PharmaSUG 2018 - Paper AD-06

Building Simulated CRF Data for Study Mock-Ups

James Blum, University of North Carolina Wilmington;
Jonathan Duggins, North Carolina State University;
Ashley Kesler, PPD International;
Hisham Madi, Syneos Health

ABSTRACT

Simulation of mock trial data has wide-ranging utility in analysis plan development, data mapping and TLF programming, quality control for clinical trials, and development of employee training activities. Simulation strategies and data use will be covered in a variety of examples.

INTRODUCTION

Specialists in biometrics can use mock data as a basis for development of several steps in a clinical trial. The typical workflow is to design and build a database, enroll subjects and wait for data to accumulate, and once sufficient data is available, monitoring and programming activities begin. Identifying key components of the data handling and analysis processes, and their impact on study endpoints, sometimes occurs retroactively. An alternate approach would be to develop a project with a reasonable approximation of the data for the clinical trial from the outset. This preparatory work may shorten specification development and initial programming durations by emphasizing a more thorough review of project setup using a larger volume of realistic data rather than a few contrived data points. Earlier development of production-level work for the study allows the biostatistics team to provide sponsors and others a fully developed analysis plan and creates a framework for a more efficient and timely delivery of results.

This paper discusses how, and when, creation of mock data can be helpful in making these steps more robust and/or efficient, and covers how mock datasets can be used for other purposes, particularly training. Strategies for simulation of data are important, and potentially vary based on the type of trial data being created and the nature of the trial itself, so attention will be paid to good programming practices to help with these processes.

UTILITY OF MOCK STUDY DATA

Mock data can be useful at several phases of the process, possibly as early as raw data from the backend of case report forms (CRF or eCRF), and often from the CDISC models for study data tabulation (SDTM) or analysis data (ADaM). If the purpose of creating mock data is to build shells for tables, listings, and figures (TLFs) for the SAP, then simulating SDTM will be the most useful approach. If the goal were to build training materials for mapping to SDTM, then simulating raw CRF data would be necessary. One could always begin with mock CRF data and create the programs to map to SDTM and then through to ADaM, but this may not always be the most efficient, or even a necessary, approach.

The value of mock data as training material is probably the most obvious of all of its potential uses. While it is possible simply to use prior studies as a basis for training, there is a significant advantage to being able to insert errors into the data. While the errors will appear random to the user, they can be built with particular structures and known positions so that trainers can evaluate the programmer’s ability to build robust code and locate errors of particular types. Even when an organization believes its commonly used eCRF platforms are robust to certain types of errors, samples can still be built to train programmers for situations where less robust procedures are in use. Additionally, for trainees and experienced programmers alike, these activities allow for the development and testing of new methods and tools without affecting ongoing studies.

A less common view of the value of mock data is its potential in increasing efficiency, or that the value of the time invested in building such data at the front end of the study has the potential for finding and/or avoiding problems later in the study. One area in which this can be highlighted is in the creation of shells for TLFs.

Often a table, listing, or figure shell is created in some ad-hoc, manual fashion, designed to create something that will have an appropriate appearance. What is often left out in such a process is how readily such an item can be built programmatically—it is likely that many of us have experienced a case where a seemingly inconsequential addition to a shell has made it exponentially more difficult to program. Even if only a minor change is required to make it more practical to program, if the shell is already approved then a change process must be undertaken. When mock data is available, shells can be created (and validated) with programs, and all of those programs are in place and ready to implement when the actual data collection for the study is complete, or even before if, for example, interim analyses are required.

Extending this process permits an entire study to be programmed ahead of the study data collection itself, recognizing potential problems and suggesting changes in direction much earlier and making them more likely to be implemented. This also helps with the potentially short turn-around that is expected at the end of data collection—if the bulk of the study has been pre-programmed, much more about it will be known to the biostatistics team working on it and issues should be more easily addressed.

This does imply a substantial investment early in the study, so how to best minimize that investment (or whether it should be made at all) is an important consideration. Using data at your disposal to simplify the process of creating mock study data is always best, so for a study that is an extension or variation on one done previously, mock data, programs, and shells from them are probably not necessary as such information from the previous study can be used directly. For studies where the protocol and/or SAP are expected to be potentially quite unstable, advance investment in mocking the study may prove to be wasteful if those efforts will have to be altered and re-created several times.

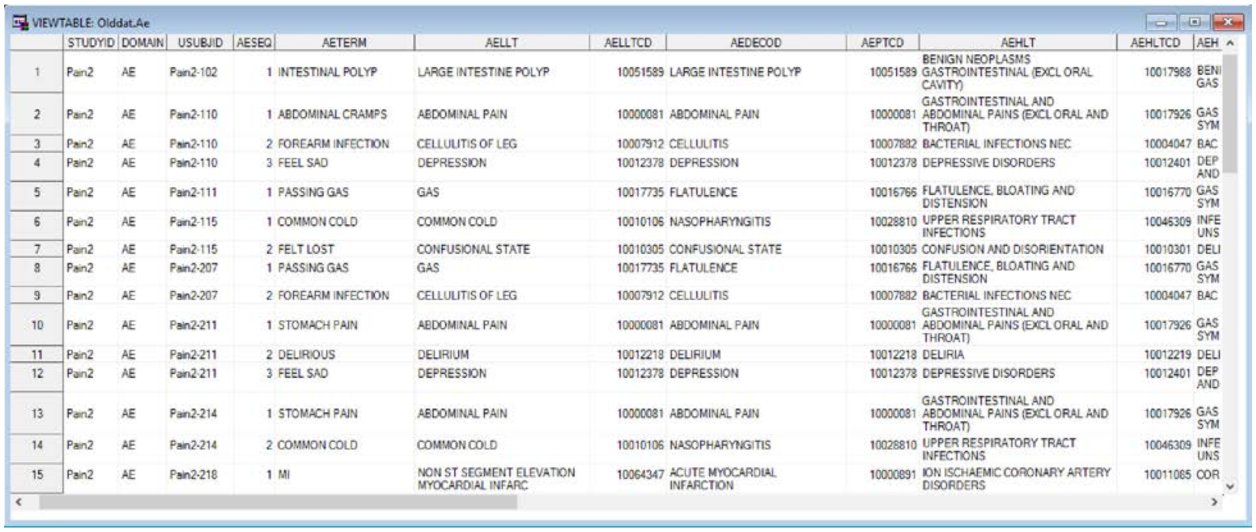
STRATEGIES FOR CREATING MOCK STUDY DATA

LOOK-UP TABLES

Given the complexity of study data of any type—raw CRF, SDTM, ADaM, or other—it is generally best to not have to generate all values via some random process. De-identified data from other studies is excellent for use in generating mock study data via random resampling of records, or groups of records, contained in it. Proper removal or anonymizing of information such as the sponsor or drug, or any other information that can identify a study must be ensured to all potential sponsors. Additionally, extra fields may need to be built in to control the resampling process. Finally, additional fields not present but needed for a full mock up can be added via some random process.

Example 1

Consider the task of making a simulated set of adverse events from an existing AE table as shown in Figure 1 below.



VIEWTABLE: Olddat.Ae

	STUDYID	DOMAIN	USUBJID	AESEQ	AETERM	AELLT	AELLTCD	AEDECOD	AEPTCD	AEHLT	AEHLTCD	AEH
1	Pain2	AE	Pain2-102	1	INTESTINAL POLYP	LARGE INTESTINE POLYP	10051589	LARGE INTESTINE POLYP	10051589	BENIGN NEOPLASMS GASTROINTESTINAL (EXCL ORAL CAVITY)	10017988	BENI GAS
2	Pain2	AE	Pain2-110	1	ABDOMINAL CRAMPS	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081	GASTROINTESTINAL AND ABDOMINAL PAINS (EXCL ORAL AND THROAT)	10017926	GAS SYM
3	Pain2	AE	Pain2-110	2	FOREARM INFECTION	CELLULITIS OF LEG	10007912	CELLULITIS	10007882	BACTERIAL INFECTIONS NEC	10004047	BAC
4	Pain2	AE	Pain2-110	3	FEEL SAD	DEPRESSION	10012378	DEPRESSION	10012378	DEPRESSIVE DISORDERS	10012401	DEP AND
5	Pain2	AE	Pain2-111	1	PASSING GAS	GAS	10017735	FLATULENCE	10016766	FLATULENCE, BLOATING AND DISTENSION	10016770	GAS SYM
6	Pain2	AE	Pain2-115	1	COMMON COLD	COMMON COLD	10010106	NASOPHARYNGITIS	10028810	UPPER RESPIRATORY TRACT INFECTIONS	10046309	INFE UNS
7	Pain2	AE	Pain2-115	2	FELT LOST	CONFUSIONAL STATE	10010305	CONFUSIONAL STATE	10010305	CONFUSION AND DISORIENTATION	10010301	DELI
8	Pain2	AE	Pain2-207	1	PASSING GAS	GAS	10017735	FLATULENCE	10016766	FLATULENCE, BLOATING AND DISTENSION	10016770	GAS SYM
9	Pain2	AE	Pain2-207	2	FOREARM INFECTION	CELLULITIS OF LEG	10007912	CELLULITIS	10007882	BACTERIAL INFECTIONS NEC	10004047	BAC
10	Pain2	AE	Pain2-211	1	STOMACH PAIN	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081	GASTROINTESTINAL AND ABDOMINAL PAINS (EXCL ORAL AND THROAT)	10017926	GAS SYM
11	Pain2	AE	Pain2-211	2	DELIRIOUS	DELIRIUM	10012218	DELIRIUM	10012218	DELIRIA	10012219	DELI
12	Pain2	AE	Pain2-211	3	FEEL SAD	DEPRESSION	10012378	DEPRESSION	10012378	DEPRESSIVE DISORDERS	10012401	DEP AND
13	Pain2	AE	Pain2-214	1	STOMACH PAIN	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081	GASTROINTESTINAL AND ABDOMINAL PAINS (EXCL ORAL AND THROAT)	10017926	GAS SYM
14	Pain2	AE	Pain2-214	2	COMMON COLD	COMMON COLD	10010106	NASOPHARYNGITIS	10028810	UPPER RESPIRATORY TRACT INFECTIONS	10046309	INFE UNS
15	Pain2	AE	Pain2-218	1	MI	NON ST SEGMENT ELEVATION MYOCARDIAL INFARC	10064347	ACUTE MYOCARDIAL INFARCTION	10000891	ION ISCHAEMIC CORONARY ARTERY DISORDERS	10011085	COR

Figure 1. AE Table for Look-Up, Partial Listing

A random re-sampling of rows, with replacement, allows for generation of a set of adverse event records of any desired size. It may also be preferred to re-sample the entire adverse event profile from prior subjects. It is possible to do this manually, using PROC SQL and/or the DATA Step, or by using PROC SURVEYSELECT. For manual selection, by row or by patient, additional identifiers will be required; for SURVEYSELECT an additional identifier will be required for selection of patient profiles.

MANUAL METHOD FOR LOOK-UP TABLE SELECTION

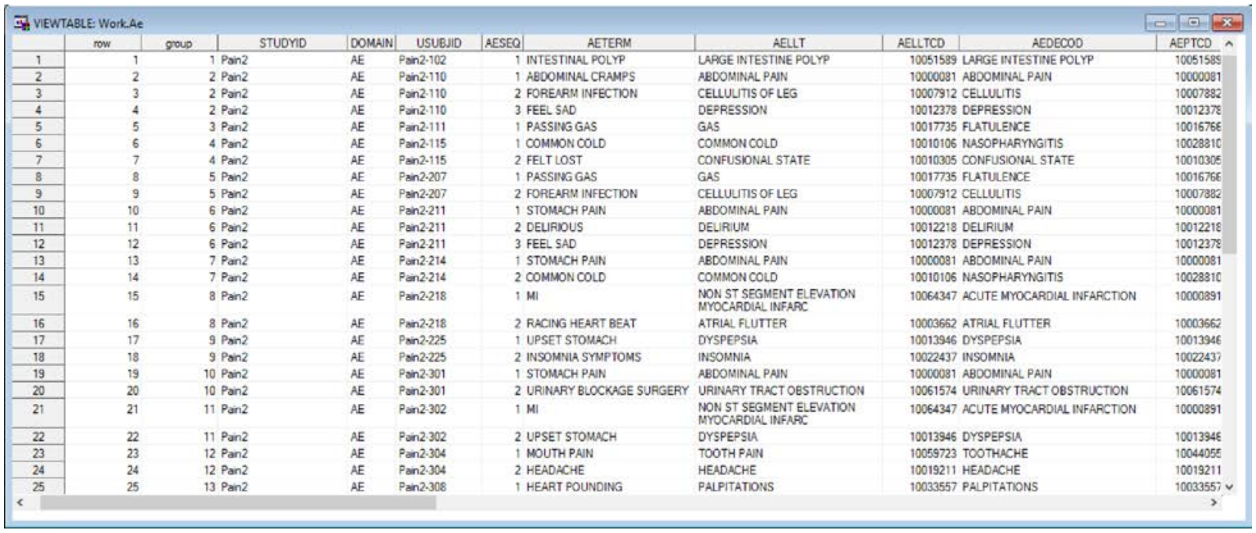
Consider the following data step that modifies the existing AE table for selection of rows or groups of rows corresponding to a subject.

Program 1: Data Step for Adding Record Identifiers

```
data ae;
 set olddat.ae;
 by usubjid;
 row = _n_;❶
 if first.usubjid then group+1;❷
run;
```

❶ The data step record counter is used to create a variable corresponding to the row number in the data set
❷ By group processing on the appropriate variable—usubjid in this case—is used to create a more basic subject numbering system

The resulting data, shown below in Figure 2, now has whole numbers corresponding to either of the units to be sampled.



VIEWTABLE: Work.Ae

	row	group	STUDYID	DOMAIN	USUBJID	AESEQ	AETERM	AELLT	AELLTCD	AEDECOD	AEPTCD
1	1	1	Pain2	AE	Pain2-102	1	INTESTINAL POLYP	LARGE INTESTINE POLYP	10051589	LARGE INTESTINE POLYP	10051589
2	2	2	Pain2	AE	Pain2-110	1	ABDOMINAL CRAMPS	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081
3	3	2	Pain2	AE	Pain2-110	2	FOREARM INFECTION	CELLULITIS OF LEG	10007912	CELLULITIS	10007882
4	4	2	Pain2	AE	Pain2-110	3	FEEL SAD	DEPRESSION	10012378	DEPRESSION	10012378
5	5	3	Pain2	AE	Pain2-111	1	PASSING GAS	GAS	10017735	FLATULENCE	10016766
6	6	4	Pain2	AE	Pain2-115	1	COMMON COLD	COMMON COLD	10010106	NASOPHARYNGITIS	10028810
7	7	4	Pain2	AE	Pain2-115	2	FELT LOST	CONFUSIONAL STATE	10010305	CONFUSIONAL STATE	10010305
8	8	5	Pain2	AE	Pain2-207	1	PASSING GAS	GAS	10017735	FLATULENCE	10016766
9	9	5	Pain2	AE	Pain2-207	2	FOREARM INFECTION	CELLULITIS OF LEG	10007912	CELLULITIS	10007882
10	10	6	Pain2	AE	Pain2-211	1	STOMACH PAIN	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081
11	11	6	Pain2	AE	Pain2-211	2	DELIRIOUS	DELIRIUM	10012218	DELIRIUM	10012218
12	12	6	Pain2	AE	Pain2-211	3	FEEL SAD	DEPRESSION	10012378	DEPRESSION	10012378
13	13	7	Pain2	AE	Pain2-214	1	STOMACH PAIN	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081
14	14	7	Pain2	AE	Pain2-214	2	COMMON COLD	COMMON COLD	10010106	NASOPHARYNGITIS	10028810
15	15	8	Pain2	AE	Pain2-218	1	MI	NON ST SEGMENT ELEVATION MYOCARDIAL INFARC	10064347	ACUTE MYOCARDIAL INFARCTION	10000891
16	16	8	Pain2	AE	Pain2-218	2	RACING HEART BEAT	ATRIAL FLUTTER	10003662	ATRIAL FLUTTER	10003662
17	17	9	Pain2	AE	Pain2-225	1	UPSET STOMACH	DYSPEPSIA	10013946	DYSPEPSIA	10013946
18	18	9	Pain2	AE	Pain2-225	2	INSOMNIA SYMPTOMS	INSOMNIA	10022437	INSOMNIA	10022437
19	19	10	Pain2	AE	Pain2-301	1	STOMACH PAIN	ABDOMINAL PAIN	10000081	ABDOMINAL PAIN	10000081
20	20	10	Pain2	AE	Pain2-301	2	URINARY BLOCKAGE SURGERY	URINARY TRACT OBSTRUCTION	10061574	URINARY TRACT OBSTRUCTION	10061574
21	21	11	Pain2	AE	Pain2-302	1	MI	NON ST SEGMENT ELEVATION MYOCARDIAL INFARC	10064347	ACUTE MYOCARDIAL INFARCTION	10000891
22	22	11	Pain2	AE	Pain2-302	2	UPSET STOMACH	DYSPEPSIA	10013946	DYSPEPSIA	10013946
23	23	12	Pain2	AE	Pain2-304	1	MOUTH PAIN	TOOTH PAIN	10059723	TOOTHACHE	10044055
24	24	12	Pain2	AE	Pain2-304	2	HEADACHE	HEADACHE	10019211	HEADACHE	10019211
25	25	13	Pain2	AE	Pain2-308	1	HEART POUNDING	PALPITATIONS	10033557	PALPITATIONS	10033557

Figure 2. AE Table for Look-Up, Partial Listing

Generating a sample is now achieved by generating a set of random whole numbers in the appropriate range. Generally, it will be advantageous to determine the range of values first, generate a set of random numbers from that set, and select those rows. Program 2 illustrates a way to achieve this for either strategy, the Subjects data set is presumed to be a single column of dummied subject numbers.

Program 2: Resampling Rows or Groups of Rows

```
proc sql;
 select max(row) into :rows from ae;❶
 select max(group) into :groups from ae;❶
quit;

data rows;
 set subjects;
 num=ceil(ranuni(0)*3);❷
 if ranuni(0) le .20 then do;❸
  do j=1 to num;
   row=ceil(ranuni(0)*&rows);❹
    output;
  end;
 end;
 else delete;
run;

data groups;
 set subjects;
 if ranuni(0) le .20 then group=ceil(ranuni(0)*&groups);❺
  else delete;
run;

proc sql;
 create table ae_mock as select *
 from olddat.ae join rows on ae.row eq rows.row;❻

 create table ae_mock2 as select *
 from olddat.ae join groups on ae.group eq groups.group;❻
quit;
```

❶ Using the INTO clause the largest value of either identifying variable can be stored in a macro variable
❷ RANUNI is uniform on (0,1), so multiplying by a whole number, N, will be uniform on (0,N). Using the ceiling function on this makes it discrete uniform on 1, 2, 3, …, N. Here this is used to determine the number of AEs a subject will get.
❸ Here a 20% probability of getting AEs is set, other subjects who do not have AEs will not be included in this mock table.
❹ A set of row numbers are now randomly chosen for the given subject to receive the simulated number of AEs
❺ Generating groups of records corresponding to a subject profile is somewhat more straightforward than the individual row method
❻ Regardless of method, the final step is to use the selected row or group numbers as an index to join on AE information from the look-up table

Using PROC SURVEYSELECT can potentially make the resampling a bit easier, two versions of SURVEYSELECT are given in Program 3, one for selecting rows and another for selecting groups.

Program 3: Using SURVEYSELECT to Resample

```
proc surveyselect data=ae out=ae_mock_a method=urs❶ sampsize❷=20 outhits❸;
run;

proc surveyselect data=ae out=ae_mock_b method=urs sampsize=8 outhits;
 samplingunit group;❹
run;
```

❶ URS provides unrestricted sampling, or sampling with replacement
❷ SAMPSIZE= sets the number of units that will be sampled, based on the sampling unit specified. SAMPRATE= could also be used; however, it is based on the size of the data set sampled from, not the sampling unit.
❸ By default, SURVEYSELECT outputs each record selected with a variable that indicates how many times it was sampled, OUTHITS puts out a copy of the record for each time it is sampled.
❹ The SAMPLINGUNIT statement allows for a variable to be used to set up groups of records as a sampling unit, which works well for the patient profile method.

At this point, there is still some work to do create random subject numbers to join to this data (if desired), which could be achieved by running SURVEYSELECT on the set of mock subject identifiers, using the same sample sizes.

For either of these approaches, it would be possible to extend these principles to the entire set of SDTM domains and resample entire patient profiles from a previous study or an aggregate of several previous studies.

DIRECT SIMULATION OF VALUES

While look-up tables are very useful in generating mock study data, particularly for qualitative data with a large set of verbose categories, there is often a need to simulate individual values directly. Since many possible methods could be employed, it is useful to adopt strategies that are flexible enough to use in a wide variety of situations. The following is a discussion of simulating eCRF data at the individual record level, but these strategies can be utilized any time random generation of quantitative or simple categorical variables is required.

Program 4: Simulation of Collection of Demographic Data

```
%let seed=0;
data init_visit;
```

```
 length sf_reas $50;
 do site=1 to 4;
  num_site=floor(31*ranuni(&seed)+20);
   do vis=1 to num_site;
      screen=1; sf_reas=''; notif_date=.;❶
      dov=floor(90*ranuni(&seed)+'15JAN2016'd);❷
      age=round((74-18)*rantri(&seed,(45-18)/(74-18))+18,1)*365;❸
      dob=dov-age;

      if yrdif(dob,dov,'ACT') lt 21 or yrdif(dob,dov,'ACT') gt 69 then do;
        screen=0; sf_reas='AGE OUT OF RANGE'; notif_date=dov;
      end;❹

      g=round(ranuni(&seed),1);
      if g=1 then sex='F'; else sex='M';❺

      rand_prg=ranuni(&seed);
      if g=1 and age lt 45*365 and rand_prg le 0.08 then do;❻
        screen=0; sf_reas='PREGNANT'; notif_date=dov;
      end;

       rand_ic=ranuni(&seed);
       if rand_ic lt 0.025 then do;
        ic='N'; screen=0; sf_reas='DECLINED CONSENT'; notif_date=dov;
       end;
       else ic='Y';❼
   output;
  end;
 end;
 format dob dov notif_date mmddyy10.;
 drop num_site g rand:;❽
run;

proc sort data= init_visit;
 by site dov;
run;

data init_visit;
 retain site subject;
 set init_visit;
 by site;
 if first.site then subject=0;
 subject+1;
 drop vis;
run;❾
```

❶ Initialize some variables for later use or possible changes—here the screening flag is 1 until a simulated failure occurs, at which point the reason and notification date variables will be populated.
❷ This sets up a random recruiting visit date, uniform over 90 days starting on January 15th, 2016. In general it is best to pass specific parameters as macro variables and create modules for each simulated component for use with %INCLUDE.
❸ Subject age is generated by a random triangular distribution on the interval 18 to 74 with a mode of 45, in days. The triangular distribution is advocated because it is a relatively simple unimodal distribution, whose support and mode can be easily set. In general, a triangular distribution on (A,B) with mode M can be simulated with:

(B-A)*rantri(&seed,(M-A)/(B-A))+A

❹ The age range for participants in this study was set to be 21 to 69, inclusive. While it would have been

possible to simulate only values in this range, the distribution described above was chosen to intentionally include the potential for screening failures. The screen failure variables are updated for those simulated cases that fail. Various levels of realism in simulation can be employed; more realism does generally imply more complexity.
❺ Sex is simulated as having an equal distribution between males and females here. The round function is not necessary, but is shown to illustrate that the value of *g* could also be used if a 0/1 variable was sufficient.
❻ For a pregnancy flag, an extra attempt at realistic results is made by limiting gender and age in addition to the pregnancy rate. In certain instances, it may be decided not to do this to introduce intentional errors, but a controlled rate of such errors should probably be included.
❼ Informed consent is also simulated; lack of informed consent not only sets or resets values for three of the screening variables, but also establishes the value for its own flag. Here care is taken to ensure values are set whenever they should be and nothing is accidentally left missing.
❽ Some variables that aid in the simulation are removed; choosing the random number generators with a common prefix allows this set to be easily removed.
❾ Post processing of the simulated demographic table allows for a realistic subject numbering to be created.

To continue with this example, a baseline visit for collection of vital signs and lab panels is simulated, with the vital signs simulation shown in Program 5.

Program 5: Simulation of Vital Signs

```
data baseline_vs;
  set init_visit (where=(screen=1) rename=(dov=demov)
                  keep=site subject dov screen sf_reas sex age);
   dov=demov+ceil(6*ranuni(&seed)+2);❶

   sbp=round((150-90)*rantri(&seed,(100-90)/(150-90))+90,1);
   dbp=round((95-60)*rantri(&seed,(72-60)/(95-60))+60,1);❷

  if sbp gt 140 or dbp gt 90 then do;
   screen=0; sf_reas='HIGH BLOOD PRESSURE'; notif_date=dov;
  end;❸ bp_units='mm/hg';❹

  pulse=round(dbp+10*(ranuni(&seed)-0.5),1); p_units='beats/min';
  temp=round((100-97)*rantri(&seed,(98.5-97)/(100-97))+97,.1 );
  temp_units='F';

  if sex='F' then
         weight=round((200-90)*rantri(&seed,(145-90)/(200-90))+90,1);
    else weight=round((280-140)*rantri(&seed,(200-140)/(280-140))+140,1);
  weight_units='lb';

 rand_pos=ranuni(&seed);
  if rand_pos lt .25 then do;
   rand_error=ranuni(&seed);
   if rand_error lt 0.5 then pos='RECUMBENT';
    else if rand_error lt 0.8 then pos='RECUMBANT';
     else pos='RECLINED';
  end;
   else do;
    rand_error=ranuni(&seed);
    if rand_error lt 0.5 then pos='SITTING';
     else if rand_error lt 0.8 then pos='SEATED';
      else pos='RECLINED';
   end;❺

  if screen eq 1 then do;
   rand_pain=ranuni(&seed);
   if rand_pain le .65 then pain=3;
    else if rand_pain le .95 then pain=2;
     else pain=1;
   if pain eq 1 then do;
    screen=0; sf_reas='LOW BASELINE PAIN'; notif_date=dov;
   end;
  end;❻

  drop rand: demov;
  format dov notif_date mmddyy10.;
run;
```

❶ Simulate baseline visit day, 2 to 8 days after demographic collection—for the sake of realism, if needed, interdependencies between simulated values can be built.
❷ Again, the triangular distribution can prove useful for quantitative values. Mixtures of triangular distributions can be used for multi-modal cases, mixtures of triangular and uniform distributions can be used to make heavier tails.
❸ In this case the blood pressure values are simulated to create possible screen failures and the appropriate variables are updated here when that occurs, similar to the methodology of the previous program.
❹ Units are hard-coded in this program; however, this is clearly a case it would be advantageous to have a prepared table to join to the simulated values.
❺ This is an example of a deliberate attempt to simulate a case where the permitted input into a CRF was not well controlled and values that are entered as distinct are not truly different.
❻ The simulation was for a study on a basic pain reliever, with a three-point ordinal scale for pain. Low baseline pain was also a screening criterion, so failures could be simulated here as well and should be properly set when they occur.

While a variety of other data sources would be simulated for a case like this one, the structure would be similar. Additionally, randomization and data from subsequent visits will be generated, and Program 6 shows ways in which this simulated data can be built from the simulated data for previous visits.

Program 6: Continuing the Simulation

```
data rand(keep=site subject dov arm) dose1(keep=site subject dov dose)❶;
 set baseline_lab(where=(screen=1) keep=site subject dov screen)❷;
    rand_arm=ranuni(&seed);
    if rand_arm le .25 then do; arm=1; start_trt='T'; end;
      else if rand_arm le .5 then do; arm=2; start_trt='R'; end;
        else if rand_arm le .75 then do; arm=3;start_trt='R'; end;
          else do; arm=4; start_trt='T'; end;
   dose=1;
   drop rand_arm;
   label arm='Study Arm Code' start_trt='Next Treatment Element Assigned';
run;

data visit_3mo;
  merge rand(keep=site subject arm in=in1)
        baseline_vs(where=(screen=1) in=in2)❸;
  by site subject;
  if in1 and in2;

  dov=dov+ceil(90+8*(ranuni(&seed)-0.5));❹
  sbp=sbp+ceil(6*(ranuni(&seed)-0.5));
  dbp=dbp+ceil(2*(ranuni(&seed)-0.5));❺

  if sbp gt 140 or dbp gt 90 then do;
   screen=0; sf_reas='DROPPED: HBP'; notif_date=dov;
  end;❺
  /**...some similar simulations omitted...**/

  if screen eq 1 then do;
   rand_pain=ranuni(&seed);
   select(arm);
     when(1,4) do;
      if rand_pain le .2 then pain=pain-1;
       else if rand_pain ge .97 then pain=pain+1;
        else pain=pain;
        start_trt='R';
     end;
     when(2,3) do;
      if rand_pain le .1 then pain=pain-1;
       else if rand_pain ge .9 then pain=pain+1;
        else pain=pain;
        start_trt='T';
     end;❻
   end;
   if pain lt 0 then pain=0;
   if pain gt 3 then pain=3;❼
  end;

  drop rand:;
  format dov notif_date mmddyy10.;
  label arm='Study Arm Code' start_trt='Next Treatment Element Assigned';
run;
```

❶ In this situation, it was decided that building raw data for randomization and initial dosing could be simulated in the same data step, keeping separate variable sets in each output data set.
❷ The simulation of lab data (not shown) followed vital signs; therefore, simulation of the randomization is built from simulated subjects who have passed screening at all preceding steps.
❸ Vital signs and other information (including labs, again not shown) for the next visit will be built for patients still on the study, and related to their values from the first visit.
❹ The visit date is 90 days after the original visit, on average, with some potential random deviation.
❺ New values are simulated as perturbations of prior values, and are checked if they correspond to inclusion/exclusion criterion.
❻ New values for the endpoint are simulated as well, directly related to prior values—here the simulation gives a greater probability to the test product being effective. At this point, the next treatment is also assigned.
❼ The nature of the simulation of this endpoint can cause the value to go out of bounds. It may be desirable in certain circumstances to allow this for assessing quality checks and/or defensive programming; here the simulation does not permit this problem to occur.

While the techniques here are used to simulate raw data, they can readily be adapted to simulation of SDTM or other data. In addition, the look-up table methods of the previous section can be used to simulate raw data as well. Of course, many attempts at simulating trial data will be well served by a combination of these ideas.

CONCLUSION

The programs presented here give a basic view of what is possible, but are not sufficiently dynamic for a production environment. A significant investment in simulation tools—macros and smaller modules for certain tasks, data tables for establishing simulation parameters, libraries of look-up tables, and other re-usable items—is necessary to consistently and effectively put these ideas into practice. This level of investment to create mock data sets for planning and development of clinical trial analyses can seem daunting, so careful consideration should be given to how and when these ideas are employed; however, the long-term benefits are well worth the investment.

CONTACT INFORMATION

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

James Blum
UNC Wilmington
blumj@uncw.edu