Paper 2320-2015

Implementing a Discrete Event Simulation Using the American Community Survey and SAS® University Edition

Michael C. Grierson

ABSTRACT

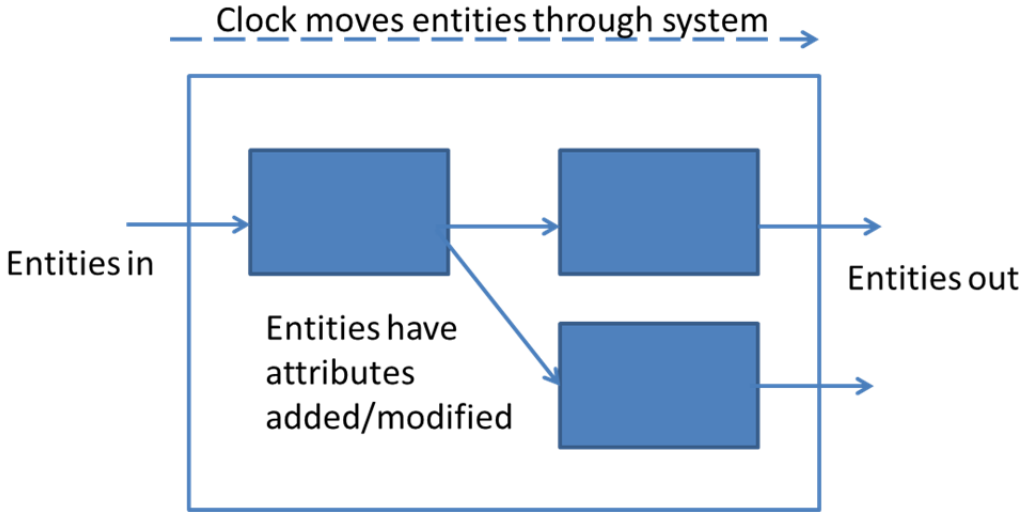
SAS® University Edition is a great addition to the world of freely available analytic software, and this 'how-to' presentation shows you how to implement a discrete event simulation using Base SAS® to model future US Veterans population distributions. Features include generating a slideshow using ODS output to PowerPoint.

INTRODUCTION

Discrete Event Simulation is the simulation of a real system at discrete moments in time. Between those discrete moments, much computing can occur. The idea is that by simulating the status of some phenomenon or system periodically, we can understand that phenomenon better and perhaps estimate future states of that system. In this paper, I address some concepts of using SAS® University Editions Base SAS® for discrete event simulations and then apply those concepts for simulating the future distributions by age of the United States Veterans Population. The reader should get an introductory understanding of Discrete Event Simulators. The reader will also get an understanding the SAS® University Edition's ability to use the American Community Survey PUMS data sets for getting insights about distinct American populations.

DISCRETE EVENT SIMULATION WITH BASE SAS®

A Discrete Event Simulator can be used to model a system that contains entities which in turn have attributes. For example a toy factory may be a 'system', and the factory may produce toys (entities) which have attributes (toy type, toy color, toy age). Another example would be the Department of Defense is a system that produces Veterans (entities) and those Veterans have attributes (like Veteran Age and Veteran gender). Entities flow through the system being modeled. Each entity has a starting point and most have an ending point in the model or system. Discrete Event models involve a sequence of discrete time periods so these models usually have a clock mechanism, which has a defined unit of time (hours, days, years). These concepts are represented in Figure 1 below.



Error! Reference source not found. **Discrete Event Simulator blocks**

Conveniently, entities can be represented by rows in a dataset and entity attributes are columns in datasets rows. You can use explicit output statements to conditionally send output to multiple blocks as the entities flow through the system. You can concatenate datasets during a single clock cycle to send multiple input streams into a block. Essentially, simulator blocks can be represented by datasets.

The clock cycle in a discrete event simulator can be represented by a macro call that passes in the time. The macro call would move all the entities between the blocks as appropriate for that model. In the model that will be used for this paper, the clock cycle is one year, so the macro that moves the entities between various blocks would be a macro that passes in the year value. Each year, the macro advances all the entities as appropriate for a clock cycle. Note that the macro calls would involve a sequence of years like below. In this example, a rental car store has a current inventory of cars, and each year, they buy more cars (new_cars) and they sell all of their cars that are older than three years of age. The attributes of the new_cars bought would shape the future of this rental car store and each year, you could pass in a varied set of new cars to project the future of this enterprise.

```
libname rentals '/folders/myfolders/RentalCars';

data current_cars;
   set sashelp.cars;
   car_year=2003 ;
run;

data new_cars;
   set sashelp.cars;
run;

%macro cycle(yr);
   data current_cars cars_to_sell;
      set current_cars new_cars;
      if car_year = . then car_year=&yr;
      if enter = . then enter = &yr;
      yeardiff=&yr-car_year;
      if yeardiff > 3 then do;
         output cars_to_sell;
      end;
      if yeardiff <= 3 then do;
         output current_cars;
      end;
   run;
%mend;

%cycle(2004);
%cycle(2005);
%cycle(2006);
%cycle(2007);
%cycle(2008);
%cycle(2009);
%cycle(2010);
%cycle(2011);
%cycle(2012);
```

The code above runs to a steady state number of cars for the rental car store. It's just an example of creating entities, adding them to the model, and removing entities based on criteria. Additional attributes of the new rental cars could be added (like resale value) that would make this model more useful. At this stage, it simply demonstrates movement of entities to datasets and a clock cycle (discrete event) implemented via the macro call cycle.

SAS® UNIVERSITY EDITION AND THE AMERICAN COMMUNITY SURVEY

SAS® University Edition provides a complete SAS® Programming Environment distributed as a Virtual Machine (either VMWare or Virtual Box). This Edition uses a SAS® Studio interface to a CentOS virtual guest machine. The guest machine is 'fixed' at 2 CPUs and supports a variable amount of RAM (it's set a 1 Gigabytes initially). The University Edition setinit is shown below:

```
Operating System:   LIN X64 .
 Product expiration dates:
 ---Base SAS Software
 ---SAS/STAT
 ---SAS/IML
 ---SAS/Secure 168-bit
 ---SAS/ACCESS Interface to PC Files
 ---SAS/ACCESS Interface to ODBC
 ---SAS/IML Studio
 ---SAS Workspace Server for Local Access
 ---SAS Workspace Server for Enterprise Access
---High Performance Suite
```

This SAS® installation includes software to read and analyze stratified survey data like the American Community Survey (ACS). The ACS is a product of the Census bureau and the Public Use Microdata Sample (PUMS) files are provided as SAS® datasets at ftp.census.gov. The 1 year ACS identifies about 3.5 million samples to represent the 300+ million Americans and their attributes. The ACS is a tremendous data resource for America and its communities. It's intended to help community planners of all kind with data to support resource decisions. The community of Veterans is one such community with some interesting attributes.

VETERANS DATA IN THE AMERICAN COMMUNITY SURVEY

The Questionnaire

The Veteran population in the US can be characterized using the American Community Survey based on the ACS questions 26 and 27. These questions and using SAS® to assess the Veteran population were addressed in an SAS® Tech Note that showed a Veteran population model using SAS® Simulation Studio (see the recommended reading list at the end of this paper). These two questions are shown below in figure 2.

Figure 2. ACS Questions used in this paper.

Question 26 can be used to determine if the citizen is a Veteran and Question 27 can be used to estimate when the person became a Veteran. These two questions are used in conjunction with all of the other citizen answers, like age. By determining the last period of military service, one can estimate at what age the person became a Veteran. These questions permit one to estimate the age attributes of the group of current veterans (current up to the year 2012 as we are using the ACS 2012 1 year sample) and the age distribution of new veterans.

Age Distributions estimated from ACS
Using the surveyfreq procedure you can determine and estimate of the age distribution of these two groups (new and current Veterans). Details of using the surveyfreq procedure this way are shown in the 2nd paper listed in the references documents. These age distributions are shown in Figure 3 below. Notice in these figures, probability distributions are used. This permits use of a more convenient number of entities (smaller than the actual populations may be convenient).

Figure 3. Age Distribution attribute of the Current and New Veteran Populations.

The code used to make the graphs above for the current Veterans is below.

```
%let odson=ods graphics on / width=800px height=600px imagefmt=png
imagemap=on imagename="vBarPlot" border=off; ;

%let odsoff=ods graphics on / reset=all imagename="MyResetplot";
libname pus2012 "/folders/myfolders/censusdata/2012";

data mf.freqoutCV;
  set pus2012.freqoutcv;
  pb = Percent/100;
run;

&odson;
proc sgplot data=mf.freqoutCV;
  title "Current Veterans";
  series x=agep y=pb ;
  xaxis label=&xal;
  yaxis label="Probability" max=0.04;
  footnote 'from the 2012 American Community Survey';
run;
&odsoff;
```

The ACS indicates about 22 million Veterans in 2012, but the simulation may run faster if you use only 2.2 million. You have to be sure to scale the new veteran population similarly (divide by 10 for example for both the new and the current populations). So, the challenge here is to create a set of entities with the characteristics of age that are distributed just like the current Veterans and the new Veterans. More generally, discrete event simulators often use distributions to describe sets of entities, distributions like the triangle, normal, or exponential distributions. The SAS® rand function supports many distributions, including distributions defined by a table of probabilities, like the Current Veteran and New Veteran distributions above.

Generating distributions for the Simulation of future Veteran Populations
Note that the pus2012.freqoutcv data set was created in the second paper in the references section. This dataset has a table of probabilities that can be used in the rand function to generate a similar distribution, but with a different number of entities. This is demonstrated in the code below. First you need to transpose the table of probabilities for use in an array as shown below:

```
proc transpose data=mf.freqoutCV out=mf.vetagepdf_trans(drop=_LABEL_
       rename=(_NAME_=var)) prefix=x;
  var agep percent pb;
```

This transpose gives you three rows and 80 columns (including one extra for a summary column). We are interested in the probabilities (pb) row and the first 79 columns for use in the rand function as show in the code below (Note that I simply cut and pasted the probabilities into the array statement):

```
%let N= 2200000;
data mf.CurrentGen(keep=x);
  call streaminit(3215);
  array p[79] _temporary_ (0.000057 0.000226 0.000585 0.001495 0.001964
    0.002686 0.003524 0.00425 0.005592 0.006175 0.006553 0.0073 0.007474
    0.007776 0.007957 0.007778 0.006833 0.006979 0.007792 0.007536 0.007653
```

```
    0.00805 0.008835 0.010932 0.011336 0.012419 0.012083 0.011974 0.013082
    0.012756 0.014264 0.015287 0.015392 0.016714 0.015429 0.014817 0.015652
    0.016245 0.016967 0.016689 0.016341 0.015962 0.017712 0.019692 0.021515
    0.025923 0.030602 0.033758 0.037967 0.02707 0.023553 0.023029 0.023335
    0.020031 0.018102 0.017247 0.018098 0.017139 0.016526 0.016478 0.016383
    0.017033 0.017713 0.017295 0.01569 0.014107 0.012599 0.012683 0.011943
    0.011209 0.009764 0.008518 0.006331 0.003533 0.000909 0.002626 0.008474
    0.0057 0.000299);
  do i = 1 to &N;
    x1 = rand("Table", of p[*]);
    x=x1+16;
    output;
  end;
run;
```

The result is a dataset called CurrentGen that contains the one variable for age (x). This column of generated age data has the same distribution as the Current Veterans data from the ACS, but this dataset only as 2.2 million rows (N in the code). Also note that we shifted all of the ages to the right by 16 years since the first age in the ACS derived distribution was 17. We apply the same technique to generate 18000 entities representing the age distribution of the New Veterans as derived from the ACS. The use of the number 180000/10 is described in the second reference.

The ACS indicates about 22 million Veterans in 2012, but the simulation may run faster if you use only 2.2 million. You have to be sure to scale the new veteran population similarly (divide by 10 for example for both the new and the current populations). So, the challenge here is to create a set of entities with the characteristics of age that are distributed just like the current Veterans and the new Veterans. More generally, discrete event simulators often use distributions to describe sets of entities, distributions like the triangle, normal, or exponential distributions. The SAS® rand function supports many distributions, including distributions defined by a table of probabilities, like the Current Veteran and New Veteran distributions above.

USING THE GENERATED DATA IN A SIMULATION

Now that we have a way to generate a finite number of entities for representing the current Veterans and an estimate representing the new Veterans, we are nearly ready to build a discrete event simulation. However, we need criteria for removing entities from the model. Much like the criteria for the rental cars (4 or more years old cars were sold off) aging from the model, we use the Social Security Administrations period of life tables to predict how many veterans were going to die. This is also described in the prior paper (the second reference) except for this paper we used the period of life tables for 2010. If we have criteria for entities entering the model initially and over time and criteria for entities leaving the model over time, we can write some data steps to implement the clock cycle, then call those data steps via a macro representing the clock cycle, just like with the rental cars simulation described earlier.

We generate 2.2 million current veteran entities (to represent 22 million actual veterans) and we generate 18000 new veteran entities every year (to represent 180000 actual new veterans or people leaving active duty military service). This process is shown in the block diagram in Figure 4 below.

A method to generate scaled distributions was shown earlier. A candidate clock cycle macro (called %macro cycle(yr)) is shown below. The mechanism suggested for advancing the clock or discrete events is shown below. This has three main sections and is only a small example of the power of base SAS® for this purpose.

The first section creates a dataset copy of the generated New Veterans for each year of the cycle. The first section also adds the attribute of the year that these entities entered the model.

The second section adds the new entities with their entry year attribute added to the entities already in the model. The second section also makes the entity one year older (x=x+1).

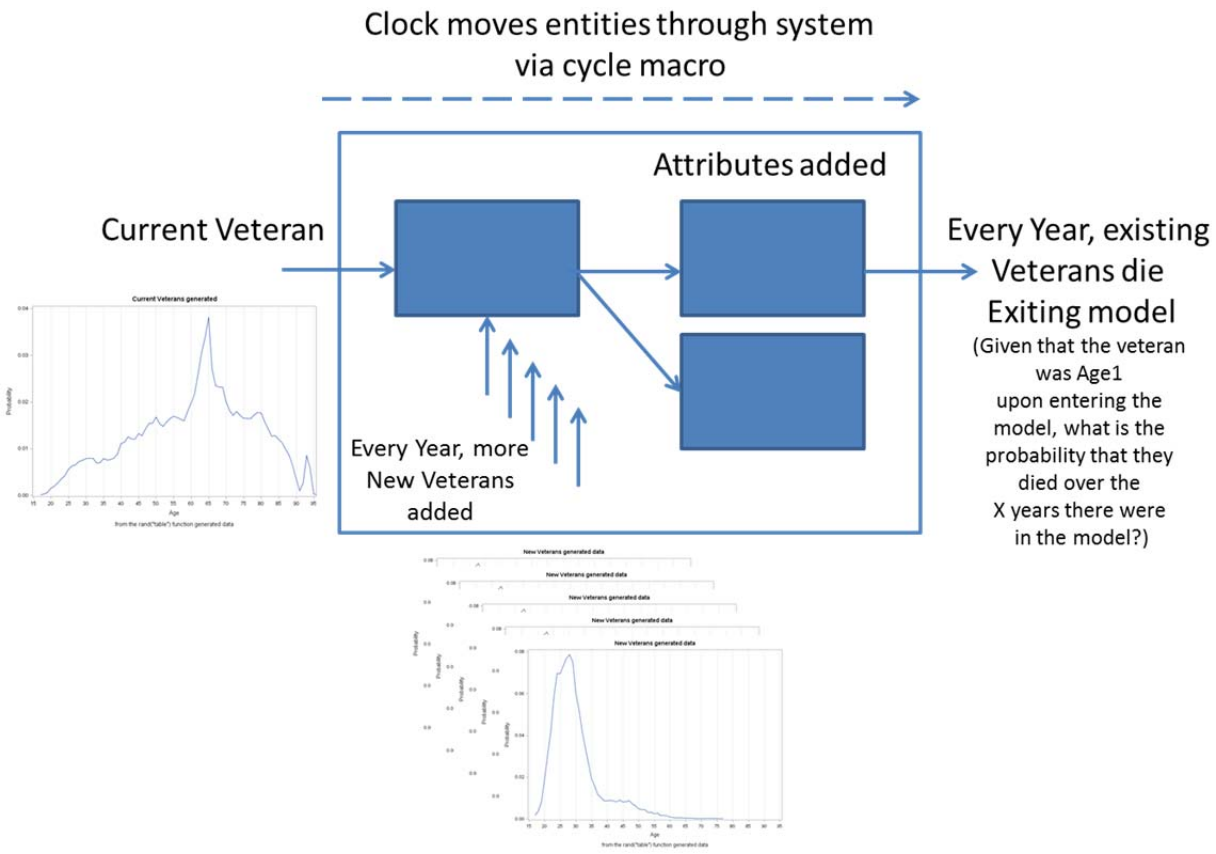


Figure 4. Block diagram of Veteran Population Model.

The third section essentially removes veterans from the model by applying the probability of death from the period of life tables. The SQL is used to group the entities in the model by current age and entry year. With the current age, the entry year and the current year of the model (the discrete event you are currently on), one can get the probability of death for these value pairs, and can reduce the counts of veterans based on the probability of death. The result is a dataset that has age and entity counts that have been adjusted to accommodate the Veterans leaving the model by what is essentially an actuarial calculation.

Notice that prior to the cycle macro, the code to define the 'getnol_macro' function. This is described in the second reference, but is included here to show that the Base SAS® capabilities far exceed that functionality available in a typical discrete event simulator.

```
options cmplib=work.ds;

%macro getnol;
%global RC;
%let RC=;
%let inreturned=;
proc sql noprint;
  select mnl into :inret
    from mf.lt where ea = &a ;
  select 1-mnl/&inret into :p
```

```
    from mf.lt where ea = &b ;
quit;
%mend getnol;
proc fcmp outlib = work.ds.functions;
function getnol_macro(a, b);
  rc = run_macro('getnol', a, b, p);
  if rc eq 0 then return(p);
    else
  return(.);
endsub;
run;

%macro cycle(yr);
  data mf.newgen&yr;
    set mf.newgen;
    entryyr=&yr.;
  run;
  data mf.currs;
     set mf.currs mf.newgen&yr. ;
     x = x + 1;
  run;
  proc sql;
    create table mf.alivescnt&yr as
      select x, entryyr, count(*) as cnt
        from mf.currs
        where entryyr le &yr
        group by x, entryyr;
  quit;
  data mf.alive&yr.;
    set mf.alivescnt&yr;
    inc=&yr-entryyr;
    pd=getnol_macro(x,x+inc);
    newcnt=round(cnt*(1-pd));
  run;
%mend cycle;
```

And the macro is simply called repeatedly. Note that it's very easy to make this macro take many more passed in variables. The number of new Veterans, which is constant at 18000 would be a good choice to make variable here by turning the code that generates the new veterans into a macro and simply calling it with a new N passed in. I did not do this below for this example.

```
%cycle(2013);
%cycle(2014);
%cycle(2015);
%cycle(2016);
%cycle(2017);
     *
     *
     *
     *
```

The result of the calls of the cycle() macro are a set of datasets for each discrete event (in this case year) called alives&yr, for example alives2020.

GRAPHICAL OUTPUT

Most discrete event simulation software packages provide output in a graphical form at various stages of the simulation. In this problem, I suggest using an ODS output technique that leverages the power of all of the Base SAS® graphing capabilities included in SAS® University Edition. By using the ODS output to PowerPoint features included with SAS® University edition, one can create a slideshow of discrete event results. .

The previous section described how to create the alives&yr (for example alives2020) datasets. This section will describe how to use those datasets to create a frequency distribution and output the related graph to a series of PowerPoint slides using ODS. First, you'll need to initialize ODS output. Below I create an initial slide with a title page for all the rest of the slides.

```
ods _all_ close;
title1;
footnote 'From the American Community Survey 2012 and the Social Security
Administrations Period of Life tables 2010';
ods escapechar = "^";
ods PowerPoint file="/folders/myfolders/pops4.pptx" style=PowerPointLight
layout=titleslide
    nogtitle nogfootnote;

proc odstext;
p "Population Distributions of Veterans" / style=presentationtitle;
p "Projected forward by using SAS University Edition" /
    style=presentationtitle2;
run;
```

Next a macro is called that will process each of the discrete events. Remember that your output to PowerPoint is still active. The macro below first stores the total count in a macro variable, then scales that total count by 10 (recall that we are using a 1/10th model), then manually creates the frequency distribution by using an SQL group by clause on the age (x). Then the 'alives(yr)' macro sets the image_dpi for PowerPoint output. The next PowerPoint slide had the project population total in the title followed by a call to proc sgplot to create the probability distribution graphic which is also output to the slide.

```
%macro alives(yr);

proc sql noprint;
  select sum(newcnt) into :tot from mf.alive&yr.;
  select sum(newcnt)*10 into :bt from mf.alive&yr;
  create table mf.alive&yr.sums as
  select x as age, sum(newcnt) as cnt, sum(newcnt)/&tot as pb
    from mf.alive&yr
    group by x
    order by x desc;
quit;

ods PowerPoint image_dpi=300 layout=_null_ ;

proc sgplot data=mf.alive&yr.sums;
  title3 " " ;
  title4 "population projected to be &bt for &yr" ;
  title5 " " ;
  series x=age y=pb ;
  xaxis label=&xal;
  yaxis label="Probability" min=0 max=0.04;
```

```
  footnote ' ';
run;
%mend alives;
```

A series of slides are created by calling this macro for every discrete event.

```
%alives(2013);
%alives(2014);
%alives(2015);
%alives(2016);
   *
   *
   *
```

Each discrete event ‘year’ has a slide created that looks like in the figure below. Because the slides are all created in sequence, a slide show can be used to show how this distribution changes over time. This slide show can be used to show how the age distribution of Veterans may change for future years to support resource planning (budgeting). A trend of the totals for the set of discrete events is shown in

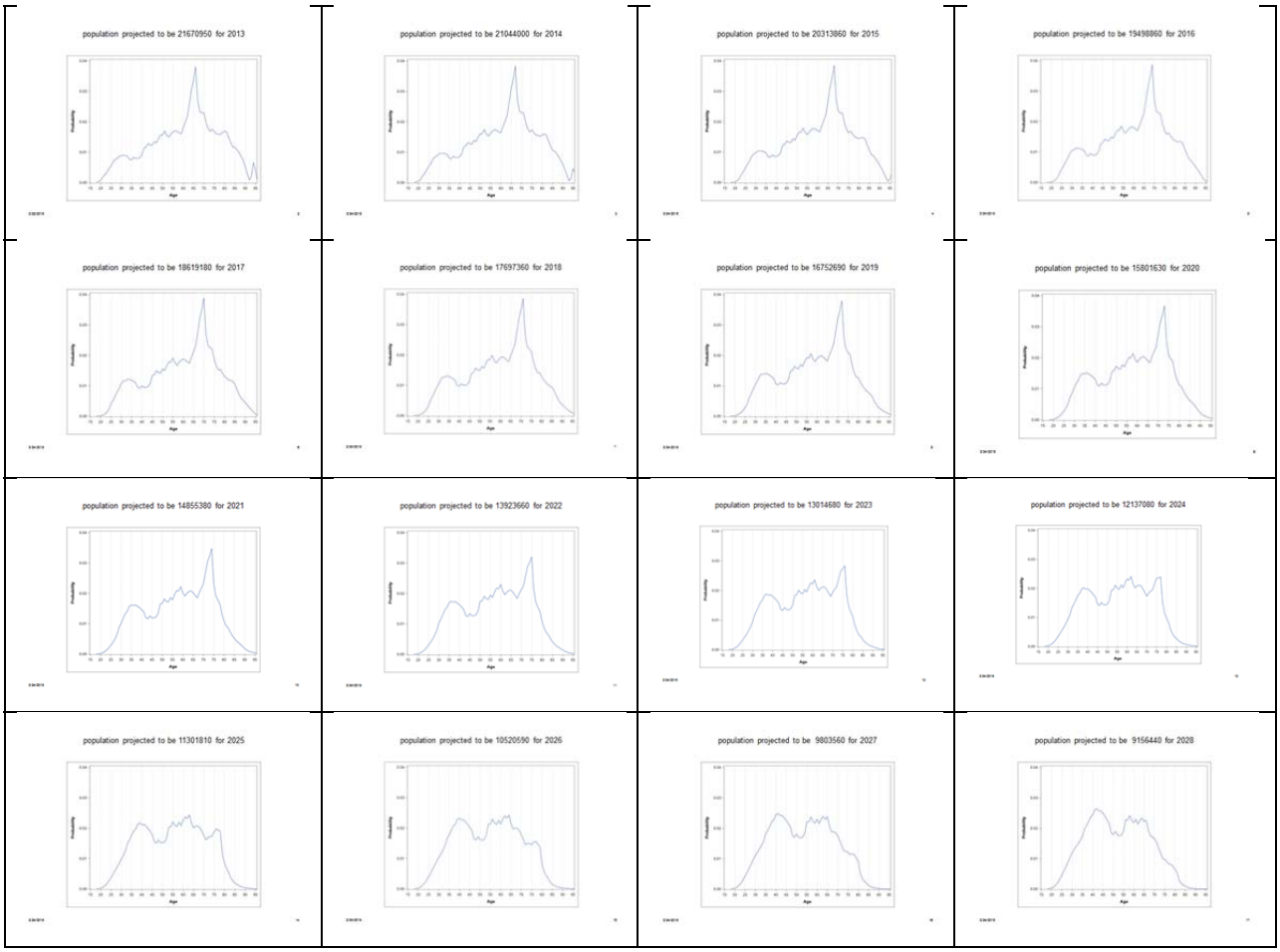


Figure 5. PowerPoint slide for year 2013 using ODS output.

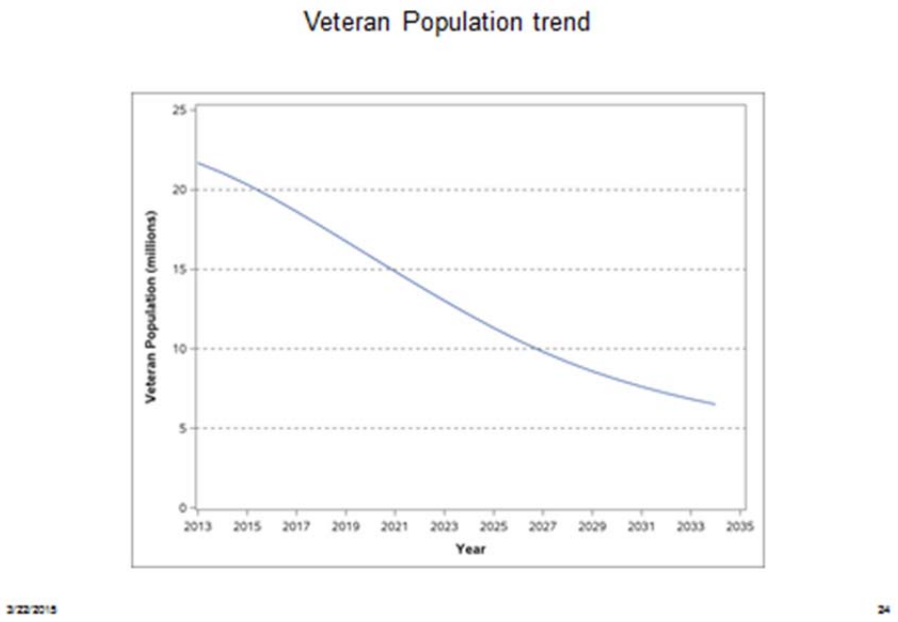


Figure 6. PowerPoint slide for the Veteran Population trend using ODS output.

SUMMARY

This paper describes a generalized approach for using Base SAS® to create a discrete event simulation. In particular, it describes the unique features of SAS® University Edition for processing data from stratified surveys like the American Community Survey. Since most national surveys are stratified, this is a great benefit to students. The paper also demonstrates how to create graphical output for your survey results using the very powerful ODS output system, also included with SAS® University Edition.

CONCLUSION

SAS® University edition can be used to teach some Operations Research courses and coupled with other freely available to students tools (like GLPK) can be used for most Operations Research coursework. Base SAS® has always been the most useful part of the SAS® suite of tools.

REFERENCES

- *The ACS Questionnaire for 2012 located here* *http://www.census.gov/acs/www/Downloads/questionnaires/2012/Quest12.pdf*
- *A paper showing how to use SAS® Simulation Studio to create a similar discrete event simulation* *http://www.sascommunity.org/wiki/Estimating_the_discrete_probability_distribution_of_the_age_characteristic_of_Veteran_populations_using_SAS%C2%AE,_SAS/OR%C2%AE_and_SAS_Simulation_Studio%C2%AE_for_use_in_population_projection_models*

RECOMMENDED READING

- *Base SAS® Procedures Guide, especially the sections on stratified surveys.*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mike Grierson
mcg@mcg-ct.com
http://www.mcg-ct.com/ or on the
http://www.sascommunity.org wiki