library(tseries)

library(rstan)

library(loo)

library(tswge)

data(nino)

options(scipen=999)

y = as.numeric(nino3.4)

D = list(y=y,T=598)

#

# MLE fit, detect best fit with AR and MA up to five lags

#

A=aic.wge(y, p = 0:5, q = 0:5, type = "aic")

B =arma(y, order = c(1, 1), lag = NULL, coef = NULL,include.intercept = T)

# parameterises intercept differently to arma

C = arima(y, order = c(4, 0, 1))

**#**

**# ARMA(1,1)**

**#**

ARMA11.stan <- "

data { int<lower=1> T; // length of series

real y[T];

}

parameters { real phi0; // intercept

real phi1; // AR coeff

real gamma1; // moving avg coeff

real kappa; // composite preseries term

real<lower=0> sigma; // residual sd

}

transformed parameters { real y\_fit[T];

real error[T];

for (t in 1:T) {error[t] =y[t]-y\_fit[t]; }

// kappa is composite pre-series term, for t=1

y\_fit[1] = phi0 + kappa;

// fitted model t>1

for (t in 2:T) {y\_fit[t] = phi0 + phi1 \* y[t-1] + gamma1 \* (y[t-1]-y\_fit[t-1]); }

}

model { real eps[T];

phi0 ~ normal(0,10);

kappa ~ normal(0,10);

phi1 ~ normal(0,2);

gamma1 ~ normal(0,2);

sigma ~ cauchy(0,5);

// period 1 model

eps[1] = y[1] - phi0 - kappa;

eps[1] ~ normal(0,sigma);

// periods t>1

for (t in 2:T) { eps[t] = y[t] - (phi0 + phi1 \* y[t-1] + gamma1 \* eps[t-1]);

eps[t] ~ normal(0,sigma); }}

generated quantities {

vector[T] log\_lik;

for (t in 1:T ) { log\_lik[t] = normal\_lpdf(y[t] |y\_fit[t], sigma); } }

"

**# Initial Values and Estimation**

INI <- list(list(phi0=5,gamma1=0.3,phi1=0.9,sigma=0.5,kappa=0),

list(phi0=10,gamma1=0.5,phi1=0.8,sigma=0.7,kappa=0.5))

sm <- stan\_model(model\_code=ARMA11.stan)

fit1 <- sampling(sm,data =D,pars =c("phi0","phi1","gamma1","y\_fit","log\_lik"),iter = 10000,warmup=500,chains = 2,seed= 12345,init=INI)

print(fit1)

**# Fit**

loo(as.matrix(fit1,pars="log\_lik"))

**#**

**# ARMA(1,1) Conditioning on y1**

**#**

ARMA11Cond.stan <- "

data { int<lower=1> T; // length of series

real y[T];

}

parameters { real phi0; // intercept

real phi1; // autoregression coeff

real gamma1; // moving avg coeff

real<lower=0> sigma; // residual sd

}

transformed parameters { real y\_fit[T];

y\_fit[1] = y[1];

for (t in 2:T) {y\_fit[t] = phi0 + phi1 \* y[t-1] + gamma1 \* (y[t-1]-y\_fit[t-1]); }

}

model { real eps[T];

phi0 ~ normal(0,10);

phi1 ~ normal(0,2);

gamma1 ~ normal(0,2);

sigma ~ cauchy(0,5);

eps[1] = 0;

for (t in 2:T) { eps[t] = y[t] - (phi0 + phi1 \* y[t-1] + gamma1 \* eps[t-1]);

eps[t] ~ normal(0,sigma); }}

"

**# Initial Values and Estimation**

INI <- list(list(phi0=5,gamma1=0.3,phi1=0.9,sigma=0.5),

list(phi0=10,gamma1=0.5,phi1=0.8,sigma=0.7))

sm <- stan\_model(model\_code=ARMA11Cond.stan)

fit2 <- sampling(sm,data =D,pars =c("phi0","phi1","gamma1","y\_fit"),iter = 20000,warmup=500,chains = 2,seed= 12345,init=INI)

print(fit2)

**#**

**# ARMA(4,1)**

**#**

ARMA41.stan <- "

data { int<lower=1> T; // length of series

real y[T];

}

parameters { real phi0; // intercept

real phi[4]; // autoregression coeff

real gamma1; // moving avg coeff

real kappa[4];

real<lower=0> sigma; // residual sd

}

transformed parameters { real y\_fit[T];

real error[T];

for (t in 1:T) {error[t] =y[t]-y\_fit[t]; }

// kappa[1] is composite parameter for combined lag effects of y[0],y[-1], etc, and

// gamma1\*(y[0]-y\_fit[0])

y\_fit[1] = phi0 + kappa[1];

y\_fit[2] = phi0 +phi[1] \* y[1]+ gamma1\*(y[1]-y\_fit[1]) +kappa[2];

y\_fit[3] = phi0 +phi[1] \* y[2] +phi[2] \* y[1]+ gamma1\*(y[2]-y\_fit[2]) +kappa[3];

y\_fit[4] = phi0 +phi[1] \* y[3] +phi[2] \* y[2] +phi[3] \* y[1]+ gamma1\*(y[3]-y\_fit[3]) +kappa[4];

for(t in 5:T){y\_fit[t]=phi0+phi[1]\*y[t-1]+phi[2]\*y[t-2]+phi[3]\*y[t-3]+phi[4]\*y[t-4]+

gamma1\*(y[t-1]-y\_fit[t-1]);}

}

model { real eps[T];

phi0 ~ normal(0,10);

kappa ~ normal(0,10);

phi ~ normal(0,2);

gamma1 ~ normal(0,2);

sigma ~ cauchy(0,5);

eps[1] = y[1] - phi0 - kappa[1];

eps[1] ~ normal(0,sigma);

eps[2] = y[2]-( phi0 +phi[1] \* y[1]+ gamma1\*(y[1]-y\_fit[1]) +kappa[2]);

eps[2] ~ normal(0,sigma);

eps[3] = y[3]-( phi0 +phi[1] \* y[2] +phi[2] \* y[1]+ gamma1\*(y[2]-y\_fit[2]) +kappa[3]);

eps[3] ~ normal(0,sigma);

eps[4] =y[4]-( phi0 +phi[1] \* y[3] +phi[2] \* y[2] +phi[3] \* y[1]+ gamma1\*(y[3]-y\_fit[3]) +kappa[4]);

eps[4] ~ normal(0,sigma);

for (t in 5:T) { eps[t] = y[t] -(phi0+phi[1]\*y[t-1]+phi[2]\*y[t-2]+phi[3]\*y[t-3]+phi[4]\*y[t-4]

+gamma1\*eps[t-1]);

eps[t] ~ normal(0,sigma); }}

generated quantities {

vector[T] log\_lik;

for (t in 1:T ) { log\_lik[t] = normal\_lpdf(y[t] |y\_fit[t], sigma); } }

"

**# Initial Values and Estimation**

sm <- stan\_model(model\_code=ARMA41.stan)

INI <- list(list(phi0=8,gamma1=0.8,phi=c(0.3,0.9,-0.4,-0.2),sigma=0.5,kappa=c(-1,1,-1,1.5)),

list(phi0=7,gamma1=0.9,phi=c(0.4,0.8,-0.3,-0.1),sigma=0.7,kappa=c(-2,1.5,-1.5,2)))

fit4 <- sampling(sm,data =D,pars =c("phi0","phi","gamma1","y\_fit","kappa","log\_lik"),iter = 10000,warmup=500,chains = 2,seed= 12345,init=INI)

print(fit4)

**# Fit**

loo(as.matrix(fit4,pars="log\_lik"))