Multivariate Temporally Correlated Data Analysis

Jackson Curtis

December 2018

1 Introduction

Multivariate regression is a technique for jointly modeling the effect of multiple independent variables on multiple dependent variables. Although the dependent variables could each be modeled separately (and in fact the MLEs would be the same), We lose important information about how the estimates behave jointly when we model them separately. Multivariate regression accounts for the joint behavior of the data and leads to better inferences.

Multivariate regression, like univariate regression, relies on the assumption of independence given the covariates (each row in the dataset is independent of the others). However, many datasets are created by generating one observation in each time period. When data is gathered like this, it is likely that random fluctuations at one time period will carry over, affecting the time periods close to it. Autoregressive models model these perturbations directly, with the assumption that one perturbation will carry over to the next time period and affect the measurements in that time, resulting in correlated errors.

In this project we will describe an AR(1) model that allows for estimation of multivariate regression parameters while accounting for time correlation. We will motivate this with an example using data from the recent midterm elections. We will describe the technical details and challenges of fitting this model in R and show the results.

2 Motivating Example

Fivethirtyeight.com is a data-driven news site that specializes in election forecasting. They produce a forecast for every House and Senate race based on polling, historical, and economic data [2]. These forecasts predict the probability of victory as well as predicted voter share for each candidate. Each forecast is updated daily with new polling information in the months leading up to the election.

After the election, it is often interesting to look at the performance of the forecasts, particularly with an eye for ways that the model might be improved in the future. For this example, we will look only at House races where at least one Democrat and one Republican were running. We will look at three aspects of the forecast and how they vary over time:

- Democrat voter share - The average predicted voter share for the Democratic candidate(s) in each race
- Mean squared error - The accuracy of the forecast, as measured by the mean squared error of the predicted vote share for the Democratic candidate compared to the actual result
- Calibration - A measure of the accuracy of the probabilistic statements (if 20 candidates are given a 40% chance of winning, a well-calibrated model should have around 8 win),

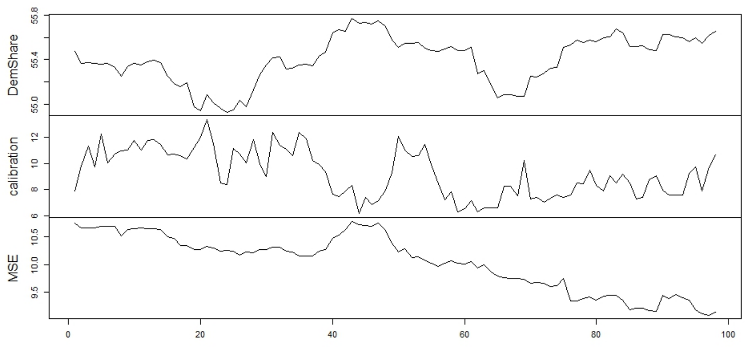

Figure 1: Our measurements calculated from the days predictions, 100 days before the election

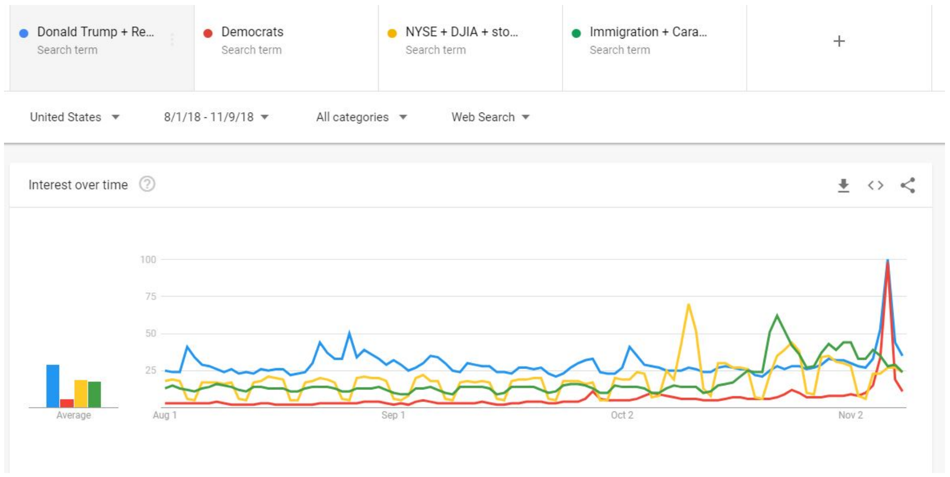

Figure 2: Search interest volume from Google Trends

measured as a goodness of fit statistic between the expected and actual number of victories for candidates in each probability range.

Figure 1 shows each of these measurements for each day in the hundred days before the election. These measurements are functions of the same underlying data, so we do not expect them to behave independently, making them good candidates for a multivariate model. An interesting question is how trends in news coverage could affect each of these measurements. We can use Google search queries (obtained from Google Trends) as a close substitute for news coverage, as much of the search traffic will be driven by news stories. We binned our trend data into four main storylines that we will investigate: Trump/Republicans, Immigrants/Caravan, Stock Market/Economy, and Democrats/Clinton/Obama. Figure 2 shows how these trend over the 100 days proceeding the election. Using our model we will look for relationships between these trends and the measurements we took on the forecasting model. Note, because stock trends have a strong time component (people research the market when it's open on weekdays), we will first regress the market data on weekday/weekend and use the residuals as our covariate.

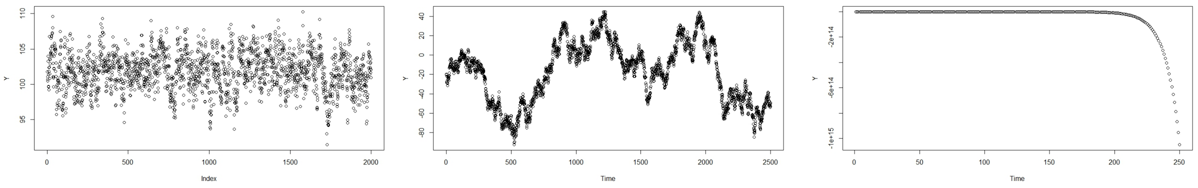

Figure 3: Left: A stationary time series that stays centered on its mean. Middle: A random walk from summing all previous errors. Right: A non-stationary time series where each error is compounded in the next time step.

3 The Model

We will define our AR(1) model in the following way:

$$
\begin{aligned}
y_t &= X_t\beta + \epsilon_t \qquad (1)\\
\epsilon_t &= \Phi\epsilon_{t-1} + \omega_t\\
\omega_t &\sim MVN(\mathbf{0}, \Sigma)
\end{aligned}
$$

In the above equations y_t is a 1x3 vector of observations for day t. X_t is a 6x1 row vector with an intercept, a linear time trend, and our four covariates. β is a 6x3 matrix of regression coefficients. Σ is 3x3 diagonal matrix of independent error terms for each of our three response variables. ϵ_t is the sum of the influence of previous errors with the new error introduced at day t.

Φ is a 3x3 matrix that controls how the last time step's error affects the current error. The effects of this matrix vary dramatically based on the properties of the elements. For example, if each row of the matrix has a sum less than one, the effect of ϵ_1 will go to 0 in the limit as t increases. Thus, the most recent errors carry the biggest weights, and the older errors fade in importance. This ensures that the overall mean error stays close to zero. This property is referred to stationarity because the mean does not change except through the covariates. If Φ was set to the identity matrix, the error at time t would be the sum of the ωs for all time $<= t$. This exhibits the properties of a random walk where the response is equally likely to drift in any direction. Finally, if the sum of the rows is greater than one, the previous errors get exacerbated as time goes on. If data is generated in this model, each y will eventually drift to positive or negative infinity as the previous errors compound on each other. Examples of simulated data using these three properties are shown in Plot 3. A brief simulation study will show how effective our estimation method is in each of these three scenarios.

4 Model Estimation in R

Despite the relative simplicity in writing out the model, finding ways to estimate the model in R proved challenging. Most R packages (including the vars package for vector autoregression) specify time series of the form:

$$y_t = \Phi y_{t-1} + X_t\beta + \epsilon_t \qquad (2)$$

While these provide adequate models for most time series data, they don't allow good estimation of our quantities of interest (the βs) because the βs from the previous time step are multiplied by Φ and added again into the equation. We would like our method to only be autoregressive on the error terms, not the covariates.

	Coverage of 95% Interval		
Parameter	Stationary	Identity	Non-Stationary
Small Error	1	12	6
Medium Error	2	15	11
Large Error	48	30	52
Φ Elements	83.6	71.2	81.2
Intercept Terms	54.6	14	7.7
Linear Trend Terms	70.7	37	30.3
Other Covariates	94.7	96.9	96.9

Table 1: The coverage achieved through 100 simulations (if multiple parameters are grouped the coverage is averaged over all parameters)

The MARSS package in R provided a way to estimate the model specified in Equation 1 [1]. MARSS stands for Multivariate Auto-Regressive(1) State Space, and although its primary focus is the estimation of state space vectors, carefully specifying constraints allows us to estimate the parameters we would like. The general framework of MARSS is to provide an extremely overparameterized model, and then provide the user an easy way to set constraints on the model. Many different time series analysis can be written as constrained versions of the overparameterized model, including the one we would like to fit. Model estimation is then performed using the EM algorithm or BFGS optimization. A benefit of using the EM algorithm is that the package will seamlessly handle missing data; however, attempting to estimate our model using the EM algorithm resulted in an incomprehensible error message, so we utilized BFGS optimization. Confidence intervals are calculated from the asymptotic distribution of MLEs. Bootstrapped intervals are also supported, but do not run in a reasonable amount of time.

5 Simulation

We will test the MARSS estimation technique when the data is generated with the three different scenarios for Φ. We will look at true coverage of the 95% confidence intervals. Each data set will have 100 observations to be close to our motivating example. Because BFGS estimation is quite slow, we will have a small sample of 100 simulations for each scenario.

Details of the parameters used to generate the simulated data are in the appendix. Table 1 shows the resulting coverage for our simulation. By far the worst coverage is seen in estimating the error variances. The MARSS package authors note that "maximum-likelihood estimates of the variance in MARSS models are fundamentally biased...the bias does not go to zero as sample size increases." [1] They also note that this bias is worse when the variance is close to zero, which is what we see also in the simulation.

The intercept and linear trend coefficients do not achieve nominal coverage, especially when the data is not stationary. However, in the stationary case all other parameters in β do seem to have nominal coverage and perhaps even conservative coverage in the non-stationary cases. This gives us more confidence in answering the questions of interest regarding our parameters. Despite poor estimation of everything else in the model, the confidence intervals of our parameters of interest seem to be performing as designed. Surprisingly, while the intercept and linear terms get worse as you increase the non-stationarity, the estimation of Φ does not seem to change.

6 Election Model Results

Term	$\hat{\beta}_{MLE}$	95% CI	
(DemShare,Intercept)	55.543	(55.369, 55.717)	*
(calibration,Intercept)	9.094	(7.016, 11.171)	*
(MSE,Intercept)	10.709	(10.511, 10.906)	*
(DemShare,Time)	-0.010	(-0.044, 0.024)	
(calibration,Time)	-0.039	(-0.093, 0.014)	
(MSE,Time)	-0.015	(-0.022, -0.007)	*
(DemShare,Stocks)	0.001	(-0.001, 0.002)	
(calibration,Stocks)	-0.012	(-0.041, 0. 017)	
(MSE,Stocks)	-0.002	(-0.004, -0.0002)	*
(DemShare,Democrats)	0.000	(-0.002, 0.003)	
(calibration,Democrats)	-0.015	(-0.060, 0.031)	
(MSE,Democrats)	0.001	(-0.002, 0.005)	
(DemShare,Immigration)	-0.001	(-0.003, 0.001)	
(calibration,Immigration)	0.018	(-0.012, 0.048)	
(MSE,Immigration)	-0.002	(-0.004, 0.001)	
(DemShare,Trump)	-0.001	(-0.003, 0.001)	
(calibration,Trump)	0.038	(-0.002, 0.078)	
(MSE,Trump)	0.002	(-0.001, 0.005)	

Table 2: Coefficient estimates produced by MARSS

We fit the MARSS model to our election forecasting data. We constrain the error matrix Σ to be a diagonal matrix with separate variance terms down the diagonal. We allow β and Φ to be completely unconstrained. All other terms in the overparameterized model were set either to the identity or zero so they would not affect the results. From our data, a reasonable hypothesis would be that the news trends would have a delayed effect on the forecast because it would take several days for the news to influence the polls and the polls to be released. Therefore we considered models that lagged the values of the news 0, 1, 2, and 5 days. We then used AIC to choose the best performing model of the four. Surprisingly, it selected the zero day lag, so we do not model any delay in the news trends.

Unfortunately, as can be seen in Table 2, there is no obvious relationship between the news trends and the model performance measures. Unsurprisingly, of the two tests that showed statistical significance at the 0.05 level, one was the time trend on the MSE. This matches what we would expect from Figure 1 which shows that the mean squared error of the predictions is clearly going down over time. The other significant predictor was that when interest in stocks went up, MSE went down. However because we tested 18 variables at a .05 significance level, we would avoid putting too much emphasis on that, as it is quite likely that at least one spurious variable would show as significant.

Table 3 shows that our model estimated the Φ matrix to be right on the border of stationary vs. non-stationary. A stationary model may be overly restrictive for this dataset because of how much we see the data move in a cyclic pattern without covariates to explain the movement. We might be surprised to see a large, negative loading of the MSE error on the calibration error (-1.26), however, these need to be interpreted with regards to the scale of the errors. The calibration errors were over 200 times bigger than the MSE errors (1.2 to .006), so the error contribution from MSE would amount to little more than a rounding error in the calibration term.

	DemShare	Calibration	MSE
DemShare	1.02	0.00	-0.08
Calibration	0.24	0.55	-1.26
MSE	0.00	-0.01	0.89

Table 3: Loading Matrix for the autoregressive errors (Φ_{MLE})

	DemShare	Calibration	MSE
DemShare	0.0049	0.00	0.00
Calibration	0.00	1.2153	0.00
MSE	0.00	0.00	0.0065

Table 4: Σ_{MLE}

7 Conclusion

The MARSS model allowed us to perform multiple linear regression with autocorrelated errors. Although estimating the parameters was not always stable and making inferences was challenging, the model was able to provide good confidence intervals on our simulated datasets for our parameters of interest. We applied the MARSS model to our election data, but failed to find significant relationships between our covariates and dependent variables.

On the application side, future work could focus on better ways to quantify what's happening in the political news world into actionable data. More complex metrics than simple search volume might yield better insights. Modeling these in relation to the polling data directly would also be a more direct link if we wanted to understand how the news affected the elections.

On the theory side more work could be devoted to comparing multiple ways to perform the analysis. Comparing the coverage and confidence interval size of our multivariate AR(1) model with a standard multivariate regression would be useful. Important work still needs to be done on making the MARSS model more conducive to inference. Functions that simplify and speed up the fitting of these AR(1) regression models would be very beneficial to the R community.

8 Appendix

The following inputs were used to generate the data used in the simulation study:

$$\beta = \begin{bmatrix} 0 & 5 & 10 \\ .1 & 0 & -.01 \\ 1 & 0 & 0 \\ .5 & .5 & .5 \\ 0 & .1 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 8 \end{bmatrix}$$

For the stationary model:

$$\Phi = \begin{bmatrix} .6 & 0 & .1 \\ .2 & .5 & .2 \\ 0 & 0 & .8 \end{bmatrix}$$

For the random walk model:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For the non-stationary model:

$$\Phi = \begin{bmatrix} 1.01 & 0 & 0 \\ .05 & .9 & .06 \\ 0 & .5 & .51 \end{bmatrix}$$

It should be noted that the non-stationary model needed rows to sum to something very close to one, otherwise the errors would compound so much that the data generated would be close to positive or negative infinity by t=100, which resulted in a lack of convergence for the MARSS algorithm.

In all cases, the X-matrix was a constant intercept, a time trend term (going from 1 to 100) and 4 columns of random numbers drawn uniformly between -2 and 2.

References

[1] Holmes E, Ward EJ, Scheuerell MD (2018) Analysis of multivariate timeseries using the MARSS package. URL https://cran.r-project.org/web/packages/MARSS/vignettes/UserGuide.pdf, r package version 1.8-3

[2] Silver N (2018) Forecasting the race for the house. URL https://projects.fivethirtyeight.com/2018-midterm-election-forecast/house/