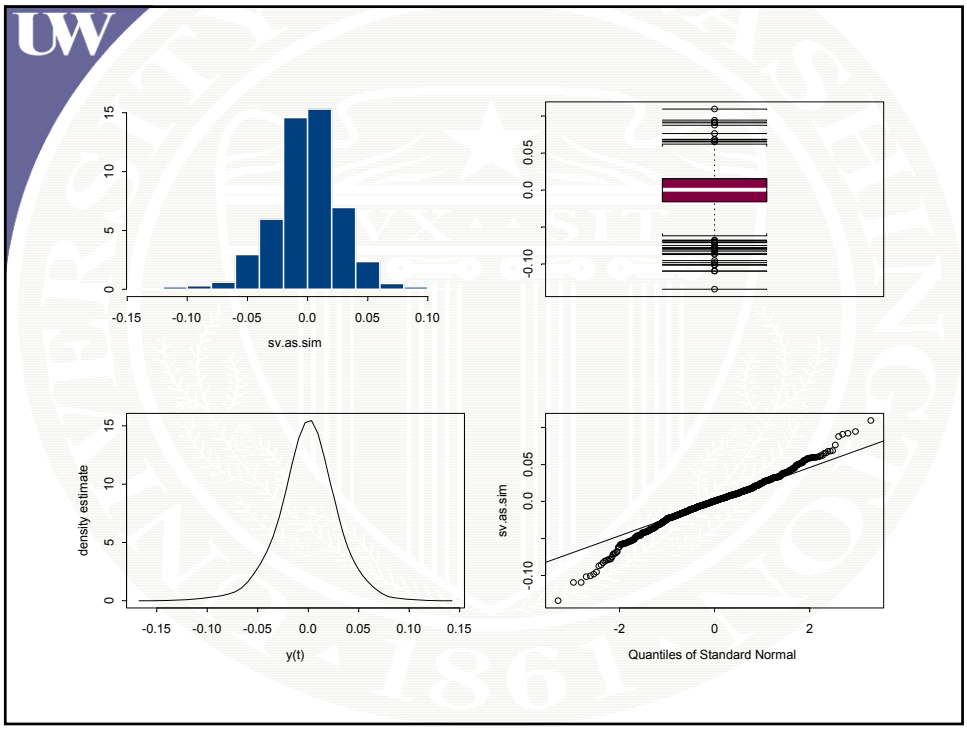
```
Coefficients:
       Value Std.Error  t value Pr(>|t|)
  mu -0.0407    0.0909  -0.4478   0.6547
 psi  0.4199    0.0789   5.3207   0.0000
sig2  0.7840    0.0737  10.6391   0.0000

Optimization Info:
Number of Iterative Steps: 9

Test of Overidentification:
 J-stat Df P.value
 13.06  1  0.0003

  Reject MA(1) specification!
```

Simulated data from SV model with

ω_0=-0.736, β_0 = 0.90, $\sigma_{0,u}$ = 0.363




Series : sv.as.sim
ACF
Lag
Series : abs(sv.as.sim)
ACF
Lag


sv.as.sim
density estimate
y(t)
sv.as.sim
Quantiles of Standard Normal

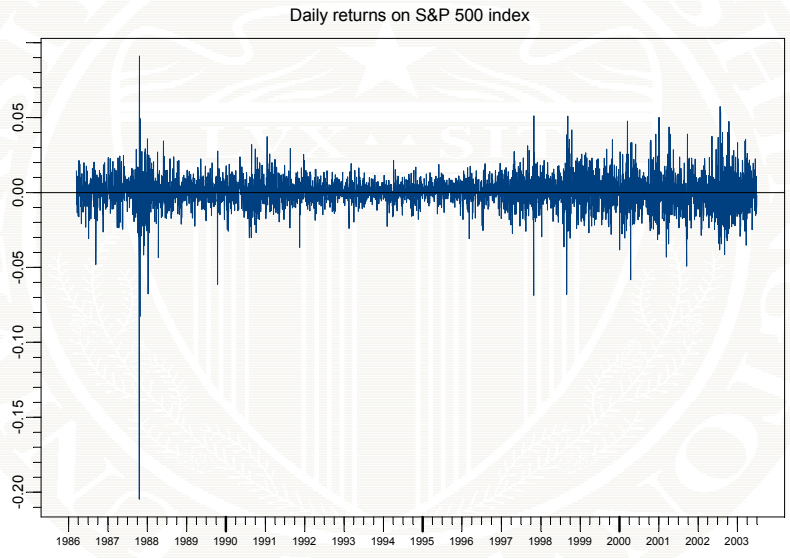
```
## define moment equations
sv.moments = function(parm, data=NULL)
{
omega = parm[1]
beta = parm[2]
sigu = parm[3]
mu = omega/(1-beta)
sig2 = (sigu*sigu)/(1-beta*beta)
#
E.sigma = c(sqrt(2/pi) * exp(mu/2 + sig2/8),
exp(mu + sig2/2),
2 * sqrt(2/pi) * exp(3*mu/2 + 9*sig2/8),
3 * exp(2*mu + 2*sig2))
E.sigma.c = c(2/pi * exp(2*(mu/2 + sig2/8) + beta^(1:10) *
sig2/4),
exp(2*(mu + sig2/2) + 4 * beta^(1:10) * sig2/4))
#
t(t(data) - c(E.sigma, E.sigma.c))
}
```

```
start.vals = c(0,0.5,0.5)
names(start.vals) = c("omega","beta","sigu")
sv.fit.1 = GMM(start.vals, sv.moments,
          method="iterative",
          ts=T,
          var.hac =
          var.hac.control(window="bartlett"),
          data=sv.data)
```

Note: software default for bandwidth: q(t) = 4(n/100)^(1/4)

ω_0=-0.736, β_0 = 0.90, $\sigma_{0,u}$ = 0.363

```
Coefficients:
        Value Std.Error  t value Pr(>|t|)
omega  -0.4628    0.3731   -1.2405   0.2151
 beta   0.9378    0.0502   18.6959   0.0000
 sigu   0.2233    0.0978    2.2841   0.0226

Optimization Info:
Number of Iterative Steps: 7

 Test of Overidentification:
  J-stat Df P.value
  18.88  21 0.5931
```



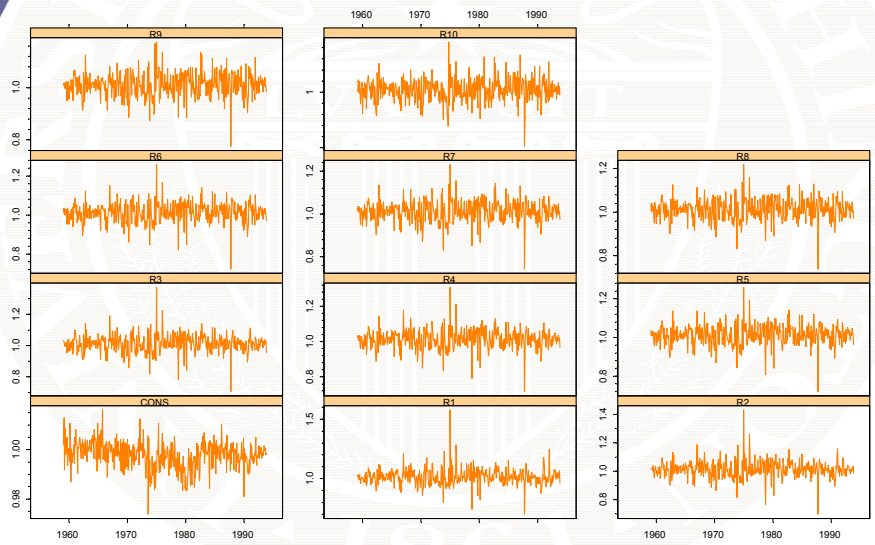
Iterated efficient GMM estimation of log-normal SV model for daily S&P 500 returns

```
Coefficients:
        Value Std.Error  t value Pr(>|t|)
omega  -0.1541   0.1758   -0.8767   0.3807
 beta   0.9838   0.0184   53.3692   0.0000
 sigu   0.1548   0.0908    1.7041   0.0884

Test of Overidentification:
 J-stat Df P.value
 39.93  21 0.0076

Optimization Info:
Number of Iterative Steps: 9
```

SV mode rejected by GMM

```
> pricing.ts@title
[1] "Asset pricing data from Marno
Verbeek's A Guide to Modern Econometrics“

> pricing.ts@documentation
[1] "Monthly observations from February
1959 to November 1993 (T=418)"
[2] "on the returns on 10 size-based
portfolios, the riskfree rate and"
[3] "and consumption growth"

> colIds(pricing.ts)
 [1] "CONS" "R1"   "R2"   "R3"   "R4"
"R5"   "R6"   "R7"   "R8"   "R9"   "R10"
"RF"
```

Monthly Consumption growth and Gross Returns



Sample Statistics on Consumption growth and Gross Returns

Sample Moments:

	mean	std	skewness	kurtosis
CONS	0.9977	0.005565	-0.44136	3.971
R1	1.0139	0.070121	1.22797	14.480
R2	1.0128	0.061475	0.42222	10.248
R3	1.0121	0.058338	0.08018	8.559
R4	1.0121	0.055595	-0.17291	7.072
R5	1.0112	0.053274	-0.34109	6.458
R6	1.0115	0.051786	-0.36335	6.545
R7	1.0108	0.050190	-0.32176	5.787
R8	1.0110	0.048885	-0.30743	5.887
R9	1.0100	0.046201	-0.21773	4.913
R10	1.0085	0.041246	-0.16767	5.186

Number of Observations: 418

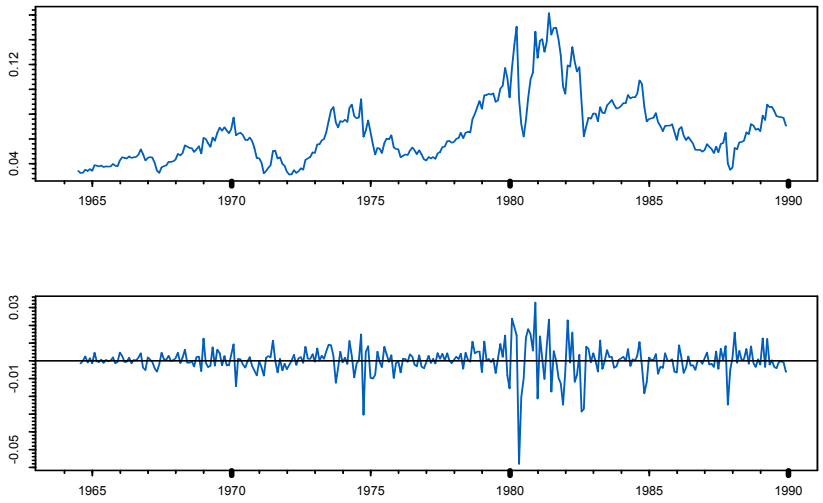
```
# moment function for basic Euler equation
# asset pricing model with power utility
# and one risky asset
euler1.moments <- function(parm,data=NULL) {
# parm = (beta,gamma)
# data = (1+R(t+1),C(t+1)/C(t),1,C(t)/C(t-1),
#         C(t-1)/C(t-2),R(t),R(t-1),...)
ncol = numCols(data)
euler = data[,1]*parm[1]*(data[,2]^(-parm[2]))
        - 1
as.matrix(rep(euler,(ncol-2))*data[,3:ncol])
}
```

Iterated Efficient GMM Estimation

```
Coefficients:
       Value Std.Error t value Pr(>|t|)
 beta  0.8314  0.1170    7.1049  0.0000
alpha 57.4020 34.3021    1.6734  0.0950

Test of Overidentification:
 J-stat Df P.value
 5.657  9  0.7737

Optimization Info:
Number of Iterative Steps: 9
```

Unbelievably large degree of risk aversion

Monthly 30-Day T-Bill Data – Annual Rate




```
# function to compute CKLS moments
# note: dt defines the discretization step. For example,
#       if r(t) is annualized rate sampled monthly,
#       set dt = 1/12
#       if r(t) is monthly rate sampled monthly, set dt = 1
ckls.moments <- function(parm,data=NULL,dt=1/12) {
# parm = (alpha,beta,sigma,gamma)'
# data = [r(t+dt)-r(t),r(t)]
# dt = discretization step
e.hat = as.vector(data[,1] - (parm[1] +
                  parm[2]*data[,2])*dt)
m2 = e.hat*as.vector(data[,2])
m3 = e.hat^2 -
   dt*parm[3]*parm[3]*(as.vector(data[,2])^(2*parm[4]))
m4 = m3*data[,2]
cbind(e.hat,m2,m3,m4)
}
```

```
# compute data for GMM estimation
data.ckls.ts =
seriesMerge(diff(ckls.ts),tslag(ckls.ts))
colIds(data.ckls.ts)[1] = "RF.diff"
data.ckls =
as.matrix(seriesData(data.ckls.ts))

# GMM estimation of just identified ckls model
start.vals = c(0.06,-0.5,1,1)
names(start.vals) =
c("alpha","beta","sigma","gamma")
gmm.ckls = GMM(start.vals,ckls.moments,ts=T,
               data=data.ckls,dt=1/12)
```

```
Coefficients:
       Value Std.Error t value Pr(>|t|)
alpha  0.0419  0.0193    2.1710  0.0307
 beta -0.6077  0.3454   -1.7592  0.0796
sigma  1.3337  1.0004    1.3331  0.1835
gamma  1.5081  0.2900    5.2010  0.0000

Test of Overidentification:
model is just-identified


Optimization Info:
 Number of Iterations: 14
 Convergence: absolute function convergence
```