

Homework #2

Time Series Regression

Jude C. Hays
TSCS Analysis
DUE: February 10, 2021

February 3, 2021

In this section, you will analyze the dynamics and determinants of aggregate policy mood (1960Q1-2011Q4). Policy Mood is a time series measure of public support for government programs on the liberal-conservative continuum (Stimson, 1991), with higher values representing a more liberal policy mood. Along with this variable, the dataset includes growth in real GDP (from the BEA) and economic expectations from the University of Michigan's survey of consumer attitudes. More specifically, the latter variable is the aggregate percentage of survey respondents who answer "better off" to the question: do you think that a year from now you (and your family living there) will be better off financially, worse off, or just about the same as now?

1 Time Series Regression

- (a) Develop an argument and hypotheses about the relationships between real GDP growth, economic expectations and policy mood.
- (b) Estimate some static models. Specifically, estimate a static model with Newey-West standard errors and a model with serially correlated disturbances using FGLS. What do these models suggest about the relationships between your independent variables and policy mood?
- (c) Next, estimate several dynamic time series regressions—the autoregressive distributed lag (ADL), finite distributed lag (FDL), and partial adjustment (PA) models. Use diagnostics to justify your specification choices. What are the differences implied by these dynamic models? Which model fits the data best? Which model makes the most sense theoretically?
- (d) Do the static and dynamic models give dramatically different results regarding the (total) effects of the independent variables on policy mood? Explain.
- (e) Overall, do the results support your hypothesis (or hypotheses)?

2 Optional Supplemental Exercises

Start by building a univariate ARIMA model for the policy mood variable. Next, check to see if the Reagan “revolution” is associated with a shift in the steady-state equilibrium in policy liberalism. Finally, explore the relationships between policy mood and real GDP growth and the public’s economic expectations.

2.1 Build an ARIMA Model

- (a) Take a look at the graph of policy mood. Based on your initial (ocular) evaluation, characterize the data generating process (DGP) of your time series. Does the series look more like an autoregressive or moving-average process? Explain.
- (b) Next, examine the ACF and PACF in order to identify a preliminary ARIMA model. Fit the model and evaluate it. Try additional models to see if you can do better. At a minimum, your model evaluations should include fit statistics and residual diagnostics. A rigorous analysis would include a comparison of out-of-sample forecast performances and test for parameter stability.
- (c) Did the analysis support your initial characterization of the DGP? Why (not)?

2.2 Intervention Analysis

- (a) Formulate a hypothesis about an intervention effect, of either a pulse or permanent type, related to the Reagan revolution (or any other important disjuncture in American politics).
- (b) Identify an ARIMA model using either the pre or post-intervention part of your sample, and then estimate this model with your intervention using the full sample.
- (c) Interpret the results in relation to your hypothesis. If there’s an intervention effect, plot this effect.

2.3 Transfer Function Models

- (a) Make a theoretical argument for why the public’s economic expectations might have a causal impact on policy mood.
- (b) Identify an ARMA model for the expectations series.
- (c) Filter the expectations and mood series, and then examine the CCF (or CCVF) to identify the autoregressive structure of the mood series as well as the transfer function.
- (d) Estimate the model and evaluate the disturbances. If they are not white-noise, make the necessary adjustments to your model and reestimate.
- (e) What is the effect, if any, of an innovation in economic expectations on the policy mood?

3 Monte Carlo Exercise

Monte Carlo studies use simulated data to analyze the small-sample properties of alternative techniques for inference under varying experimental conditions. Some questions that Monte Carlo methods can address, among many others, include the following: how does Feasible Generalized Least Squares (FGLS) compare to Generalized Least Squares (GLS) and Ordinary Least Squares (OLS) when the true data generating process includes an AR1 in the disturbances? How well do static models estimate long-run equilibrium relationships when the true data generating process is a dynamic lagged dependent variable model? How well do minimally restricted, reduced-form models recover structural relationships in small samples?

In this exercise, we will evaluate the importance of getting the temporal dynamics right when it comes to inferring systematic relationships among time series variables. Next week, we will cover the classic spurious regressions problem for non-stationary (first-order integrated) variables. It turns out spurious relationships can be a problem with stationary (near-integrated) variables in small samples. Generate samples of 30, 50 and 100 observations using $y_t = .95y_{t-1} + \varepsilon_{y,t}$ and $z_t = .95z_{t-1} + \varepsilon_{z,t}$, regress y and z , and report the 90, 95, and 99% critical values for the t -statistics for 1000 simulations. Do these critical values correspond closely with those from a t -distribution?

Below I have provided some sample R code that implements the Granger and Newbold spurious regression simulations, “Generating the Dickey-Fuller Distribution,” in Enders (pp. 204-206, 3rd edition). This should provide a useful example for you to follow.

Sample R Code

```
#Clear
rm(list=ls())
#This code will perform the Granger and Newbold spurious regression
simulations, showing that the rate of false positives is much higher
#than suggested by critical values from a t-distribution. First, a
#function is created to run each trial of the simulation. In each
#trial, two independent random walks are generated, and then one is
#regressed on the other. Second, the function is run a large number
#of times (1000 in the code currently) and the results are collected
#and summarized.

# the runmc() function generates a dataset and runs each analysis
# the parameter "trial" keeps track of which time we're calling runmc()
runmc = function(trial) {
  # the result vector will hold the estimates temporarily
  result = matrix(0,1)

  # generate y as a random walk
  y<-rep(NA,100)
  y[1]<- rnorm(1)
```

```

    for (i in 2:100) y[i]<-y[i-1]+rnorm(1)
    #generate z as a random walk
    z<-rep(NA,100)
    z[1]<- rnorm(1)
    for (i in 2:100) z[i]<-z[i-1]+rnorm(1)
    #regress y on z and save the results in "sum"
    result = lm(y~z)
    sum=summary(result)
    #pull out the t-statistic
    tstat = sum$coef[2,3]
    return(c(trial, tstat))
}

#run the experiment 1000 times and collect the results
res2 = as.data.frame(t(sapply(1:1000, runmc)))
#label the columns
names(res2) <- c("trial","tstat")
head(res2)
#summarize the distribution of absolute-value t-statistics
summary(abs(res2[,2]))

```