Making ends meet: Projecting demand for class seats at a major American university

Nyssa Bulkes

Springboard Data Science Career Track

Capstone #2 Report

1. Introduction

One of the biggest challenges facing university administrators and department heads is meeting the needs of students. The common story involves students applying for college, getting in, and pursuing the degree of their dreams. As a business, however, other facets play a role in determining the success of students and the university, as a whole. Namely, if students are quick to graduate and be successful in pursuing a degree, that helps with the reputation of the university. The opposite is also true, where if students struggle to complete their degree in a timely manner, this can also reflect poorly on the institution. University operations are funded by student tuition dollars, and as a business, student success equals business success. If we carry the business analogy forward, this suggests that students are receiving a service or product offered by the university—classes and education, to name a couple. If a student is able to secure space in the classes they need to earn their degree of choice, that university can be said to be catering to its clientele.

However, actually providing those seats in a balanced, informed manner requires calculation. For example, imagine a scenario where a university offers 10 sections of a course for 100 students who want to take it. While that results in a rather desirable faculty-to-student ratio of 1:10, it raises the question of whether the university could pay that instructor the same amount of money and increase the seats in each class to 15 or even 20 students per class. Economically, if the university is going to pay the instructor a set rate, it makes more business sense to fit as many students in the class as possible without compromising quality of education and, at the same time, not spread its instructional resources too thin.

To project how many seats are needed in a course, even more factors must be accounted for. For example, is the class an upper- or level-level course? Naturally, upper-level classes may have a lower cap on the maximum number of students allowed in a section due to class format. For which degrees is the class required? If the course is part of a general education plan, it may have a larger pool of potential students that may want to take it and it may even need to be offered more than once an academic year, every semester, for example. Semester types, too, present other challenges for enrollment. For example, a pattern of enrollment for a fall semester will very likely—if not certainly—differ from that for a summer semester.

In the current project, we'll explore the challenges wrought by seat projection from the perspective of a large American university. I will demonstrate how different factors—i.e. semester type—modulate the enrollment patterns expected for a given course and, finally, use data-science techniques to show how these patterns can be computationally modelled.

1.1.1 The scope

About two-thirds of the way through my Springboard Data Science Career Track, I interviewed for and was offered a job as a data analyst for University Analytics and Institutional Research (UAIR) at the University of Arizona. UAIR is responsible for housing all of the UA's data, from student to financial to administrative data. The organization also provides data, as a service, to the university community in the form of ad-hoc requests or larger-scale dashboards and analytics products. The intention of this project was to start to model seat projections in a way that, down the road, could be used to do this very thing at the UA. I will qualify this report by acknowledging that this is merely a starting point and that there are numerous ways in which the following project could be expanded and presented to interested parties within the university. This report describes the first steps toward a tool to this end.

1.1.2 The data

This project incorporates data from the last 30 academic years for ENGL101, a 100-level, general-education course required to complete a bachelor's degree of any kind—bachelor of arts, bachelor of science—at UA. As a general-education course, we can also anticipate that many new students will take this course early in their academic tenure so that they can move on to more specific courses of interest later on that may require ENGL101 as a prerequisite. We also know that compared to other more specialized classes, facilitators of this course will need to plan for capacity, as enrollment in it will be much larger in scale than, for example, enrollment in a 500-level art seminar. As a business, administrators will need to accommodate the volume of students who will need—and not just be interested in taking—this class in order to prevent a bottleneck of students who require the course but cannot proceed to other more advanced courses without having passed it first.

The data used in this project contain no identifiable information. Data were aggregated at the level of class enrollment and cannot be traced back to the level of any particular student. With two separate SQL queries, I pulled class enrollment data for ENGL101 at the level of the semester from UAIR's data warehouse. Within the academic calendar, there are static snapshots taken of the data to capture what they data were "then as of then". What this means is, if we were to go back and reconstruct enrollment, we would only be able to show the data "then as of now". To retain historical record of the data, these snapshots are taken on what are called census days, or regular days in each term that can be compared against each other. The 21st day in a semester is a census, as is the 45th day. The data I pulled for this project were from these census snapshots, not "live" data, as we are looking for a historical record of previous semesters and not the data as it looks today.

From this census data, I included data from fall, summer, and spring semesters--ENGL101 hasn't been offered in winter term in the last 30 years. For this reason, winter-semester data has been excluded in this analysis. Because the data were pulled from the database by me, cleaning was not largely necessary, as I made sure not to pull rows with null values nor did I pull data for courses that were not ENGL101. I did not pull any data from winter semesters as the particular course of interest here is not offered during winter terms. You could say this is one of the advantages of working so closely with the data is that a dataset can be made to uniquely fit the problem at hand.

1.1.3 The application

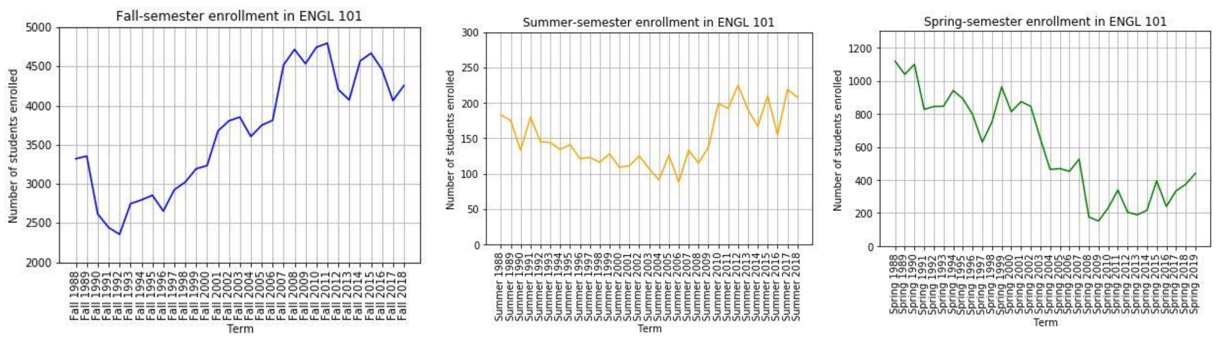
In the future, we as data scientists want to anticipate how such a product as this might be used by its expected customer or user. Knowing that we're dealing with seat projections at a large, public, American university, administrators or leadership might be logical users. When attempting to estimate the number of seats needed in a course per term, I anticipate that a department head or administrator will have the following parameters specified: 1) the term for which the seats will be needed; and 2) the specific course.

With these usability parameters in mind, we will move forward with subsetting the data by semester with the aim of improving model accuracy--e.g. less noise from other semesters--and facilitating the specificity needed to project a specific class that will ultimately be defined by the user. For the purposes of this project, I will stick to modelling ENGL101, both for the reasons that these data offer an appropriate scope for a Capstone 2 and also to economize on the available data provided by the university.

2. Exploratory data analysis

Once I exported the class enrollment counts data into Excel, I read the data into pandas using Python to start data wrangling. Once in Python, I subsetted the data into the different semester types: fall, summer, and spring. As a scientist, when attempting to model data, it is important to acknowledge any a-priori sources of noise that might contaminate or diminish the predictive power of a model. In the current case, we know that fall semesters have different enrollment patterns than summer semesters, for example. By starting our analysis with acknowledging this known source of noise and handling the data in subsets, we can use this to build a more informed approach.

Figure 1. Fall, summer, and spring semester data visualizations



Visualization of the data (Figure 1) shows that, indeed, the three terms have different patterns of enrollment between Fall 1988 and Fall 2018 (shown in all three subplots along the x-axes). Further, the fall semester demonstrates a history of anywhere between 2300 and 4800 students enrolled in a semester. This shows quite a difference when compared with summer terms, where enrollment spans between 90 and 225 students enrolled. Taking world knowledge into account, this is unsurprising, as summers in hot Tucson, Arizona are unattractive time periods for students to want to take in-person classes. Relatedly is the enrollment we see in spring, with numbers anywhere between 150 students and 1100 students. Spring, in fact, shows even more variability with respect to the range of students over the last 30 years. In sum, EDA demonstrates that these three semesters embody different patterns. If we are to successfully use past enrollment to predict future enrollment it would help to isolate these three semesters as datasets and treat them separately.

3. Model selection

Statistically, we know that a simple Ordinary Least Squares regression is inappropriate in this scenario based on the assumptions of OLS. OLS assumes that the predicted value can take on any form, for example, an integer or fraction, a positive or negative value. While enrollment could potentially be 0 in the case of an unsuccessful course, it can never be a negative number. Also, enrollment data is count data and therefore can never be a fraction.

Considering the data being used, time-series analysis is most appropriate, as univariate time-series modelling allows for a potentially underlying structure in the data (i.e. trend, seasonality). There are a few different variations and combinations of time-series approaches to consider. First, a simple AR, or auto-regressive, model, could be of value, and is defined as:

$$X_t = \delta + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-1} + A_t$$

Autoregression is a linear approach where the current value in a series is regressed against the series' previous value or values. *X* represents the time series, with t indicating particular values in the series; *A* represents the white noise; and δ represents the process mean. The order of the AR model is represented by *p*.

3.1 Assessing model assumptions

To proceed with this approach, *stationarity* must be assessed. Stationarity is the extent to which a stochastic process's distribution shifts over time. To assess this, each subset of the data was bisected, and the means and variances were calculated for both pieces. If a dataset demonstrates stationarity, the means and variances should be similar.

Table 1. Stationarity testing results

	Subset 1	Subset 2

Fall data	Mean = 2999.267
	Variance = 182566.067
		Mean = 4289.188
		Variance = 153229.763

Summer data	Mean = 137.867
	Variance = 580.695
		Mean = 160.125
		Variance = 2214.783

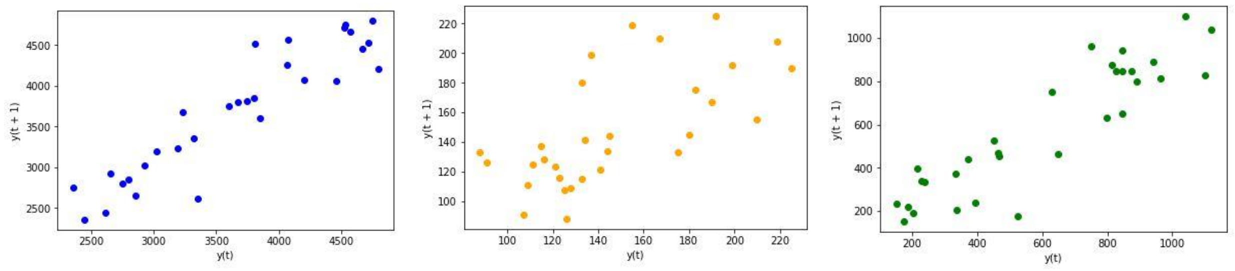
Spring data	Mean = 870.688
	Variance = 19395.429
		Mean = 325.188
		Variance = 15342.696

The test demonstrates these data are far from stationary. The greatest variance is between the two subsets of the fall dataset, and this is unsurprising. A remedy we can use to make the data stationary is *differencing*, a step we will return to below when discussing the ARIMA model. We can employ automated methods to select an optimal instantiation of an ARIMA, but before we consider a more complex approach, it will be instructive to see how a very simple model performs.

For a simple autoregression, we need to check that the underlying assumptions are met, for example, that the data are normally distributed. To show this, I conducted a Shapiro-Wilks test to investigate this assumption in the fall data, the largest of the data subsets. With at test statistic of 0.940 and p-value of 0.083, we see that while the data is trending toward significance, it ultimately does not pass our threshold of 0.05 to be considered non-normally distributed, and we can proceed assuming that the fall data resembles a Gaussian distribution.

Further, sometimes, when using time-series analyses, because of the nature of the data—i.e. one data point is related in some way to the preceding or following observation—we see evidence of autocorrelation. Visually, this would manifest as datapoints appearing correlated to one another, exhibiting a clear visual trend. We can investigate this in the semester-level data by constructing autocorrelation plots; these plots are included below, with fall plotted in blue, summer in yellow, and spring in green (Figure 2).

Figure 2. Autocorrelation plots



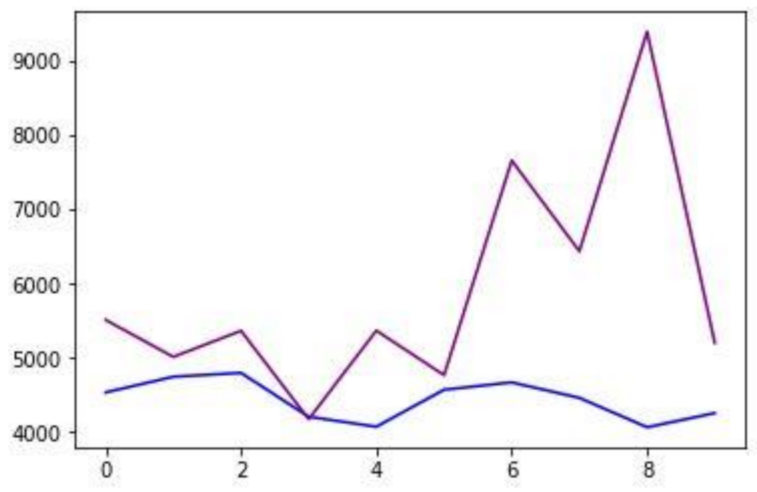
Visualizations demonstrate tangential visual evidence of correlation. While the datapoints show some relationship (i.e. trend), they are not strongly correlated. Further, the patterns in these data are corroborated by the three line graphs plotted of them (review Figure 1). As these data are understood to not be random and visualization shows they are not strongly or immediately correlated, we will proceed.

3.2 Model training

3.2.1 Autoregression

Knowing that there are 31 observations in the fall dataset (and, similarly, 31 observations in the summer and 32 in the spring), we can use a 70/30 ratio to split the data for training and testing purposes. We'll pilot the AR model using the fall data only (Figure 3). As we'll see later, there are more complex, appropriate approaches that can be used to model the full ENGL101 dataset. However, for instructional purposes, I will attempt to fit the model and choose parameters by hand.

Figure 3. Comparing observed, predicted values for fall data autoregression



Visualization of the fall predictions relative to the observed values isn't encouraging, and the MSE of 4528764.060 is enormous. This model is clearly underfit, and, as visualization shows, the model seems to be making superfluous predictions farther out in time.

However, knowing what we do about autoregression, this isn't surprising. For example, autogression is often used to model random processes that vary with time. While enrollment can be affected by a number of factors, it would be difficult to argue that enrollment is random. At the aggregate, we know that enrollment will be within a particular range based on the number of students currently enrolled as well as those newly admitted in a semester. We also know that autoregression is based in a linear model, and the plots above show evidence against the assumption that enrollment datapoints are linearly related.

3.2.2 ARIMA, by hand

A more appropriate model to consider is the full ARIMA, or *auto-regressive*, *integrated*, *moving-average*, model, defined as:

$$y_t = \delta + \{\phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p}\} + \{\theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}\} + \epsilon_t$$

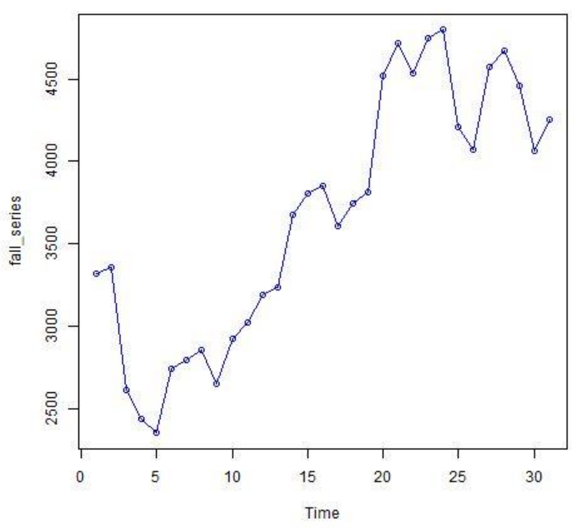
$$\implies y_t = \delta + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

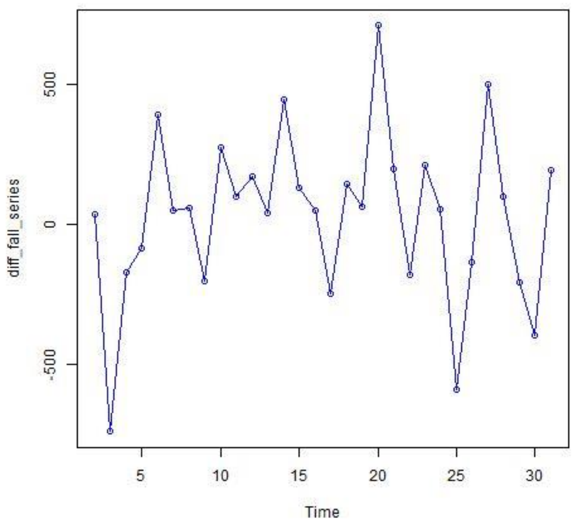
An ARIMA model incorporates three components: 1) the auto-regressive piece from the AR model tried above; 2) a differencing step needed to remove any degree of seasonality from the data; and 3) a moving-average piece, which uses an average of the datapoints in the specified window to calculate estimations for future time points.

Conceptually, it makes more sense to employ an ARIMA for our data, knowing there very well may be both trend and seasonality in the enrollment data. Specifically, it can help us to capture the pattern of the data, while more effectively capture the variation over time by allowing a non-constant mean. To incorporate an ARIMA, we'll try two approaches. First, we'll hand-select parameters *p, d, q* and compare three different instantiations of the ARIMA based on informative visualizations—ACF and PACF plots. Then, we'll try out the auto-ARIMA approach offered by the “forecast” package in R to see how the model we chose by hand compares with the model selected by automation.

Returning to the idea of stationarity, in order to make the data stationary and remove the trend component, we need to take a difference and keep taking differences until the trend is removed. The number of times it takes to do this will be the *d* parameter in the ARIMA.

Figure 4. Fall data pre-differencing (left plot) and after one differencing (right plot)

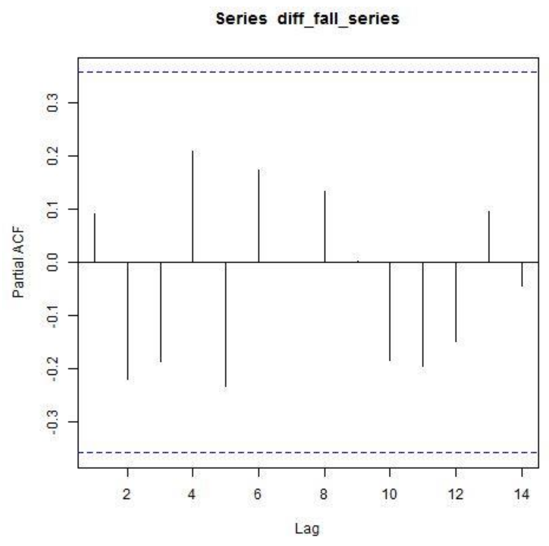


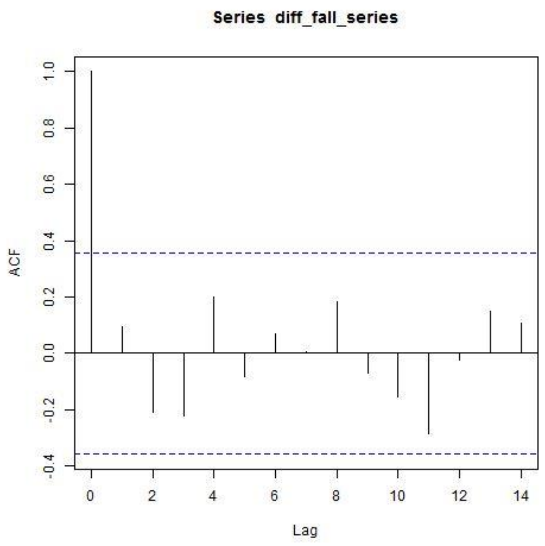


After one differencing, the positive trend apparent in the pre-differencing plot (Figure 4) looks to have been removed. As a conservative approach, we can stop with one differencing—in fact, we get an error when we try to run a second differencing. For the ARIMA, this will give *d* = 1. These steps were subsequently followed for the summer and spring data.

After choosing a value for the *d* parameter, we need *p* and *q*. For *p*, we'll use a partial auto-correlation plot to see after how many lags, we see a drop-off. That drop-off point will be the value we choose for the *p*, the ARIMA's AR parameter. For the model's AM parameter, *q*, we'll look at an auto-correlation plot of the differenced data for each semester for a similar cut-off point (Figure 5).

Figure 5. PACF (left) and ACF (right) plots for fall data





For the fall data, both the PACF and ACF plots show a cut-off after 1 lag, so for the fall ARIMA, we'll specify that p = 1 and q = 1. The spring data is similar to the summer data, in that the PACF plots showed we see no cut-off in the PACF, but again the sharp cut-off after 1 in the ACF. We'll use the same parameters for summer and spring, p = 0 and q = 1.

For the fall data, I tested three separate models: ARIMA(0,1,1), ARIMA(1,1,0), and ARIMA(1,1,1). For each of the summer and spring data, I tested ARIMA(0,1,0) and ARIMA(0,1,1). For the fall data, the second model instantiation, ARIMA(1,1,0) returns the best AIC of the three hand-selected model parameter groupings, and we can use AIC as a measure of model quality. For the summer data, the second instantiation, ARIMA(0,1,1), returns the best AIC, and for the spring data, ARIMA(0,1,0) delivered the best AIC.

So far, we have completed parameter-selection process for fitting the most appropriate ARIMA for each of the three data subsets. In the next section, I'll explore what an automated process produces and compare the result with the results just presented.

3.2.3 ARIMA, automated

To automate the ARIMA for each of the seasonal subsets, we will use the “forecast” package in R. For this kind of analysis, the automated method R provides is advantageous, as it automatically selects the optimal parameters for a given dataset. In the output, accuracy measures such as the AIC and log-likelihood are also given so that we can compare the results of the automation with those conducted above manually.

For the fall data, the automation found ARIMA(0,1,0) to be the best fit; for the summer data, ARIMA(1,1,0); and for the spring, ARIMA(0,1,0). Ultimately, the only model parameters that the automated ARIMA and the manual ARIMA agreed upon were for the spring data. But, comparing the results of the selection by-hand and by automation, the results are not widely disparate. The degree of differencing needed for each of the datasets was consistent across both procedures, and the automatically selected parameters for *p* and *q* were close.

4. Outcomes and conclusion

As an outcome, we might choose to go with the automated ARIMA, as the automated process provides a rigorous search through multiple combinations of parameters, and the by-hand selection only tested a handful of combinations. However, the process to step through model building is an instructive one as it forces the developer to understand how the model works “under the hood”, ultimately providing a more informed understanding of how a solution can be produced, automated or not. However, as data scientists, we are in an age where allowing the computer to do "computer tasks" is an advantage we are afforded. Namely, we can use the computer to automate tasks and take advantage of speed, and it is just one of the benefits of using statistical techniques and tools such as these. Ultimately, though, the interpretation of the results and their application is the job of a data scientist. Unlike computers, humans are skilled at the “why” of a solution, in both explaining the implications and the ways in which it can ethically and responsibly be applied to a real-world problem.

In the future, such a tool as this could be presented to users on a dashboard with prompts to allow the selection of semester type and course number. From this dashboard, historical data could be pulled from the warehouse and the algorithms and methods described here used to model the data for that particular course. Additional prompts are possible as well, for example, the college to which the course belongs or the upper- versus lower-level division, in the event a course is offered as both a lower-level course and an upper-level version. Projecting seats at the level of each individual class, however, will ultimately minimize the amount of noise and allow are more tailored, likely more accurate approach.