PharmaSUG 2015 - Paper QT24

Reproducibly Random Values

William Garner, Gilead Sciences, Inc., Foster City, CA
Ting Bai, Gilead Sciences, Inc., Foster City, CA

ABSTRACT

For questionnaire data, multiple responses may be provided. However, a single value, chosen at random, is required for analysis. We propose a method to randomly select between the multiple responses which yields the same value at a subsequent analysis, when additional data has been collected.

INTRODUCTION

In HIV clinical trials, subjects come in for assessments at various visits (eg Week 24, Week 48, Week 96) and analyses are carried out once all subjects hit a visit of interest, such as those specified above. Questionnaire data is one of many sources of data captured. On a questionnaire, the subject has the option to select multiple responses. For analysis purposes, though, a single value is required. Thus, a method for handling such data is needed. Moreover, this method needs to yield reproducible results at subsequent analyses.

MOTIVATING EXAMPLE

A Phase 3 trial with a sample size of around 400 subjects collected questionnaire data at 10 visits, leading to over 100,000 records. Subjects were allowed to provide multiple responses to a question, and approximately 6,000 records had more than one response marked.

The study contained an interim analysis (when the last subject enrolled reached Week 48) and a final analysis when the last subject reached Week 96. Thus, the same subjects at both the interim and final analysis, with the difference between additional longitudinal data.

Three possible ways to handle questions with multiple responses are as follows:

1. Set the response to missing for that question.
2. Take the worst (typically largest) of the values selected.
3. Choose one of the responses at random.

Option (1) is not preferred, since 6% of the data contains multiple responses, we may lose valuable information and come to an incorrect conclusion regarding treatment differences.

Option (2) is better, but may introduce bias if there is significant imbalance between treatment groups. We want to specify a method that is robust to these imbalances.

Option (3) seems to be the most reasonable approach to take. We impose the following rule before we begin imputing the data:

> "If the subject marks two adjacent responses (eg 1 2), then one of the responses will be chosen at random. If more than two responses are marked (eg 1 2 3) or two non-adjacent responses are selected (eg 1 3), the response for that question will be set to missing."

The reason for this is trying to understand the subject's intent. (To minimize bias, we do not query questionnaire data.) If there are two adjacent responses, it would seem that the subject wanted to select something in the middle. When more than two responses or two non-adjacent responses are given, the subject's intent is no longer discernible.

A SIMPLE EXAMPLE

From the motivating example above, we have decided to choose a response at random in the case of multiple responses. However, the data is rather complex, as we have hundreds of subjects with ten visits, with multiple questionnaires. This will be difficult to follow, so we illustrate the problem and our proposed solution using a simpler example.

Suppose we administer a 4-item questionnaire at 3 timepoints, baseline, week 24, and week 48. Each question has 5 possible responses ranging from 0 (Very Good) to 4 (Very Bad). Subjects are allowed to mark multiple responses. As stated above, we will only impute values if two adjacent responses are marked.

Table 1, below, shows data collected for 3 subjects at baseline and week 24.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)
1	1001	Baseline	1	4	13	1002	Week 24	1	1
2	1001	Baseline	2	1 2	14	1002	Week 24	2	2
3	1001	Baseline	3	0	15	1002	Week 24	3	0
4	1001	Baseline	4	0	16	1002	Week 24	4	0
5	1001	Week 24	1	0 4	17	1003	Baseline	1	2 3
6	1001	Week 24	2	2	18	1003	Baseline	2	1 2 3 4
7	1001	Week 24	3	1	19	1003	Baseline	3	4
8	1001	Week 24	4	0	20	1003	Baseline	4	3
9	1002	Baseline	1	0 1	21	1003	Week 24	1	0
10	1002	Baseline	2	3	22	1003	Week 24	2	1
11	1002	Baseline	3	0	23	1003	Week 24	3	2
12	1002	Baseline	4	1	24	1003	Week 24	4	1

Table 1. Questionnaire Data at the time of the first analysis

Based on the imputation rule, observation #5 and #18 will be set to missing. However, we need to impute values for observations #2, #9, and #17.

Suppose we set a random seed and assign a numerical analysis value at random for adjacent responses. This can be done via the following code:

```
data example_wk24_aval;
  set example wk24;
  call streaminit(1234);
  if length(strip(avalc)) = 1 then aval = input(avalc,1.);
  else if length(strip(avalc)) = 3 then do;
    val1 = input(scan(avalc,1," "),1.);
       val2 = input(scan(avalc,2," "),1.);
    if abs(val2-val1) = 1 then do;
         x = rand('UNIFORM');
         if 0 <= x <= 0.5 then aval = val1;
         else aval = val2;
       end;
  end;
  else if length(strip(avalc)) > 3 then aval = .;
  drop val1 val2;
run;
```

The result for observations #2, #9, and #17 is as follows:

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
2	1001	Baseline	2	1 2	2	0.7566113088
9	1002	Baseline	1	0 1	1	0.740637776
17	1003	Baseline	1	2 3	2	0.2975459404

Table 2. Imputed Values for Questionnaire Data at the time of the first analysis

The real work is done if the length of AVALC is 3. If the difference is equal to 1 (that is, there are two adjacent responses), we will randomly select a value after obtaining a random value from a uniform distribution.

Suppose we continue to collect data and conduct a second analysis at week 48. Each subject now has 4 additional records, corresponding to the Week 48 analysis window, highlighted in yellow. The observation numbers in Table 3 have been modified (shifted by 100) to help differentiate between observations at the Week 24 analysis and the Week 48 analysis.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)
101	1001	Baseline	1	4	119	1002	Week 24	3	0
102	1001	Baseline	2	1 2	120	1002	Week 24	4	0
103	1001	Baseline	3	0	*121*	*1002*	*Week 48*	*1*	*0*
104	1001	Baseline	4	0	*122*	*1002*	*Week 48*	*2*	*1 2*
105	1001	Week 24	1	0 4	*123*	*1002*	*Week 48*	*3*	*1 3*
106	1001	Week 24	2	2	*124*	*1002*	*Week 48*	*4*	*1*
107	1001	Week 24	3	1	125	1003	Baseline	1	2 3
108	1001	Week 24	4	0	126	1003	Baseline	2	1 2 3 4
109	*1001*	*Week 48*	*1*	*3 4*	127	1003	Baseline	3	4
110	*1001*	*Week 48*	*2*	*3*	128	1003	Baseline	4	3
111	*1001*	*Week 48*	*3*	*1*	129	1003	Week 24	1	0
112	*1001*	*Week 48*	*4*	*0*	130	1003	Week 24	2	1
113	1002	Baseline	1	0 1	131	1003	Week 24	3	2
114	1002	Baseline	2	3	132	1003	Week 24	4	1
115	1002	Baseline	3	0	*133*	*1003*	*Week 48*	*1*	*0*
116	1002	Baseline	4	1	*134*	*1003*	*Week 48*	*2*	*1*
117	1002	Week 24	1	1	*135*	*1003*	*Week 48*	*3*	*1*
118	1002	Week 24	2	2	*136*	*1003*	*Week 48*	*4*	*2*

Table 3. Questionnaire Data at the time of the second analysis

This time, we need to impute values for observations #102, #109, #113, #122, and #125. We note that #102, #113, and #125 correspond with observations #2, #9, and #17, respectively, in Table 2.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
102	1001	Baseline	2	1 2	2	0.7566113088
109	1001	Week 48	1	3 4	4	0.740637776
113	1002	Baseline	1	0 1	0	0.2975459404
122	1002	Week 48	1	1 2	1	0.2951936834
125	1003	Baseline	1	2 3	3	0.9959194604

Table 4. Imputed Values for Questionnaire Data at the time of the second analysis

Recall, in Table 2, observation #9 was imputed as 1. Here, observation #113 is imputed as 0. Similarly, observation #17 was imputed as 2 before, but now observation #125 is imputed as 3.

The seed values are the same for the first three records in Table 4. This is due to the initialization of the stream via, "`call streaminit(1234);`"

Recall, our goal was to reproduce the random values that were assigned at the first analysis. This method has failed to do so. The seed is set once and the insertion of additional data has altered the assignment of random values, and thus the selection of an analysis value. Thus, we must seek an alternative method.

A POSSIBLE SOLUTION

One way to ensure consistency with the first analysis would be to load the dataset from the first analysis and then merge in the previously assigned values. Afterwards, we only impute values that are missing.

We ensure that values are not overwritten by including "`and aval = .`" in the code (highlighted in yellow).

```
/* Merge in the imputed values from Week 24 */
data example_wk48_with_wk24;
  merge example wk48(in=a) example_wk24_aval(in=b drop=obs x);
  by subjid avisit qnum;
run;
/* Now, impute values *only* if there is no value currently assigned (ie, previously
decided) */
data example_wk48_with_wk24_aval;
```

```
  set example wk48 with wk24;
  call streaminit(1234);
  if length(strip(avalc)) = 1 then aval = input(avalc,1.);
  else if length(strip(avalc)) = 3 and aval = . then do;
    val1 = input(scan(avalc,1," "),1.);
    val2 = input(scan(avalc,2," "),1.);
    if abs(val2-val1) = 1 then do;
         x = rand('UNIFORM');
         if 0 <= x <= 0.5 then aval = val1;
         else aval = val2;
       end;
  end;
  else if length(strip(avalc)) > 3 then aval = .;
  label aval = "Analysis Value (N)"
        x = "Random Value";
  drop val1 val2;
run;
```

By performing the merge, we would bring in values for observation #102, #113, and #125. They would be 2, 1, and 2, respectively. Only 2 records, observation #109 and #122 need to be imputed.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
109	1001	Week 48	1	3 4	4	0.7566113088
122	1002	Week 48	1	1 2	2	0.740637776

Table 5. Alternative Imputed Values for Questionnaire Data at the time of the second analysis

One challenge with this method is that one must know what the most recent dataset is when carrying out the merge. Typically, for one analysis timepoint, multiple analyses may be carried out. At the time of each analysis, additional data may be added. As a result, it is critical to keep updating the location of the data. This could easily lead to a mistake and thus inconsistent values may appear between analyses. Another approach, which does not require knowing the location of a previous dataset, would eliminate this challenge.

SETTING A RANDOM SEED (INCORRECTLY)

To overcome the challenge of the previous solution, the idea was proposed to set a random seed which is based on the information contained in the row, such as subject ID, analysis visit, and question number. We construct a 7 digit seed number which concatenates the 4-digit subject ID, 2-digit analysis visit (ie, Baseline = “00”, Week 24 = “24”, and Week 48 = “48”), and the question number.

For example, in Table 3, the seed for observation 102 would be *1001*00**2**

The “*1001*” in italics is the subject ID. The “00” underlined is the visit (baseline), and the “**2**” in bold represents the question number.

However, as Rick Wicklin pointed out in his online article “Random number seeds: Only the first seed matters!”, only the first call of the seed matters. To illustrate this point, let us use the RANUNI() function to generate random uniform values for the Week 24 and Week 48 data.

```
/* Incorrect method for employing an observation-based seed for random number
generation */

data example_wk24_aval_method2_bad;
  set example_wk24;
  if AVISIT = "Baseline" then AVISITN = "00";
  else if AVISIT = "Week 24" then AVISITN = "24";
  else if AVISIT = "Week 48" then AVISITN = "48";
  seed = input(strip(SUBJID)||strip(AVISITN)||strip(qnum),8.);
  if length(strip(avalc)) = 1 then aval = input(avalc,1.);
  else if length(strip(avalc)) = 3 then do;
    val1 = input(scan(avalc,1," "),1.);
       val2 = input(scan(avalc,2," "),1.);
    if abs(val2-val1) = 1 then do;
```

```
        x = ranuni(seed);
        if 0 <= x <= 0.5  then aval = val1;
        else aval = val2;
      end;
  end;
  else if length(strip(avalc)) > 3 then aval = .;
run;
```

For the Week 24 data, the results for observations #2, #9, and #17 using this (incorrect) method appear in Table 6.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
2	1001	Baseline	2	1 2	1	0.1231643293
9	1002	Baseline	1	0 1	1	0.8206141078
17	1003	Baseline	1	2 3	2	0.212688156

Table 6. Imputed Values for Questionnaire Data at the time of the first analysis using RANUNI()

Similar code was used on the Week 48 data and the results for observations #102, #109, #113, #122, and #125 using this (incorrect) method appear in Table 7.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
102	1001	Baseline	2	1 2	1	0.1231643293
109	1001	Week 48	1	3 4	4	0.8206141078
113	1002	Baseline	1	0 1	0	0.212688156
122	1002	Week 48	1	1 2	1	0.3231994791
125	1003	Baseline	1	2 3	2	0.2612495293

Table 7. Imputed Values for Questionnaire Data at the time of the second analysis using RANUNI()

The same issue that we saw in Tables 2 and 4 appears again. Namely, the random values generated come from the same stream and are not changing based on the seed that we are creating. This is because the seed is only set once in the DATA step. Subsequent seeds are ignored.

One way to overcome this would be to write a macro to read in one observation at a time, call the seed based on the information contained in that single line, assign a value, and then stack all of the resulting observations. However, when dealing with a dataset with over 100,000 observations, this is not a practical approach to take.

SETTING A RANDOM SEED (CORRECTLY)

Thankfully, there is a way, however, to update the stream in a single DATA step using the CALL RANUNI() function. The code is only a one-line modification of the "bad" code above, highlighted in yellow.

```
/*Correct method for employing an observation-based seed for random number generation
*/

data example_wk24_aval_method2_good;
  set example_wk24;
  if AVISIT = "Baseline" then AVISITN = "00";
  else if AVISIT = "Week 24" then AVISITN = "24";
  else if AVISIT = "Week 48" then AVISITN = "48";
  seed = input(strip(SUBJID)||strip(AVISITN)||strip(qnum),8.);
  if length(strip(avalc)) = 1 then aval = input(avalc,1.);
  else if length(strip(avalc)) = 3 then do;
    val1 = input(scan(avalc,1," "),1.);
    val2 = input(scan(avalc,2," "),1.);
    if abs(val2-val1) = 1 then do;
        call ranuni(seed,x);
        if 0 <= x <= 0.5  then aval = val1;
        else aval = val2;
      end;
  end;
```

```
  else if length(strip(avalc)) > 3 then aval = .;
run;
```

For the Week 24 data, the results for observations #2, #9, and #17 using this method appear in Table 8.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
2	1001	Baseline	2	1 2	1	0.1231643293
9	1002	Baseline	1	0 1	0	0.4344587417
17	1003	Baseline	1	2 3	3	0.9307157239

Table 8. Imputed Values for Questionnaire Data at the time of the first analysis using CALL RANUNI()

Similar code was used on the Week 48 data and the results for observations #102, #109, #113, #122, and #125 using this method appear in Table 9.

Obs	Subj ID	Analysis Visit	Quest Num	Analysis Value (C)	Analysis Value (N)	Random Value
102	1001	Baseline	2	1 2	1	0.1231643293
109	1001	Week 48	1	3 4	3	0.4993879807
113	1002	Baseline	1	0 1	0	0.4344587417
122	1002	Week 48	1	1 2	2	0.6469203726
125	1003	Baseline	1	2 3	3	0.9307157239

Table 9. Imputed Values for Questionnaire Data at the time of the second analysis using CALL RANUNI()

Notice that the first random value, 0.1231643293, is the same value that appeared in Tables 6 and 7. As mentioned previously, this is because the first seed, 1001002, was the seed used for that record. However, unlike before, the random value for observation #9 matches that for observation #113 and similarly for observations #17 and #125.

While not shown, the SEED values will be updated. For example, in the final dataset, observation #2 has a SEED value of 1937711079. This is because the CALL RANUNI() updates the seed value, based on the original input provided (eg 1001002).

In passing, we note that while we generated random uniform values to illustrate the problem and solutions, since our desire is to determine if we should choose the first or second value, an alternative function, RANTBL(), was used. It allows for the generation of random values from a tabled probability distribution. Thus, calling the following

```
call rantbl(seed,0.5,0.5,x);
if x = 1 then aval = aval1;
else if x = 2 then aval = aval2;
```

would give x-values that are equal to 1 or 2 with probability 0.5 each and we can then choose the AVAL appropriately.

LIMITATIONS

The seed used in RANUNI() and RANTBL() needs to be $< 2^{31} - 1 = 2147483647$, which limits the length to approximately 10 digits. In our example above, a seed of 7 digits was required. In the motivating example, since multiple questionnaires were used, seeds of 9 digits were needed. For a study with a duration beyond 96 weeks (say 144 weeks), one more digit would be required and this would yield digits beyond the permitted length.

Using the RAND() function recommended by Rick Wicklin has an *extremely* long period ($2^{19937} - 1$), which contains over 6,000 digits. However, to use it would require a method to process each observation one at a time.

CONCLUSION

By using information unique to an observation such as a subject identifier, and visit and question number, a unique value (seed) for each observation can be assigned which is robust to updates as additional longitudinal data is collected. The CALL RANUNI() function can be used to assign a random value which is based on this unique seed, allowing for the creation of reproducibly random values for multiple analyses.

REFERENCES

Rick Wicklin, “Random number seeds: Only the first seed matters!”, website: http://blogs.sas.com/content/iml/2012/01/30/random-number-seeds-only-the-first-seed-matters/. Accessed 01AUG2014.

ACKNOWLEDGEMENTS

Thank you to Eric Wang for helping validate the randomization method.

CONTACT INFORMATION

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

Name: William Garner
Company: Gilead Sciences, Inc.
Address: 333 Lakeside Dr
City, State ZIP: Foster City, CA 94404
Work Phone: (650) 522-5691
E-mail: will.garner@gilead.com

Name: Ting Bai
Company: Gilead Sciences, Inc.
Address: 333 Lakeside Dr
City, State ZIP: Foster City, CA 94404
Work Phone: (650) 524-3954
E-mail: ting.bai@gilead.com