Conducting Markov Chain Enrollment Projections at a Public Research University using SAS — A Case Study

This paper was written to accompany the 2018 SAIR Pre-Conference Workshop “Predicting Student Enrollment Using Markov Chain Modeling in SAS” Presented October 7, 2018

Samantha Bradley

Business and Technology Applications Analyst
Office of Institutional Research
University of North Carolina at Greensboro

Submitted for consideration for the 2018 SAIR Best Paper Award December 27, 2018
Conducting Markov Chain Enrollment Projections at a Public Research University using SAS — A Case Study

Abstract

The following is a case study detailing one method of predicting student enrollment at a large public research university. These efforts were conducted by the Office of Institutional Research (OIR) at the University of North Carolina at Greensboro (UNCG) as part of the university’s annual funding and enrollment growth planning cycle.

The purpose of this case study is twofold: 1) to establish that a Markov Chain model is a good fit for university enrollment projections, and 2) to illustrate how to conduct such modeling using Base SAS software.

Background

The University of North Carolina at Greensboro (UNCG) is a public, coeducational state university founded in 1891. UNCG is one of 16 public universities within the University of North Carolina (UNC) system. All members of the UNC System are accredited, independent universities that award their own degrees. UNCG is the largest state university in the Piedmont Triad area of Greensboro, High Point, and Winston-Salem, comprised of eight academic units with nearly 20,000 students, 2,700 faculty and staff, and an annual economic impact of more than $1 billion. State appropriations comprise about 40% of UNCG’s budget, with the rest consisting of revenue generated from tuition and fees, grants and contracts, federal financial aid, sales and services, other operating revenue, and gifts.

To obtain state appropriations, UNCG’s budget planning committee conducts an extensive process beginning with developing enrollment change estimates, compiling expansion requests, and gathering continuation budget requests. Next, these requests are passed on to the UNC System Office (SO) and Board of Governors. The requests then move on to the Office of the State Budget and Management and the Governor. The UNC General Assembly, the NC House and Senate, and the Governor must review all the budget requests from each school in the UNC System and come to an agreement on funding for the upcoming academic year. Finally, the UNC SO and Board of Governors approves the allocations and sends them to UNCG, where the Chancellor allocates the funds throughout the university.

The Importance of Predicting Enrollment

Enrollment projections are a major component of the budget request package, so it is extremely important to develop accurate predictions. During the budget planning process, UNCG’s budget planning committee utilizes several different methods of predicting enrollment through different constituents across the university. The Office of Institutional Research (OIR) is responsible for producing mathematically-based estimates of future enrollment. The deans of UNCG’s eight academic units also produce qualitative unit-level and department-level estimates, while Enrollment Management provides estimates of incoming freshmen cohorts and the Division of Online Learning predicts online student enrollment trends. The budget planning committee compiles and reviews the various enrollment projections and then aggregates them to obtain an overall enrollment estimate that is submitted to SO.
The enrollment projections produced as part of the budget planning cycle also have other useful applications across campus. OIR shares our enrollment projections with department heads to help them plan their internal resource allocation. Our enrollment projections also help the university administration identify areas of growth potential, such as which sectors of the student population are growing fastest, which are retaining, and which could be improved.

OIR annually prepares two forms of enrollment projections; we predict headcounts by student level, and also student credit hours by cost category. This case study focuses on the Markov Chain model OIR used to estimate headcount enrollment for the most recent budget planning cycle, specifically predicting enrollment for the Spring 2018 semester.

**Markov Chain Model**

For the uninitiated, the Markov Chain is a matrix-based time series model of a stochastic process that estimates the evolution of a population over time. The population must be categorized into exhaustive, mutually exclusive groups, referred to as “states”. The Markov Chain model estimates the probability of moving from one state to another, or staying in the same state, between time periods. Applying these probabilities to a base population in one time period lets us predict the distribution of the population in the next time period.

The Markov Chain model is well-established as a method of predicting enrollment at many universities, including large public universities similar to UNCG such as the University of Memphis, Indiana University, and the University of Central Florida.

Use of the Markov Chain model is well-established for several reasons. First, the Markov Chain model is built upon a simple, intuitive equation. It is scalable; so long as the states defined within the model are exhaustive and mutually exclusive, we can select broad or specific population categories and estimate movements accordingly.

Second, the data needed for a Markov Chain model is readily available. For enrollment projections, the Markov Chain makes use of cross-sectional, student-level historical enrollment data that many colleges and universities already maintain.

Finally, the progression of a student’s academic career follows a sequential flow of possible movements that fits well into the Markov Chain framework. The Markov Property, known as the “memoryless” property, of the Markov Chain framework states that the conditional probability of the next state occurring, given the current state and the sequence of state preceding it, is dependent only on the current state.

**Basic Markov Chain Model**

For example, say we’re estimating next semester’s enrollment for a current undergraduate population divided into four levels: freshmen, sophomores, juniors, and seniors. The Markov Property dictates if a student’s current state is a junior, any previous states (i.e. whether they were once a freshman or sophomore) have no bearing on what level they will be next semester. The Markov Chain model takes a student who is currently a junior and estimates the probability that they will remain a junior or move to some other level next semester. In other words, we don’t need to know what they used to be, we only need to know what they are now to predict what they’re going to be.

We can demonstrate the simplest application of a Markov Chain in a university enrollment model by beginning with the undergraduate population in a given semester at time $t$, divided exhaustively and
mutually exclusively into $N$ unique states. For this example, we’ll use student level for our states: freshmen, sophomores, juniors, and seniors ($N=4$). We arrange the headcounts of students in each state at time $t$ into a $1 \times N$ baseline matrix:

$$\begin{bmatrix}
F_t & S_t & J_t & R_t
\end{bmatrix}$$

**Figure 1.** Categorizing the baseline undergraduate student population into four states.

To predict the number of students in each state next semester, time $t+1$, we multiply the baseline matrix by an $N \times N$ matrix that contains the probabilities of moving amongst each state. This is called the “transition probability matrix”, and in our example its dimensions are $4 \times 4$. The rows represent states from which a student can move, and the columns represent states to which a student can move.

We build the transition probability matrix by analyzing previous semesters of enrollment data. For example, say we are currently in a Fall semester and we want to predict enrollment for the upcoming Spring semester. We can think of this as predicting how students are going to move amongst states between this Fall and next Spring.

Using current enrollment data, we already know how many students we have in each state this Fall. Using historical data, we also know how many students we had in each state last Fall and last Spring. We use the previous year’s data to track movements between last Fall and last Spring, and use that movement as a predictor of how students will move between this Fall and next Spring.

We simply compare the number of students in each state last Fall to the number of students in each state last Spring, and find the percentage distribution of the Fall freshmen who remained freshmen in Spring, became sophomores in Spring, became juniors in Spring, and became seniors in Spring. These percentages populate the first row of the transition probability matrix. The sum of the first row in the matrix ($P_{FF} + P_{FS} + P_{FJ} + P_{FR}$) is the total percentage of freshmen who remain in the undergraduate population in any one of the four states next semester. Because some students drop out of the university, we expect this sum to be less than 100%.

We do this for each state in the model to obtain the full $N \times N$ transition probability matrix.

Below is the full equation we have built for a simple Markov Chain model predicting enrollment. Given an undergraduate population divided into four states, we take the number of students in each state at

$$\begin{align*}
\text{Fall (t)} & \quad \rightarrow \quad \text{Spring (t+1)} \\
\text{Freshman} & \quad \rightarrow \quad \text{Freshman} \\
\text{Sophomore} & \quad \rightarrow \quad \text{Sophomore} \\
\text{Junior} & \quad \rightarrow \quad \text{Junior} \\
\text{Senior} & \quad \rightarrow \quad \text{Senior}
\end{align*}$$

**Figure 2.** Possible movements among states within an undergraduate population from one semester to another.

1 As in this example, to predict enrollment for a Spring semester, the model focuses on movements from Fall semesters into Spring semesters. To predict enrollment for a Fall semester, the model would focus on movements from Spring semesters into Fall semesters.
baseline semester $t$ and multiply by the calculated transition probability matrix. This produces a $1 \times N$ matrix of the estimated number of students in each state in the next semester $t+1$.

\[
\begin{bmatrix}
F_t \\ S_t \\ J_t \\ R_t \\
\end{bmatrix}
\times
\begin{bmatrix}
P_{FF} & P_{FS} & P_{FJ} & P_{FR} \\
P_{SF} & P_{SS} & P_{SJ} & P_{SR} \\
P_{jF} & P_{jS} & P_{jJ} & P_{jR} \\
P_{RF} & P_{RS} & P_{RJ} & P_{RR} \\
\end{bmatrix}
= \begin{bmatrix}
F_{t+1} \\ S_{t+1} \\ J_{t+1} \\ R_{t+1} \\
\end{bmatrix}
\]

Baseline population vector at time $t$ Multiplied by Transition probability matrix Equals Estimated number of students in each state at time $t+1$

Figure 3. The equation for a simple Markov Chain forecasting undergraduate enrollment from one semester to another.

This is a simple example of a Markov Chain model predicting enrollment based on one previous year of data. In practice, this is a straightforward, sufficient model that is easily calculated and can provide valuable insight for decision making and budgeting at a university. However, with the wealth of data available through campus’s enterprise resource planning systems like Banner and the analytics capabilities of modern IR offices, we can further develop this traditional model to improve its accuracy and its strength.

**Enhanced Markov Chain Model**

The first way that OIR enhanced the traditional model is to expand the time periods used to obtain the transition probability matrix. As Chen (2014) describes, we can calculate transition probability matrices for multiple time periods, and when we have multiple transition probability matrices that are consistent and time-homogenous, we can average them to create the most efficient and accurate overall transition probability matrix. This is especially useful and applicable for an institutional research office, given the robust historical trend data at our disposal.

When building the model to predict enrollment for Spring 2018, OIR used five years of historical enrollment data to calculate five transition probability matrices for the following semester pairs:

- Fall 2012 to Spring 2013,
- Fall 2013 to Spring 2014,
- Fall 2014 to Spring 2015,
- Fall 2015 to Spring 2016,
- and Fall 2016 to Spring 2017

which were then averaged to create a master transition probability matrix.

The next thing we did to enhance the model was to increase the specificity of the states within the model. We concatenated four different variables to create states that are exhaustive, mutually-exclusive, and extremely detailed. This allows us to track even minute student movements as they progress throughout their academic careers. For example, 3_1_1_F represents a new freshman pursuing a Bachelor’s degree enrolled full-time.

The four variables concatenated into states are shown below:
The table below describes the degree a student is currently seeking, their current enrollment status, the student's current class level, and their current time status:

<table>
<thead>
<tr>
<th>Value</th>
<th>Label</th>
<th>Value</th>
<th>Label</th>
<th>Value</th>
<th>Label</th>
<th>Value</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Post Baccalaureate Certificate</td>
<td>1</td>
<td>New Student</td>
<td>1</td>
<td>Freshman</td>
<td>F</td>
<td>Full-time</td>
</tr>
<tr>
<td>3</td>
<td>Bachelor’s</td>
<td>2</td>
<td>New Transfer Student</td>
<td>2</td>
<td>Sophomore</td>
<td>P</td>
<td>Part-time</td>
</tr>
<tr>
<td>4</td>
<td>Master’s</td>
<td>3</td>
<td>Continuing Student</td>
<td>3</td>
<td>Junior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Post Master’s Certificate</td>
<td>4</td>
<td>Returning Student</td>
<td>4</td>
<td>Senior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Unclassified</td>
<td>6</td>
<td>Unclassified</td>
<td>6</td>
<td>Unclassified Undergraduate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Doctoral Professional</td>
<td>7</td>
<td>Graduate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Variables used to categorize student population into states.

When applied to UNCG’s enrollment data for the past five years, concatenating these variables resulted in 84 distinct states.

We also considered deficiencies in the traditional model and sought ways to improve its accuracy. After analyzing historical predictions and comparing our results to actual enrollments, we determined that the traditional model was mostly deficient in not accounting for new incoming students.

Within the Markov Chain framework, exits from the system are accounted for in the transition probability matrix (a row percentage less than 100% inherently estimates exits), but new entries to the system must be modeled separately. OIR analyzed the pools of new incoming students every semester over the last five years and found UNCG’s population of new entries to be very stable and consistent from semester to semester. Therefore, we chose to model new entries using linear regression.

Figure 5. An abbreviated example of possible movements into, out of, and among 84 distinct states within UNCG’s enhanced Markov Chain model from one semester to another.
We counted the number of new entries into each of the 84 states for the past five Spring semesters. We conducted linear regression on the new entries in each state to estimate new entries into every possible state in the model. The estimates of new entries were then compiled into a 1x84 vector to be added onto the model.

<table>
<thead>
<tr>
<th>Semester</th>
<th>Number of New Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2018 estimate</td>
<td>141</td>
</tr>
<tr>
<td>Spring 2017</td>
<td>118</td>
</tr>
<tr>
<td>Spring 2016</td>
<td>109</td>
</tr>
<tr>
<td>Spring 2015</td>
<td>74</td>
</tr>
<tr>
<td>Spring 2014</td>
<td>61</td>
</tr>
<tr>
<td>Spring 2013</td>
<td>41</td>
</tr>
</tbody>
</table>

**Figure 6.** Example of modeling new entries into a particular state.

After expanding the time periods used to build the transition probability matrix, creating very detailed states to categorize and track our student population, and using linear regression to estimate new entries, the resulting model satisfied OIR’s need for efficiency, accuracy and practicality. Below is the complete formula for UNCG’s enhanced Markov Chain enrollment model.

\[
\begin{align*}
\begin{bmatrix}
3\_1\_1\_F_{t} & 3\_1\_1\_P_{t} & 3\_2\_3\_F_{t} & \ldots \\
\end{bmatrix}
\times
\begin{bmatrix}
P_{3\_1\_1\_F} & P_{3\_1\_1\_F} & P_{3\_1\_1\_F} & \ldots \\
P_{3\_1\_1\_P} & P_{3\_2\_3\_F} & P_{3\_2\_3\_F} & \ldots \\
P_{3\_2\_3\_F} & P_{3\_2\_3\_F} & P_{3\_2\_3\_F} & \ldots \\
\vdots & \vdots & \vdots & \ddots \\
\end{bmatrix}
+
\begin{bmatrix}
3\_1\_1\_F_{\text{new}} & 3\_1\_1\_P_{\text{new}} & 3\_2\_3\_F_{\text{new}} & \ldots \\
\end{bmatrix}
=
\begin{bmatrix}
3\_1\_1\_F_{t+1} & 3\_1\_1\_P_{t+1} & 3\_2\_3\_F_{t+1} & \ldots \\
\end{bmatrix}
\end{align*}
\]

**Baseline population vector at time \( t \)\)**

**Multiplied by**

**Overall transition probability matrix**

**Plus**

**Predicted new entry vector**

**Equals**

**Estimated number of students in each state at time \( t+1 \)**

\( (\text{Fall 2017 actual enrollment data}) \)

\( (\text{Average of five transition probability matrices based on historical data}) \)

\( (\text{Predicted new entries in Spring 2018}) \)

\( (\text{Spring 2018 enrollment projections}) \)

**Figure 7.** The equation for an enhanced Markov Chain model forecasting enrollment from one semester to another.

**Markov Chain Modeling in SAS**

As mentioned above, to predict enrollment for Spring 2018, OIR used historical enrollment data reaching back to Fall 2012. Our historical enrollment dataset contains one record per student per semester, is built and maintained in SAS, and at the time of our analysis had 201,445 records and 324 variables.
The Office of Institutional Research at UNCG is a SAS-focused office. The majority of our data are stored in SAS datasets, and most analyses are conducted in Base SAS 9.4 or SAS Enterprise Guide. OIR’s enrollment projections model was built in Base SAS 9.4.

While Markov Chain modeling can be conducted in other programs like Microsoft Excel, SAS is by far the most effective tool, especially given the amount and complexity of the data in our model. SAS can process massive amounts of data very quickly. SAS also allows us to easily conduct multiple kinds of analyses in the same program; within one program we can cross tabulate data, run regressions, and build and perform calculations with massive matrices. Perhaps most importantly, SAS allows us to program a dynamic, efficient model using macros that requires very little user modification from semester to semester.

**Dynamic SAS Programming**

OIR intentionally built a dynamic SAS program for enrollment projections that is hands-off and user-friendly. The program makes extensive use of macro variables and macro programs; the only modification required to run the program is to input the semester code for which we’re predicting enrollment. Dynamic SAS programs minimize the risk of user-error, are simple to update, produce compact program files, and, most importantly, are efficient.

The following chart summarizes our methodology of building the Markov Chain model in SAS:

| STEP 1 | Read in the data  
Student-level data for the most recent term and past 5 years  
Concatenate variables (DEGREE, ENROLLMENT, CLASS, TIME) to create states |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 2</td>
<td>Create five datasets of semester pairings of Falls &gt; Springs</td>
</tr>
</tbody>
</table>
| STEP 3 | Create 5 transition probability matrices for each semester pairing  
Compare semester pairings to determine what percentage of students in each state retained, dropped out, or moved to another state |
| STEP 4 | Average the 5 transition probability matrices to create an overall transition probability matrix |
| STEP 5 | Create a dataset of last semester’s enrollment values as the baseline population |
| STEP 6 | Use linear regression to model new entries |
| STEP 7 | Use PROC IML to forecast enrollment for next semester |

*Figure 8. Summary of SAS methodology.*

See Appendix for full SAS code.

**Results**

OIR’s model consistently produces estimates within 2% of actual enrollments. For Spring 2018, the model predicted enrollment of 18,669 and ultimately UNCG enrolled 18,846 students. This is a difference of 177 students, or 0.9% of actual enrollment.
<table>
<thead>
<tr>
<th>Semester</th>
<th>Undergraduate</th>
<th>Non-Degree Seeking Undergraduate</th>
<th>Certificate</th>
<th>Masters</th>
<th>Specialist</th>
<th>Doctoral</th>
<th>Non-Degree Seeking Graduate</th>
<th>Total Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2013</td>
<td>13,563</td>
<td>296</td>
<td>0</td>
<td>2,066</td>
<td>100</td>
<td>778</td>
<td>529</td>
<td>17,332</td>
</tr>
<tr>
<td>Spring 2014</td>
<td>13,294</td>
<td>298</td>
<td>155</td>
<td>1,915</td>
<td>97</td>
<td>735</td>
<td>377</td>
<td>16,871</td>
</tr>
<tr>
<td>Spring 2015</td>
<td>13,702</td>
<td>360</td>
<td>127</td>
<td>1,927</td>
<td>107</td>
<td>764</td>
<td>332</td>
<td>17,319</td>
</tr>
<tr>
<td>Spring 2016</td>
<td>14,265</td>
<td>404</td>
<td>206</td>
<td>1,924</td>
<td>44</td>
<td>794</td>
<td>240</td>
<td>17,877</td>
</tr>
<tr>
<td>Spring 2017</td>
<td>14,874</td>
<td>420</td>
<td>167</td>
<td>1,838</td>
<td>22</td>
<td>920</td>
<td>270</td>
<td>18,511</td>
</tr>
<tr>
<td>Projected</td>
<td>Spring 2018</td>
<td>15,105</td>
<td>398</td>
<td>130</td>
<td>1,925</td>
<td>38</td>
<td>834</td>
<td>18,669</td>
</tr>
<tr>
<td>Actual</td>
<td>Spring 2018</td>
<td>15,116</td>
<td>388</td>
<td>143</td>
<td>1,965</td>
<td>21</td>
<td>981</td>
<td>18,846</td>
</tr>
</tbody>
</table>

**Figure 9.** Spring semester enrollment data with Spring 2018 projections versus Spring 2018 actual enrollments.

The graphs below provide a visualization of historical and projected enrollment trends for Spring semesters. These show that the estimates produced by OIR’s Markov Chain model are reasonable and in line with historical trends.

**Spring Semester Historical Enrollment vs Projections**
*(Undergraduates displayed separately)*

**Spring Semester Historical Enrollment vs Projections**
*(Undergraduates only)*

**Figure 10.** Visualization of historical and projected enrollment trends for Spring semesters.

The enhanced Markov Chain model allowed OIR to produce Spring 2018 enrollment estimates that we felt very confident in. We also used the Spring 2018 results to extend the model to predict enrollment for Fall 2018, which was then used to predict Spring 2019. While outside the scope of this brief case study, Markov Chain models can be extended to predict further into the future by plugging projections into the model as ‘actual’ enrollments and rolling the model forward, repeating the same methodology.

In October 2017 we submitted our full 2018 to 2019 projections to UNCG’s budget planning committee. Validating the model’s predictions with the true Spring 2018 enrollment numbers at census date in January 2018 further solidified the accuracy and value of the model.
Assumptions and Limitations

A necessary assumption within the Markov Chain model is that the external environment is held constant. In a university setting, this means that the Markov Chain model does not inherently account for things like changes in marketing and recruitment efforts, degree or program structures, pedagogy, or other types of activities the university may be undertaking that would logically be expected to impact future enrollment.

OIR recognizes this limitation, and addresses it by collaborating with other offices around campus who have insight to such activities. For example, when finalizing our enrollment projections, OIR meets with deans and department heads to discuss any degree programs that are being introduced or phased out in upcoming semesters. OIR also discusses issues such as potential tuition changes with the Financial Planning & Budgets office, the anticipated cohort of incoming new freshmen with Enrollment Management, and the outlook for graduate enrollment with Graduate Admissions. If some action is identified that is reasonably certain to impact enrollment for a particular segment of the student population, OIR adjusts the results of the Markov Chain model for the affected state(s). For example, if Enrollment Management reports that the new freshman application rate is greatly exceeding expected rates, OIR may increase the predicted value of state 3_1_1_F (new full-time degree-seeking freshmen), also taking into account expected acceptance and admittance rates. Our final projections are therefore a practical combination of mathematical modeling and realism.

Further Considerations

In future semesters, OIR would like to enhance the Markov Chain modeling process to incorporate more of the aforementioned external factors that influence enrollment. For example, UNCG is heavily expanding our online degree programs and course offerings, and our Division of Online Learning anticipates faster growth than our model can currently predict using historical data. Also, it may be useful to incorporate the local high school graduation rate because 40–50% of new incoming freshmen every semester come from Guilford County, where UNCG is located, and the surrounding five counties.

UNCG is also in a unique position in that it is one of seven colleges, universities, and community colleges in Guilford County. Many UNCG students transfer into or out of these schools. Given their proximity and accessibility, the effects of mechanisms like tuition rate changes and expanding or condensing program offerings at other schools are felt by UNCG. It would be challenging to incorporate this into our enrollment projections model, but it is prudent to explore ways to quantitatively account for processes that have traditionally only been intuitively adjusted for.
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Categorizing the baseline undergraduate student population into four states.</td>
<td>p. 3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Possible movements among states within an undergraduate population from one semester to another.</td>
<td>p. 3</td>
</tr>
<tr>
<td>Figure 3</td>
<td>The equation for a simple Markov Chain forecasting undergraduate enrollment from one semester to another.</td>
<td>p. 4</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Variables used to categorize student population into states.</td>
<td>p. 5</td>
</tr>
<tr>
<td>Figure 5</td>
<td>An abbreviated example of possible movements into, out of, and among 84 distinct states within UNCG’s enhanced Markov Chain model from one semester to another.</td>
<td>p. 5</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Example of modeling new entries into a particular state.</td>
<td>p. 6</td>
</tr>
<tr>
<td>Figure 7</td>
<td>The equation for a simple Markov Chain forecasting undergraduate enrollment from one semester to another.</td>
<td>p. 6</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Summary of SAS methodology.</td>
<td>p. 7</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Spring semester enrollment data with Spring 2018 projections versus Spring 2018 actual enrollments.</td>
<td>p. 8</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Visualization of historical and projected enrollment trends for Spring semesters.</td>
<td>p. 8</td>
</tr>
</tbody>
</table>
Appendix

libname epdata 'N:\Custom\Samantha\Enrollment Projections\data'; /* assign the library where your data is stored */
options mprint mlogic symbolgen; /* these options let us de-bug macros */
/* First, and most important, what term are we predicting enrollments for? */ /* This is the only element of the program that needs to be modified */
/**/ %let projection=201801;  /**/ /** CHANGE PROJECTION TERM HERE **/
/* Based on the projection semester, this code creates macro variables to pull in previous semesters of data */
DATA _null_; IF substr("&projection",5,2)="01" THEN DO; /* logic for Spring terms */
semester0=PUT(&projection,6.);
semester1=PUT(&projection-93,6.);
semester2=PUT(semester1-7,6.);
semester3=PUT(semester2-93,6.);
semester4=PUT(semester3-7,6.);
semester5=PUT(semester4-93,6.);
semester6=PUT(semester5-7,6.);
semester7=PUT(semester6-93,6.);
semester8=PUT(semester7-7,6.);
semester9=PUT(semester8-93,6.);
semester10=PUT(semester9-7,6.);
semester11=PUT(semester10-93,6.);
predict_term=substr("&projection",5,2);
season="Spring";
season_yr=substr("&projection",1,4);
END;
ELSE IF substr("&projection",5,2)="08" THEN DO; /* logic for Fall terms */
semester0=PUT(&projection,6.);
semester1=PUT(&projection-7,6.);
semester2=PUT(semester1-93,6.);
semester3=PUT(semester2-7,6.);
semester4=PUT(semester3-93,6.);
semester5=PUT(semester4-7,6.);
semester6=PUT(semester5-93,6.);
semester7=PUT(semester6-7,6.);
semester8=PUT(semester7-93,6.);
semester9=PUT(semester8-7,6.);
semester10=PUT(semester9-93,6.);
semester11=PUT(semester10-93,6.);
predict_term=substr("&projection",5,2);
season="Fall";
season_yr=substr("&projection",1,4);
END;
CALL SYMPUT('semester0',semester0);
CALL SYMPUT('semester1',semester1);
CALL SYMPUT('semester2',semester2);
CALL SYMPUT('semester3',semester3);
CALL SYMPUT('semester4',semester4);
CALL SYMPUT('semester5',semester5);
CALL SYMPUT('semester6',semester6);
CALL SYMPUT('semester7',semester7);
CALL SYMPUT('semester8',semester8);
CALL SYMPUT('semester9',semester9);
CALL SYMPUT('semester10',semester10);
CALL SYMPUT('semester11',semester11);
CALL SYMPUT('term0', (cats("_",semester0)));
CALL SYMPUT('term1', (cats("_",semester1)));
CALL SYMPUT('term2', (cats("_",semester2)));
CALL SYMPUT('term3', (cats("_",semester3)));
CALL SYMPUT('term4', (cats("_",semester4)));
CALL SYMPUT('term5', (cats("_",semester5)));
CALL SYMPUT('term6', (cats("_",semester6)));
CALL SYMPUT('term7', (cats("_",semester7)));
CALL SYMPUT('term8', (cats("_",semester8)));
CALL SYMPUT('term9', (cats("_",semester9)));
CALL SYMPUT('term10', (cats('_', semester10)));  
CALL SYMPUT('term11', (cats('_', semester11)));  
CALL SYMPUT('predict_term', predict_term);  
CALL SYMPUT('season', season);  
CALL SYMPUT('season_yr', season_yr); 
RUN;  
&PUT _USER_;  

/* Pull in Enrollment data for the past 11 terms - we need data for the last 5 full fiscal years, and the directly previous term. */  
/* Macro &semester1 resolves to the directly previous term. */  
DATA sair_sample;  
SET epdata.sair_sample;  
WHERE TERMCODE IN ("&semester11", "&semester10", "&semester9", "&semester8", "&semester7", "&semester6", "&semester5", "&semester4", "&semester3", "&semester2", "&semester1");  
RUN;  
/* Let's print a few observations to see what the data looks like. */  
PROC PRINT DATA=sair_sample (obs=30);  
TITLE "First 30 records in dataset";  
RUN;  

/* Create a 'state' variable to categorize students at every time point throughout their career */  
/* This concatenates the following variables: */  
/* DEGREE:  
0-Post Baccalaureate Certificate  
1-Bachelor's  
2-Master's  
3-Post Master's Certificate  
4-Doctoral Professional  
5-Doctorate  
6-Unclassified  */  
/* ENRL:  
1-New Student  
2-New Transfer Student  
3-Continuing Student  
4-Returning Student  
5-Unclassified  */  
/* CLASS:  
1-Freshman  
2-Sophomore  
3-Junior  
4-Senior  
5-Unclassified Undergraduate  
6-Graduate  */  
/* TIME:  
F-Full-time  
P-Part-time  */  

DATA one;  
SET sair_sample;  
state = CATS('_', DEGREE, ' ', ENROLL, ' ', CLASS, ' ', TIME);  
STATE DESC = CATX(' ', DEGREE DESC, ENROLL DESC, CLASS DESC, TIME DESC);  
RUN;  
PROC SORT DATA=one; BY SAIR_ID; RUN;  
/* Let's see what our state categories look like. */  
PROC SQL;  
SELECT DISTINCT state, state DESC  
FROM one;  
TITLE "Student 'state' categories";  
QUIT;  
/* For a Markov Chain model, we need an NxN matrix with every possible state. */  
/* We will build this in a series of dynamic steps that won't require editing each term. */  
/* This gives us a list of every unique state, and a macro variable that holds the number of distinct states. */  
PROC SQL NOPRINT;  
CREATE TABLE states AS SELECT DISTINCT state FROM one;  
SELECT COUNT(DISTINCT(state)) INTO :CnT SEPARATED BY " " FROM one;  
QUIT;  
/* Now we can transpose the list to start building our matrix template. */  
/* Make sure to use COPY to keep the columns of states. */  
/* This will make the values of the states into variables we can use. */  
PROC TRANSPOSE DATA=states OUT=state matrix NAME=state;  
VAR state;  
ID state;  
COPY state;  
RUN;
/* Create new variables with suffix _n that will eventually become our numeric variables */
/* Also create flags for each student level we'll be predicting at the end. */
DATA vars;
SET states;
state n=trim(LEFT(state))||"_n";
IF substr(state,1,2)="0" THEN student_cat="certificate";
ELSE IF substr(state,2,1)="3" THEN student_cat="undergrad";
ELSE IF substr(state,2,1)="4" THEN student_cat="masters";
ELSE IF substr(state,2,1)="5" THEN student_cat="specialist";
ELSE IF substr(state,2,1)="8" AND substr(state,6,1) NE "7" THEN student_cat="ug non-degr";
ELSE IF substr(state,2,1)="8" AND substr(state,6,1)="7" THEN student_cat="gr non-degr";
ELSE IF substr(state,2,1) IN("P","R") THEN student_cat="doctorate";
RUN;

/* Create macro variables to capture each student level. */
PROC SQL NOPRINT;
SELECT TRIM(LEFT(state))
INTO :cert SEPARATED BY ','
FROM vars
WHERE student_cat="certificate";
SELECT TRIM(LEFT(state))
INTO :ugrd SEPARATED BY ','
FROM vars
WHERE student_cat="undergrad";
SELECT TRIM(LEFT(state))
INTO :mstr SEPARATED BY ','
FROM vars
WHERE student_cat="masters";
SELECT TRIM(LEFT(state))
INTO :spcl SEPARATED BY ','
FROM vars
WHERE student_cat="specialist";
SELECT TRIM(LEFT(state))
INTO :ugnd SEPARATED BY ','
FROM vars
WHERE student_cat="ug non-degr";
SELECT TRIM(LEFT(state))
INTO :grnd SEPARATED BY ','
FROM vars
WHERE student_cat="gr non-degr";
SELECT TRIM(LEFT(state))
INTO :dctr SEPARATED BY ','
FROM vars
WHERE student_cat="doctorate";
QUIT;
%PUT _USER_;

/* The code below is adapted from SAS Article '40700 - Convert all character variables to numeric and use the same variable names in the output.' */
/* Full code is publicly available at support.sas.com/kb/40/700.html */

/* Now create three macro variables using INTO to create our template of all states. */
/* The first macro variable, c_list, has a list of all the character variables, separated by a space. */
/* The second macro variable, n_list, has a list of all our new variables we made in the previous DATA STEP with the suffix _n. */
/* The third macro variable, renam_list, has a list of all our new _n variables and all our old character variables separated by an equal sign (new = old). */
PROC SQL nobprint;
SELECT TRIM(LEFT(state)),
	TRIM(LEFT(state_n))
INTO :c_list SEPARATED BY ' ',
	:n_list SEPARATED BY ' ',
	:renam_list SEPARATED BY ' ' FROM vars;
QUIT;

/* Now we start the conversion! */
/* Create the array ch by calling the macro variable &c_list, which resolves to the list of all the character variables. */
/* Create the array nu by calling the macro variable &n_list, which resolves to the list of all the numeric variables with suffix _n. */
/* Then loop through the arrays and use the INPUT function to convert the character variables to numeric, stored in our _n variables. */
/* Then loop through the nu array and change all the missing values to 0. */
/* Finally, rename all the new _n variables back to the original variable names by calling the macro variable &renam_list. */
DATA template;
SET statematrix;
ARRAY ch(*) &c_list;
ARRAY nu(*) &n_list;
DO i = 1 TO dim(ch);
nu(i)=input(ch(i),?8.);
END;
DO i=1 TO dim(nu);
IF nu(i).= . THEN nu(i)=0;
END;
DROP i &c_list;
RENAME &renam_list;
RUN;
/** Now we have a N rows by N+1 columns matrix with each flow state and every interior value=0. Column value is N+1 because the first column contains our state values. **/
PROC PRINT DATA=template noobs;
TITLE "NxN Markov Chain Matrix Template";
RUN;
/** Create macro variables to represent each individual 'state'. **/
DATA _null_
SET vars;
count=LEFT(PUT(_n_,5.));
cALL SYMPUT(’Var’||count,state);
RUN;
&PUT _USER_

*************************************************************************************************************************************************/
/** Transpose the Enrollment data so that we have one record for every student, with their state category for each term. **/
PROC TRANSPOSE DATA=one OUT=two;
BY SAIR_ID;
VAR state;
ID TERMCODE;
RUN;
/** Make sure the data is in chronological order. */
/** Keep &term1 through &term11. We’ll be making matched pairs of Spring>Fall and Fall>Spring, up through the most recent full year. */
/** So for predicting a Fall term, we’ll make matched pairs of Spring>Fall up through the most recent full year. */
/** Similarly, for predicting a Spring term, we’ll make matched pairs of Fall>Spring up through the most recent full year. */
DATA two_clean;
RETAIN SAIR_ID &term11 &term10 &term9 &term8 &term7 &term6 &term5 &term4 &term3 &term2 &term1;
SET two;
KEEP SAIR_ID &term11 &term10 &term9 &term8 &term7 &term6 &term5 &term4 &term3 &term2 &term1;
RUN;
*************************************************************************************************************************************************/
/** We need to build probability matrices for each semester pairing. */
/** To predict for a Fall semester, we look at enrollment trends for previous years, tracking enrollment as it flows from Spring into Fall. */
/** For example- To predict enrollment for Fall 2017, the pairs are:
1. Spring 2016 to Fall 2016
2. Spring 2015 to Fall 2015
3. Spring 2014 to Fall 2014
4. Spring 2013 to Fall 2013
5. Spring 2012 to Fall 2012
**/
/** To predict for a Spring semester, we look at enrollment trends for previous years, tracking enrollment as it flows from Fall into Spring. */
/** For example- To predict enrollment for Spring 2018, the pairs are:
1. Fall 2016 to Spring 2017
2. Fall 2015 to Spring 2016
3. Fall 2014 to Spring 2015
4. Fall 2013 to Spring 2014
5. Fall 2012 to Spring 2013
/** pair1 will be the number of our dataset, based on how many years back we're going - 1 through 5 */
/** pair2 will be first term in pair */
/** pair3 will be second term in pair */
/** pair4 will be semester, equivalent to first term in pair */
%MACRO pairs(pair1, pair2, pair3, pair4);
DATA year&pair1;
SET two_clean;
WHERE &pair2 NE '';
IF &pair2 NE '' AND &pair3='' THEN &pair3="Did not return";
KEEP SAIR_ID &pair2 &pair3;
RUN;
/** Run a crosstab to compare how many students kept the same status or changed from one status to another during these two terms. */
PROC FREQ DATA=year&pair1 noprint;
TABLES &pair2*&pair3 / nocoll nocum missing OUTPCT OUT=crosstab;
RUN;
DATA compare;
SET crosstab;
KEEP &pair2 &pair3 PCT_ROW;
RUN;
/* transpose the data so we can use the crosstab as a dataset */
PROC TRANSPOSE DATA=compare OUT=compare2;
BY &pair2;
VAR PCT_ROW;
ID &pair3;
RUN;
/* set all missing values to 0 and make the values a percent */
DATA matrix_y&pair1;
SET compare2;
ARRAY CHANGE _NUMERIC_
    DO OVER CHANGE;
    IF CHANGE=. THEN CHANGE=0;
    CHANGE=CHANGE*.01;
END;
DROP _LABEL_ _NAME_
RUN;
/* prepare the blank matrix template */
DATA template_y&pair1;
SET template;
RUN;
PROC SORT DATA=template_y&pair1; BY &pair2; RUN;
PROC SORT DATA=matrix_y&pair1; BY &pair2; RUN;
/* fill in the matrix template with the data for the most recent full year */
DATA year&pair1;
FORMAT state $6.
    start_state $8.
    Did_not_return 13.10;
INFORMAT state $6.
    start_state $8.
    Did_not_return 13.10;
LENGTH state $6.
    start_state $8.
    Did_not_return 8
MERGE template_y&pair1(in=a) matrix_y&pair1(in=b);
BY &pair2;
IF a;
    start_state=&pair2;
    state="&pair3."
RUN;
%MEND pairs;
%pairs{1,&term3,&term2,&semester3}; /* &term3 resolves to _201608, &term2 resolves to _201701. so this tracks students from Fall 2016 to Spring 2017 */
%pairs{2,&term5,&term4,&semester5};
%pairs{3,&term7,&term6,&semester7};
%pairs{4,&term9,&term8,&semester9};
%pairs{5,&term11,&term10,&semester11};
/** This gives us 5 datasets containing probability matrices for student enrollment movements over the last 5 years. ***/

/***********************************************************/
/*       Step 4       */
/***********************************************************/
/* Now we have 5 transition probability matrices comparing movements between each semester. */
/* We need to calculate the average probabilities of moving into each of the states over time. */
/* Because enrollment patterns are different depending on whether it's a Fall or Spring semester, we need to model movement from Spring > Fall and Fall > Spring separately. */
/* Stack all the matrices. */
/* Drop the term-specific columns. This data is captured in the variable 'start_state'. Also drop Did_not_return, as those students fall out of the model. */
DATA prob_stack;
SET year1 year2 year3 year4 year5;
DROP &term3 &term5 &term7 &term9 &term11 Did_not_return;
RUN;
/* Sort the data by start_state, to group all start states in each term. */
PROC SORT DATA=prob_stack; BY start_state; RUN;
/* Calculate the average probability of transitioning into each state. */
/* use CNT macro var for number of flow states */
%MACRO mkv_states;
%DO i=1 %TO &cnt;
    DATA state_i;
    SET prob_stack;
    KEEP YEAR start_state &&var&i;
    RUN;
    PROC SORT DATA=state_i; BY start_state; RUN;
    PROC MEANS DATA=state_i nway nway noprint;
    CLASS start_state;
    PROC TRANSPOSE DATA=state_i OUT=state_transpose2;
    VAR 1
    ID 2;
VAR &&var&&;
OUTPUT OUT=state_&i
  [GROUP=_FREQ_ _TYPE_]
  mean=&&var&&;
RUN;
%END;
%MEND mkv_states;
/* Call macro. This will loop through the entire stack of probability matrices, and average the probability of movements between each flow state. */
%mkv_states;

/* Now we'll merge by start_state to create an overall transition probability matrix */
DATA trans_matrix;
MERGE state_1-state_&cnt;
BY start_state;
DROP start_state;
RUN;

/***********************************************************************************/
/* Step 5 */
/***********************************************************************************/

/* Now we need a dataset with the last actual known enrollment values - This will be our baseline population when forecasting. */
DATA current_enroll1;
SET two_clean;
WHERE &term1 NE '';
KEEP SAIR_ID &term1;
RUN;
PROC FREQ DATA=current_enroll1 noprint;
TABLES &term1 / norow nocol nocum nopercent OUT=current_enroll (DROP=PERCENT);
RUN;

/* We need to make sure the baseline dataset includes every possible state, even if we have no students in a particular state for this particular semester. */
/* Pull list of all possible states from the Markov Chain template. */
DATA allstates;
SET template;
RENAME state=&term1;
KEEP state;
RUN;
PROC SORT DATA=allstates; BY &term1;
RUN;
/* Merge the full list of possible states onto the baseline population. */
DATA current_enrl_matrix1;
MERGE allstates(in=a) current_enroll(in=b);
BY &term1;
IF a;
IF COUNT=.
THEN COUNT=0;
RUN;
/* Transpose the data from tall to long. */
PROC TRANSPOSE DATA=current_enrl_matrix OUT=current_enrl_matrix;
VAR COUNT;
ID &term1;
RUN;
/* Clean up the transposed data. */
DATA base_pop;
RETAIN TERM;
SET current_enrl_matrix;
DROP _NAME_ _LABEL_ ;
TERM=&semester1;
RUN;
/** Now we have a 1xN baseline population from which to forecast. **/

/***********************************************************************************/
/* Step 6 */
/***********************************************************************************/
/* New entries into each state must be separately modeled. */
%MACRO entry(i,semestera,semesterb);
/* start at the earliest term and work up */
DATA academicyear;
SET one;
WHERE termcode in("&semestera","&semesterb");
RUN;
PROC SORT DATA=academicyear;
BY SAIR_ID TERMCODE;
RUN;
DATA entry&i;
SET academicyear;
BY sair_id;
IF TERMCODE IN("&semester10","&semester11") THEN years_past=0;
ELSE IF TERMCODE IN("&semester8","&semester9") THEN years_past=1;
ELSE IF TERMCODE IN("&semester6","&semester7") THEN years_past=2;
ELSE IF TERMCODE IN("&semester4","&semester5") THEN years_past=3;
ELSE IF TERMCODE IN("&semester2","&semester3") THEN years_past=4;
ELSE IF TERMCODE IN("&semester0","&semester1") THEN years_past=5;
/* if the first instance of a student ID occurs in the second semester of the pair, they are flagged as a new entry */
IF FIRST.SAIR_ID and termcode="&semesterb" THEN entry=1;
/* only keep new entries */
IF entry NE 1 THEN DELETE;
KEEP termcode ENROLL SAIR_ID state years_past;
RUN;
%MEND entry;
%
entry(1,&semester11,&semester10);
entry(2,&semester9,&semester8);
entry(3,&semester7,&semester6);
entry(4,&semester5,&semester4);
entry(5,&semester3,&semester2);
/* Stack the new entry data. */
DATA new_students;
SET entry1 entry2 entry3 entry4 entry5;
RUN;
PROC SORT DATA=new_students;
BY termcode;
RUN;
/* Now run a frequency to summarize new entries per semester and output results to a new dataset. */
PROC FREQ DATA=new_students noprint;
TABLES years_past*TERMCODE*state/list missing OUT=new_student_terms (DROP=PERCENT);
WHERE SUBSTR(TERMCODE,5,2)="&predict_term"; /* if we're predicting Spring enrollment, we only need to summarize new entries for Spring terms (and vice versa for Fall) */
RUN;
/* Create an empty dataset for the prediction term with every possible state and a blank value for count. */
/* We need to include every state for matrix calculations later. Only the states associated with new entries will be populated, that's okay. */
PROC SQL;
CREATE TABLE predict_new AS SELECT state FROM states;
ALTER TABLE predict_new ADD years_past NUM, count NUM, termcode CHAR;
UPDATE predict_new SET years_past=5;
UPDATE predict_new SET count=0;
UPDATE predict_new SET termcode="&semester0";
QUIT;
/* Stack the counts of new entries with the empty dataset for the prediction term. */
DATA new_states_regress;
SET new_student_terms predict_new;
RUN;
/* Sort the data */
PROC SORT DATA=new_states_regress; BY termcode state;
RUN;
/* By including the prediction term with a blank count, SAS will predict the blank value using the linear regression. */
/* So we can conduct a simple linear regression where COUNT is the dependent variable (COUNT is the number of new entries for each ENRL category in each term) */
/* and years_past is the independent variable. */
%MACRO reg;
%DO i=1 %TO &cnt;
PROC REG DATA=new_states_regress NOPRINT;
MODEL COUNT=years_past;
WHERE state="&&Var&i";
OUTPUT OUT=new_&i predicted=predict_cnt residual=resid;
QUIT;
%END;
%MEND reg;
%reg;
/* Let's look at an example of predicting new entries with linear regression */
PROC PRINT DATA=new_36 noobs;
TITLE "Estimated new entries into state _3_4_4_P - Returning Seniors seeking Bachelor's Degree Part-Time";
RUN;
/* Stack all regression results and keep only the predictions. */
DATA new_predictions;
SET new_1-new_&cnt;
WHERE TERMCODE="&semester0";
IF predict_cnt=. THEN predict_cnt=0;
KEEP state predict_cnt;
RUN;
/* Transpose the data to create a 1xN vector of new entries. */
PROC TRANSPOSE DATA=new_predictions OUT=new_entries1
  DROP=NAME_ _LABEL_;
VAR predict_cnt;
ID state;
RUN;
/* Set negative new entries to 0. */
DATA new_entries;
SET new_entries1;
ARRAY CHANGE _NUMERIC_
   DO OVER CHANGE;
   IF CHANGE<0 THEN CHANGE=0;
RUN;

PROC IML;
vars={&c_list};
USE trans_matrix; READ ALL INTO trans_matrix;
USE base_pop; READ ALL INTO base_pop;
USE new_entries; READ ALL INTO new_entries;
base_pop=base_pop[1, 2:(&cnt+1)];
new_entries=new_entries[1:,&cnt];
Projection=(base_pop*trans_matrix)+new_entries;
CREATE Projection FROM Projection [COLNAME=vars];
APPEND FROM Projection;
QUIT;

/*** We have our projections! ***/
/* Now we can use those student level macro variables we created earlier to aggregate our projections. */
DATA final_projections;
FORMAT Undergraduate COMMA8.0
   UG_Nondegree COMMA8.0
   Certificate COMMA8.0
   Masters COMMA8.0
   Specialist COMMA8.0
   Doctoral COMMA8.0
   GR_Nondegree COMMA8.0
   Total COMMA8.0;
LABEL Undergraduate="Undergraduate"
UG_Nondegree="Non-Degree Seeking Undergraduate"
Certificate="Certificate"
Masters="Masters"
Specialist="Specialist"
Doctoral="Doctoral"
GR_Nondegree="Non-Degree Seeking Graduate"
Total="Total Enrollment";
SET Projection;
Certificate=ROUND(sum(&cert),1);
Undergraduate=ROUND(sum(&ugrd),1);
Masters=ROUND(sum(&mstr),1);
Specialist=ROUND(sum(&spcl),1);
UG_Nondegree=ROUND(sum(&ugnd),1);
GR_Nondegree=ROUND(sum(&grnd),1);
Doctoral=ROUND(sum(&dctr),1);
Total=sum(Certificate,Undergraduate,Masters,Specialist,UG_nondegree,GR_nondegree,Doctoral);
TERM="&semester0";
KEEP TERM Certificate Undergraduate Masters Specialist UG_nondegree GR_nondegree Doctoral Total;
RUN;
PROC PRINT DATA=final_projections noobs label;
VAR Undergraduate UG_Nondegree Certificate Masters Specialist Doctoral GR_nondegree Total;
TITLE "&season &season_yr Projections with New Entries";
RUN;
/* We can also compare our projections with historical enrollment data. */
PROC SORT DATA=one;
   BY termcode state;
RUN;
PROC FREQ DATA=one noprint;
   TABLES termcode*state /list missing OUT=historic1 (DROP=PERCENT);
RUN;
PROC TRANSPOSE DATA=historic1 OUT=historic2;
   VAR COUNT;
   ID state;
   BY termcode;
RUN;
DATA historic;
FORMAT Undergraduate COMMA8.0
   UG_Nondegree COMMA8.0
   Certificate COMMA8.0
   Masters COMMA8.0
   Specialist COMMA8.0
   Doctoral COMMA8.0
   GR_Nondegree COMMA8.0
   Total COMMA8.0;
LABEL Undergraduate="Undergraduate"
UG_Nondegree="Non-Degree Seeking Undergraduate"
Certificate="Certificate"
Masters="Masters"
Specialist="Specialist"
Doctoral="Doctoral"
GR_Nondegree="Non-Degree Seeking Graduate"
Total="Total Enrollment"

SET historic;
Certificate=ROUND(sum(&cert),1);
Undergraduate=ROUND(sum(&ugrd),1);
Masters=ROUND(sum(&mstr),1);
Specialist=ROUND(sum(&spcl),1);
UG_Nondegree=ROUND(sum(&ugnd),1);
GR_Nondegree=ROUND(sum(&grnd),1);
Doctoral=ROUND(sum(&dctr),1);
Total=sum(Certificate,Undergraduate,Masters,Specialist,UG_nondegree,GR_nondegree,Doctoral);
TERM=termcode;
KEEP TERM Certificate Undergraduate Masters Specialist Doctoral Total;
RUN;

/* Now stack the historic data with the projection data. */
DATA historic_project;
SET historic final_projections;
WHERE substr(TERM,5,2)="&predict_term"; /* this stacks Springs together or Falls together */
RUN;

/* Format the data to print nicely. */
DATA forprinting;
RETAIN TERMDESC Undergraduate UG_Nondegree Certificate Masters Specialist Doctoral GR_Nondegree Total ;
LABEL termdesc="Semester";
SET historic_project;
termdesc=CAT("&season." , " ", (substr(term,1,4)));
vline=CAT("&season." , " ", (&season_yr-1));
DROP term;
RUN;

/* Print a chart of historical enrollments compared to projections. */
PROC REPORT DATA=forprinting nowd;
COLUMNS termdesc Undergraduate UG_Nondegree Certificate Masters Specialist Doctoral GR_Nondegree Total ;
COMPUTE termdesc;
IF termdesc="(&season. &season_yr.)" THEN DO;
CALL DEFINE (_row_,'style','style={font_weight=bold font_style=italic}'); /* this makes the line with our
predicted values bold and italicized */
END;
ENDCOMP;
RUN;

/* Print line graphs of historical enrollment trends compared to projections, with vertical line marking where historical data
ends and projections begin. */
PROC SGPLOT DATA=forprinting;
YAXIS LABEL="Headcount" VALUES=(0,17000);
XAXIS DISPLAY=(NOLABEL);
SERIES X=termdesc Y=UG_Nondegree / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=Certificate / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=Masters / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=Specialist / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=Doctoral / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=UG_Nondegree / LINEATTRS=(thickness=2);
TITLE1 "&season Semester Projections Versus Historical Enrollment";
TITLE2 "((Undergraduates displayed separately))";
KEYLEGEND / LOCATION=OUTSIDE POSITION=top;
RUN;
PROC SGPLOT DATA=forprinting;
YAXIS LABEL="Headcount" VALUES=(0,17000);
XAXIS DISPLAY=(NOLABEL);
SERIES X=termdesc Y=Undergraduate / LINEATTRS=(thickness=2);
SERIES X=termdesc Y=UG_Nondegree / LINEATTRS=(thickness=2);
REFLINE vline/ AXIS=X TRANSPARENCY = 0.9 LINEATTRS=(thickness=5);
TITLE1 "&season Semester Projections Versus Historical Enrollment";
TITLE2 "((Undergraduates only))";
KEYLEGEND / LOCATION=OUTSIDE POSITION=top;
RUN;

/* THE END! */
References


