Time Series

Dr. Sarah Hunter

3/29/2020

The Problem of Time Series Data

Time series data is very common in political science research. Just think of research on polling trends, economic factors, or federal budgets. Any variable that changes over time is a time series variable. Time series data can present problems to our classical regression models. First, using two time series measures can produce the spurious regression problem, which occurs when two variables appear to have a statistical relationship, but are only related because of time trends (see your book for the example of Golf and divorce rates). The second problem time series data presents is a violation of our OLS assumptions: autocorrelation. Autocorrelation occurs when the model's errors are correlated with each other. In more formal notation:

$$cov_{u_i u_j} = 0 \; \forall \; i \neq j$$

where u_i is the error at time i and u_j is the error at time j.

Essentially, autocorrelation occurs when our error at one time is related to the error at the next time. Time series data is particularly vulnerable to this problem due to seasonal/cyclical trends and the nature of a time trend in general. If our prediction at one time is under/over estimated, then we are likely to also under/over estimate the next time point.

Due to both the spurious regression problem and autocorrelation, we have developed new methods to mitigate these issues, which include:

- Lags and Leads
- Differenced Dependent Variables
- Multiple Lags
- Koyck Model

We will go through each of these solutions and show how they are done in R. For this example, we are using data from the textbook (Kellstedt and Whitten). These data are the US presidential approval rates monthly from 1995 to 2005. Below, I set my working directory and load the data into R. However, you will notice that I had to install a new package in order to read this dataset. These data are in the form of a Stata 13 or higher file. With any file like this, the `foreign` library cannot load the dataset. So, we need the `readstata13` library, which functions very similarly as the `foreign` library. The code below shows how to use this new package.

```
setwd("/Users/sarahhunter/Desktop/Data")

#install.packages("readstata13")

library(readstata13)
presdata<-read.dta13("presap9505.dta")

#summary stats

summary(presdata)
```

```
##      presap        year_month
##  Min.   :45.00   Min.   :421.0
##  1st Qu.:54.33   1st Qu.:450.5
##  Median :59.00   Median :480.0
##  Mean   :59.59   Mean   :480.0
##  3rd Qu.:63.15   3rd Qu.:509.5
##  Max.   :88.00   Max.   :539.0
```

Lags and Leads

The first, and easiest, way to address the problem of time series data is to use lags and/or leads. Lagged values are the value of a variable at the time point before the current time point. For example, if we think presidential approval and GDP growth are related, and we think that last year's GDP growth leads to this year's presidential approval rating, we could lag GDP growth. In this case, 2015's GDP growth would be a predictor (aka an independent variable) in 2016's presidential approval rating (aka the dependent variable). In a model, this would look like:

$$Y_t = \alpha + \beta * X_{t-1} + u_t$$

Leads, on the other hand, use the same concept, just reversed: Lead values are the value of a variable after the current time point. In a model, this would look like:

$$Y_{t+1} = \alpha + \beta * X_t + u_t$$

You can create Lags and Leads in R using the `dplyr` package in R. This package contains many useful commands for data manipulation, but here, I just want to introduce you to two: `lag` and `lead`. The code below shows how to use these two commands. Both follow the same structure: `lag(x, n)`. In this structure, x is the variable that you need to lag and n is the number of time periods you would like to lag.

```
#install.packages("dplyr")

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
presdata$lagged_ap<-lag(presdata$presap, 1)

presdata$lead_ap<-lead(presdata$presap, 1)

head(presdata)
```

```
##   presap year_month lagged_ap lead_ap
## 1   45.5        421        NA      45
## 2   45.0        422      45.5      48
## 3   48.0        423      45.0      51
## 4   51.0        424      48.0      47
## 5   47.0        425      51.0      47
## 6   47.0        426      47.0      46
```

Differenced Dependent Variables

A differenced dependent variable is simple when we make the value of the dependent variable the change from one time point to the next. Instead of modeling the absolute value of the dependent variable, the differenced dependent variable approach would model the *change* in the dependent variable. This tool is particularly useful in avoiding the spurious regression problem.

Mathematically, we calculate the differenced DV as:

$$\Delta Y_t = Y_t - Y_{t-1}$$

where ΔY_t denotes the change in Y at time i, Y_t denotes the value of Y at time t, and Y_{t-1} is the value of Y at time $t-1$.

In a model, this would look like:

$$\Delta Y_t = \alpha + \beta * X_{t-1} + u_t$$

Creating a differenced DV in R is quite simple, once you have your lagged variable. You create the lag using `dplyr` and then take the difference using R's math functions. Below is an example of creating a differenced DV:

```
presdata$delta_ap<-presdata$presap - presdata$lagged_ap

head(presdata)
```

```
##   presap year_month lagged_ap lead_ap delta_ap
## 1   45.5        421        NA      45       NA
## 2   45.0        422      45.5      48     -0.5
## 3   48.0        423      45.0      51      3.0
## 4   51.0        424      48.0      47      3.0
## 5   47.0        425      51.0      47     -4.0
## 6   47.0        426      47.0      46      0.0
```

Multiple Lags

What happens is you think that one variable's effect happens over multiple time points? In other words, it is not just the last time period's value that matters, but the time before that, etc. In this case, you would use *multiple lags* in a model to better account for that. In this case, you would use multiple lags of the *independent variable* to model the dependent variable.

In a basic linear model, this would look like:

$$Y_t = \alpha + \beta_1 * X_t + \beta_2 * X_{t-1} + \beta_3 * X_{t-2} + \beta_4 * X_{t-3} + u_t$$

In this case, the effect of X on Y is $\Sigma\beta_i$ which means that we add all values of β in the the model.

Creating multiple lags in R is simply an extension of making one lag using the `dplyr` library. Below is an example of making lags for 2 and 3 time periods.

```
#lag for t-2

presdata$lag2_presap<-lag(presdata$presap, 2)

#lag for t-3

presdata$lag3_presap<-lag(presdata$presap, 3)

head(presdata)
```

```
##   presap year_month lagged_ap lead_ap delta_ap lag2_presap lag3_presap
## 1   45.5        421        NA      45       NA          NA          NA
## 2   45.0        422      45.5      48     -0.5          NA          NA
## 3   48.0        423      45.0      51      3.0        45.5          NA
## 4   51.0        424      48.0      47      3.0        45.0        45.5
## 5   47.0        425      51.0      47     -4.0        48.0        45.0
## 6   47.0        426      47.0      46      0.0        51.0        48.0
```

Koyck Model

While the effect of X on Y can last over multiple time points, that impact might decay as time progresses, meaning that the impact of X on y at time t might be much higher than the impact at t-1, which is higher than t-3, and so on. In this case, a Koyck Model would be the appropriate approach.

Another name for the Koyck Model is a Lagged Dependent Variable Model. As the name implies, a lagged DV model includes the lagged value of the dependent variable in a linear model to account for the decaying effect of the independent variable on the dependent variable. In a linear model, this would look like:

$$Y_t = \alpha + \lambda * Y_{t-1} + \beta_0 * X_t + u_t$$

With this model, the effect of X on Y (β) is calculated by $\beta = \frac{\beta_0}{1-\lambda}$.

To estimate this model in R, you can make a lagged DV using the same `dplyr` command (`lag`) as before, then including that lagged dependent variable as a predictor (independent variable) in the model.

Time Series Cross Sectional Data

Our strict definition of a time series measure is on that contains variation over time in one spatial unit, such as presidential approval rating in the US for each month. However, much of our data are **time series cross sectional** (TSCS) data, which is variation through time at multiple spatial units. For example, while the US presidential approval rating for each month is a time series measure, if we instead gathered the presidential approval rating for each state each month, that would be time series cross sectional data.

Dealing with TSCS can present several challenges. For further reading on this subject, I refer you to the experts. The following article is the most influential in dealing with TSCS data in political science:

Beck, N. and J. N. Katz (1995) What to do (and Not to Do) with Time-Series Cross-Section Data. American Political Science Review 89(3): 634-647 DOI: https://doi.org/10.2307/2082979

Making Lags for TSCS Data

The previous command shown above to create lag variables for pure time series data. In order to create a lagged variable for TSCS data, you must also indicate a grouping variable, or the spacial unit identifying variable in the data. For example, if your TSCS data's obersvations were state years, your grouping variable would be state. You can use `dplyr` again to create a lag variable for TSCS data by:

```
#Stata 13 file, so need a new package

library(readstata13)

#Can use full file path instead of setting wd
qog<-read.dta13("/Users/sarahhunter/Desktop/qog_cow_donors.dta")

#Creates a new dataset with lag
```

```r
#need dplyr for the lag function
library(dplyr)

newqog<-qog %>% group_by(donor) %>% mutate(lgdpgr= lag(imf_gdpgr))

head(newqog)
```

```
## # A tibble: 6 x 5
## # Groups:   donor [1]
##   cname        year donor imf_gdpgr lgdpgr
##   <chr>       <int> <int>     <dbl>  <dbl>
## 1 Afghanistan  2000   700    NA     NA
## 2 Afghanistan  2001   700    NA     NA
## 3 Afghanistan  2002   700    NA     NA
## 4 Afghanistan  2003   700     8.69  NA
## 5 Afghanistan  2004   700     0.671  8.69
## 6 Afghanistan  2005   700    11.8    0.671
```