Paper SA11

Programmatic Generation of Health Canada's Comparative Bioavailability Information, Appendix B

Michael S Rimler, GlaxoSmithKline, Cincinnati, Ohio, US

This paper presents a programmatic solution for generating the requirements detailed in Appendix B of Health Canada's "Preparation of Comparative Bioavailability Information for Drug Submissions in the CTD Format" (2004). The two required components are an information file and a concentration file. We present a programmatic solution for generating each component for inclusion with the submission. Two benefits of implementing a programmatic solution include (i) the ability to perform validation on the components with independent programming, consistent with many clinical programming SOPs and (ii) the ability to replicate the process for future submissions to Health Canada.

SECTION 1: BACKGROUND

When submitting an Abbreviated New Drug Submission (ANDS) to the Therapeutic Products Directorate (TPD) of Health Canada, often sponsor organizations will refer to the 2004 guidance document “Preparation of Comparative Bioavailability Information for Drug Submissions in the CTD Format” [1]. The guidance defines “the Common Technical Document (CTD) format of drug submissions which rely on comparative bioavailability studies to establish safety and efficacy.” [1] The guidance is also applicable to Health Canada submissions which involve comparative bioavailability studies such as New Drug Submissions (NDS) or other supplemental submissions.

The full scope of the guidance includes the format, language, and structure of a submission. The typical CTD format includes 5 modules:

- Module 1: Administrative Information and Prescribing Information
- Module 2: Common Technical Document Summaries
- Module 3: Quality
- Module 4: Nonclinical Study Reports
- Module 5: Clinical Study Reports

Module 1 contains the Table of Contents for the submission, application information, product labelling, Health Canada summaries, an environmental assessment statement, and electronic review documents. Module 2 includes the CTD Table of Contents, the Quality Overall Summary, and clinical and nonclinical overviews and summaries (which are optional if the submission relies solely on comparative bioavailability studies). Module 3 presents relevant information “in accordance with relevant HC guidances and policies respecting Quality.” Module 4 presents relevant information from nonclinical studies, which is not expected to be applicable for submissions based on comparative bioavailability studies. Module 5 presents information on any Clinical Study Reports (CSRs) included in the submission.

Among the electronic review documents in Module 1.6 are electronic versions of the “proposed Product Monograph, Quality Summaries, and Comprehensive Summary: Bioequivalence (CS-BE), and the BE data sets”. The BE datasets are provided in ASCII format and technical requirements are outlined in Appendix B of the guidance. They include a data file with a *.dat extension (DAT file) and an information file with a *.inf extension (INF file). It is suggested that the file name used for these two components be equivalent and identify the company, drug, and formulation.

The programmatic generation of these elements of the overall submission package is the topic of this paper. Adopting a programmatic solution for these components can be efficiency enhancing, particularly when an organization is expecting to have many submissions to Health Canada which rely on comparative bioavailability studies to establish safety and efficacy. Facilitating the electronic comparison of outputs improves the efficiency of the independent validation process, particularly on reruns of the same output due to updated data or changes in requirements. The process also becomes replicable to other submissions as the inputs and code may be easily transferable and updated.

In Section 2, we look at the details of the DAT file and its generation. In Section 3, we turn attention to the INF file. We then summarize the discussion in Section 4 and discuss the merits of the programmatic approach to creating these components of the Health Canada CTD for drug submissions relying on comparative bioavailability studies.

SECTION 2: THE DAT FILE

The DAT file contains the measured and uncorrected drug concentrations as collected in a bioequivalence study. The guidance's suggested extension for the concentration file is *.dat. It is a horizontally formatted ASCII file with one record for each subject in each study period, sorted by treatment/formulation and subject. The example structure from the guidance is included in Figure 1:

Figure 1: Example of File Structure for the DAT File

Descriptor	Position	Length	Type	Example
subject number	1-2	2	alphanumeric	09
sequence	4-5	2	alphanumeric	AB or BA
study period	7-11	5	alphanumeric	1 or 2
treatment	13-13	1	alphanumeric	A or B
conc (0)	15-21	7	numeric	
conc (1)	23-29	7	numeric	
...	...	7	numeric	
conc (t)	114-120	7	numeric	

In this example, there is information to identify the concentration profile (i.e., each record), such as subject number, sequence, study period, and treatment information. Appended to the right of this information are the drug concentrations for that particular concentration profile. Note that each field in the file is granted a fixed number of columns. Here, Subject Number is given 2 columns, Study Period is given 5 columns, and treatment is given 1 column. Each concentration field in the example is allocated 7 columns to populate the data. The example also inserts a space delimiter between fields. For example, Subject Number ends at position 2 and Sequence begins at position 4. The space delimiter is contained in position 3. The space delimiter is not an explicit requirement, but it does help the human eye to read the data.

As stated in the guidance, Figure 1 is just an example of fields and lengths to be included in the DAT file. An example of the DAT file itself, with dummy data, is presented in Table 1:

Table 1: Example of Concentration data in the DAT File

```
01 AB 1 A 000.00 000.00 052.012 095.03 122.20 065.15 046.24 019.20 014.99 000.00 000.00
02 BA 2 A 000.00 000.00 062.012 094.13 125.21 064.14 044.81 017.27    .   000.00 000.00
03 BA 2 A 000.00 000.00 054.012 091.07 126.70 068.25 044.22 014.90 014.89 000.00 000.00
04 AB 1 A 000.00 000.00 055.012 093.63 123.52    .   045.43 012.21 012.94 000.00 000.00
05 BA 2 A 000.00 000.00 057.012 094.06 122.23 065.19 046.86    .   014.19 000.00 000.00
06 AB 1 A 000.00 000.00 058.012 091.03 125.30 066.74 047.74 019.52 011.92 000.00 000.00
```

There are a few derivation rules that are required for the creation of the DAT file. First, concentrations below the limit of quantitation (LOQ) should be entered as 0. Therefore, when reviewing the DAT file, it is not possible to identify which records are below the LOQ within the file itself. Second, the drug concentration should be uncorrected and "should not be below the lowest or above the highest nominal concentrations of the standard curve." This may also require presenting concentration data in units different from the units presented in the CSR analyses. Finally, any missing data must be entered as a period. We have included 3 examples of this in Table 1, such as Subject 02 in Study Period 2 in the ninth timepoint.

SAS GENERATION OF THE DAT FILE

This paper's macro code implements the following guidance requirements assuming an input dataset similar to a standard ADPC (CDISC, ADaM IG v1.0 or v1.1, BDS structure):

1. Concentrations below LOQ are presented as 0
2. Missing concentrations are presented as a period
3. Concentrations are presented uncorrected

In addition, the macro code:

1. Presents concentration data in desired units of measure, if different from analysis units
2. Allows the number of timepoints to be data driven
3. Allows the length of concentration fields to be data driven
4. Creates a SAS dataset for the concentration file that can be used for electronic compare by an independent validation programmer
5. Creates a data definition file that will be used as an input to the INF file and for electronic compare by an independent validation programmer

We only discuss elements of the macro code here. Please refer to the Appendix for the complete macro code for creation of the DAT file. The macro requires 4 input parameters:

- the path of input data,
- the path where output will be written,
- a conversion factor to convert concentrations from analysis units to desired units, and
- a filename to use when creating the DAT file.

The first step:

- Reads in the analysis data and subsets appropriately:
```
where paramcd="CONC_P1C";
```

- Converts concentrations below the LOQ to 0:
```
if find(avalc,"NQ")>0 and aval=. then aval=0;
```

- Converts all results to desired units:
```
aval_conv = aval*%eval(&convfact.);
```

In order to make the concentration fields data driven, we need to determine both the number of concentrations collected and the maximum length required to display concentrations in collected units.

```
proc sql noprint;
   select max(TotalLength)     into : len from &prefix.DS1;
   select max(NumDPS)          into : dps from &prefix.DS1;
   select count(distinct ATPT) into : num_tpts from &prefix.DS1;
quit;
```

Where, `TotalLength = Stem + NumDPS + 1` when the concentration is not an integer and `TotalLength = Stem` otherwise. For any concentration, `Stem` is the number of digits to the left of the decimal and `NumDPS` is the number of digits to the right. We can now construct a SAS format of the form `z7.3` to display concentrations with leading zeros, in this case with 7 characters and 3 decimal places such as `012.345`.

After transposing the data from vertical (ADaM BDS structure) to horizontal, we need to ensure that every subject and period is represented. Therefore, the code in the appendix creates records for subjects that did not enter period 2 in the example.

The next step implements the requirement that missing concentrations are represented by a period. For each concentration field `ii`,

```
if C&ii. = "" then C&ii. = tranwrd(put(0,z&xfmt.),"0"," ");
```

which converts the numeric 0 to the character "`000.000`" and changes each '0' to a space, yielding " . " as a result.

The final step is outputting the DAT file. This is accomplished in our example with a `data _null_` step:

```
data _null_;
   set &prefix.DS5;
   by SORT_BY_TRT SUBJECT;
   file "&pathout./&fname..dat" ;
   put SUBJECT SEQUENCE PERIOD TREATMENT
   %do ii = 1 %to &num_tpts.;
       C&ii. $char%eval(&len. + 1 + &dps. + 1).
   %end;
   ;
run;
```

The file is also output as a SAS dataset for an independent validation programmer to use for electronic compare purposes.

```
proc sort data=&prefix.DS5
          out=dddlib.DAT_FILE(drop=SORT_BY_TRT);
   by TREATMENT SUBJECT;
run;
```

The last block of code in the Appendix for the DAT file use PROC CONTENTS to create a data definition table that serves as the description of the record layout as detailed in the guidance. We make this data driven by using `length` from PROC CONTENTS, in combination with a retain statement, to derive the start and stop position of each field. We convert the PROC CONTENTS type variable into the guidance values of "NUMERIC" and "ALPHANUMERIC". We also leverage this opportunity to provide examples of each field. An example of the created dataset is in Figure 2. The creation of this dataset allows for an independent programmer to electronically compare the file record layout as part of the validation process of the Appendix B components.

Figure 2: Data Driven Description of the DAT File Record Layout

	DESCRIPTOR	POSITION	LENGTH	TYPE	EXAMPLE
1	Subject	1-6	6	ALPHANUMERIC	961001
2	Sequence	8-10	3	ALPHANUMERIC	A/B or B/A
3	Period	12-12	1	ALPHANUMERIC	1 or 2
4	Treatment	14-14	1	ALPHANUMERIC	A or B; missing treatment=.
5	Concentration 1	16-21	6	ALPHANUMERIC	
6	Concentration 2	23-28	6	ALPHANUMERIC	
7	Concentration 3	30-35	6	ALPHANUMERIC	
8	Concentration 4	37-42	6	ALPHANUMERIC	
9	Concentration 5	44-49	6	ALPHANUMERIC	
10	Concentration 6	51-56	6	ALPHANUMERIC	
11	Concentration 7	58-63	6	ALPHANUMERIC	
12	Concentration 8	65-70	6	ALPHANUMERIC	
13	Concentration 9	72-77	6	ALPHANUMERIC	
14	Concentration 10	79-84	6	ALPHANUMERIC	

We now turn attention to the INF file.

SECTION 3: THE INF FILE

The INF file is a templated file that contains specific information about the concentrations contained in the DAT file:

i. Sampling times
ii. Drug name, drug strength, dose form, potency, and dose level administered
iii. Limit of quantitation
iv. Standard curve range
v. Study period
vi. Treatment labelling
vii. Company name of the sponsor and firm performing the study
viii. Contact person
ix. Date the file was generated
x. Description of the record layout in the data file (DAT)

An example of the INF file is given in Figure 3.

Figure 3: Example of the INF File

i. SAMPLING TIMES:	0 / 0.5 / 1.0 / 1.5 / 2.0 / 2.5 / 3.0 / 3.5 / 4.0 / 5.0 / 7.0 / 9.0 (N=12)
ii. (a) DRUG NAME:	Noname
(b) DRUG STRENGTH COMPARED:	50 mg
(c) DOSAGE FORM:	tablets
(d) POTENCY:	97.8% (A = test) and 98.1% (B = reference)
(e) DOSE ADMINISTERED:	50 mg
iii. LIMIT OF QUANTITATION (LOQ):	10 ng/mL
iv. STANDARD CURVE RANGE:	10 ng/mL to 150 ng/mL
v. STUDY PERIOD:	STUDY PERIOD 1: May 14, 1989. STUDY PERIOD 2: May 21, 1989.
vi. TREATMENT LABELLING:	A = test product; B = reference product
vii. (a) SPONSOR COMPANY'S NAME:	Bioequivalence Inc.
(b) FIRM PERFORMING STUDY:	Biostudy Inc.
viii. CONTACT PERSON:	Dr. A. Noname (phone and fax number)
ix. DATE FILE GENERATED:	October 9, 1993

The study team could simply create this document manually and review for quality. However, in our case, we decided to create this document programmatically. This approach allows for:

- The independent programmer to review the template for layout issues, allowing the focus after each run to be on the data driven elements,
- Selected components to be data driven, instead of hard coded,
- The data definition table (record layout) to be data driven, instead of hard coded, and
- Portability of the code from one submission to another

The last point would require a manual update of the template to be input for fields not selected to be data driven (e.g., standard curve range or contact person).

In our case, we started with a templated version of the final file as given below in Table 2:

Table 2: Template Version of INF File for Use as Input

```
#i. SAMPLING TIMES:                0 / 0.5 / 1.0 / 2.0 / 3.0 / 4.0 / 6.0 / 8.0 / 12.0 hours (N=<bigN>)
#
#ii.
#(a) DRUG NAME:                    FAKENAME
#(b) DRUG STRENGTH COMPARED:       Fakedrugimab 30.0 mg
#(c) DOSAGE FORM:                  tablet
#(d) POTENCY:                      Fakedrugimab 99.2%
#(e) DOSE ADMINISTERED:            Fakedrugimab 30.0 mg
#
#iii. LIMIT OF QUANTITATION (LOQ): LOQ for Fakedrugimab is 10.0 ng/mL
#
#iv. STANDARD CURVE RANGE:         10.0 to 10000 ng/mL for Fakedrugimab
#
#v.  STUDY PERIOD:                 First subject first visit (screening start): June 27, 2017
#                                  Last subject last visit (Last follow-up visit): January 18, 2019
#
#vi. TREATMENT LABELLING:          <TRTA>
#                                  <TRTB>
#
#vii.
#a.      SPONSOR COMPANY'S NAME:   GlaxoSmithKline
#b.      FIRM PERFORMING STUDY:    Rimler Pharmacology, 1600 Pennsylvania Ave NW, Washington, DC 20500
#
#viii. CONTACT PERSON:             George Washington
#                                  Phone (615) 555-1212
#                                  Fax (615) 555-1213
#
#ix. DATE FILE GENERATED:          <insert date>
#
#x. Description of the record layout in the data file
#
#DESCRIPTOR                      POSITION    LENGTH  TYPE          EXAMPLE
```

There are a number of features of this text file that is an ***input*** to the process. First, it is structured as detailed in the guidance. Second, we decided to have 4 fields of the INF file be data driven:

- **<bigN>**: The number of subjects in the appropriate analysis population
- **<TRTA>**: Treatment label for the Treatment A
- **<TRTB>**: Treatment label for the Treatment B
- **<insert date>**: The date that the file was generated

In your application, additional placeholders could be inserted to replace different elements that are hardcoded in our example. For example, perhaps the nominal timepoints for PK assessments, the drug name, or the LOQ value could be derived from the analysis database and populated into this template upon execution.

A third element of the template file is the leading '#' symbols on each line. In our application, this aided the alignment of the resulting lines on outputting the data. Without these symbols in column 1 of each line, leading spaces of indented lines were removed prior to outputting the resulting file, suppressing the formatted nature of the template. We acknowledge that this may not be the only solution to the problem.

A final element to highlight before reviewing the code is the last line, which provide a header for the data definition table. With this line imported by the macro code, we can simply set the SAS dataset created by the DAT file macro under the dataset created when this template file is read in. Other than formatting and replacing placeholders, the INF file would be ready to be output.

SAS GENERATION OF THE INF FILE

The macro code discussed in this paper reads in the template INF file and the data definition table created by the DAT file macro. In processing the data, it then:

1. Derives the current system date for the 'Date File Generated' field
2. Derives the appropriate treatment labels
3. Counts the number of subjects in the analysis population
4. Replaces all placeholders appropriately with derived values
5. Formats the data definition information for readability when concatenated with the INF template

Complete macro code for creating the INF file is included in the Appendix. We discuss elements of the macro code here.

The macro requires 5 input parameters: the path of input data, the path where output will be written, the template INF file to be read in, the data definition table filename, and a filename to use when creating the INF file.

Obtaining the system date is a standard piece of SAS code and is accomplished in the macro as:

```
data _null;
   x = put(today(),worddate.);
   call symput("today",strip(x));
run;
```

Note the format `worddate`, which outputs the date in the format "Month DD, YYYY". We then count derive treatment labels and number of subjects:

```
proc sql noprint;
   select count(distinct USUBJID) into : bigN from admlib.adsl where PKCFL="Y";
   select TRTP                    into : trta from admlib.adpc where TRTPN=1;
   select TRTP                    into : trtb from admlib.adpc where TRTPN=2;
quit;
```

The next step is reading the template file into a SAS dataset:

```
filename _in_file "&pathout./refdata/&in_file.";
data &prefix.INF1;
   length _line $200;
   infile _in_file lrecl=32767 truncover;
   input _line $1-200;
   _line = substr(_line,2); *** Ignore character in column 1 ***;
run;
```

Now that we have each line in the template file in a SAS dataset, we concatenate it with the data definition table. For records in the template file, we replace the placeholders with data driven values:

```
*** Creation Date ***;
if find(_line,"<insert date>")>0 then line = tranwrd(_line,"<insert
date>","&today.");
*** Number of subjects ***;
else if find(_line,"<bigN>")>0 then line = tranwrd(_line,"<bigN>","%eval(&bigN.)");
*** Treatment label 1 ***;
else if find(_line,"<TRTA>")>0 then line = tranwrd(_line,"<TRTA>","&trta.");
*** Treatment label 2 ***;
else if find(_line,"<TRTB>")>0 then line = tranwrd(_line,"<TRTB>","&trtb.");
else line = _line;
```

For records in the data definition table, we format for alignment. In the following, note that x is a single space.

```
line = cat(DESCRIPTOR,x,POSITION,x,x,x,x,LENGTH,x,x,x,x,x,TYPE,x,x,EXAMPLE);
```

The last step is to output the INF file:

```
data _null_;
  set &prefix.INF2 end=eof;
  file "&pathout./&fname..inf" ;
  put LINE $char200.;
run;
```

SECTION 4: SUMMARY

Appendix B of Health Canada's "Preparation of Comparative Bioavailability Information for Drug Submissions in the CTD Format" provides technical specifications for submitting bioequivalence datasets as one of the electronic review documents (Module 1.6). This paper has discussed how to generate the concentration data and information files to the specifications detailed in the guidance. We argue that programmatic generation of these components similar to the methods discussed provides at least two areas of benefit:

(i) the ability to perform validation on the components with independent programming, consistent with many clinical programming SOPs and
(ii) the ability to replicate the process for future submissions to Health Canada.

From the validation perspective, electronic comparison of outputs is facilitated through the independent and programmatic generation of

- A SAS dataset with formatted fields used directly to generate the DAT file
- A SAS dataset used directly in the INF file containing information on the DAT file record layout
- Data driven elements of the INF file that can be independently derived and compared

In addition, replication of the process is facilitated through:

- The programmatic generation of formatted DAT file (and SAS dataset) to Appendix B specifications, allowing transportability of code from one submission to another
- The programmatic generation of the INF file with input template file and data-driven fields
    - Using PROC CONTENTS from DAT generation keeps components in sync
    - Date generated field
- Simple modification of elements of the input template file which are submission specific and not data driven

Adopting a programmatic solution to the generation of these components can be efficiency enhancing, particularly when an organization is expecting to have many submissions to Health Canada which rely on comparative bioavailability studies to establish safety and efficacy.

REFERENCES

[1] "Preparation of Comparative Bioavailability Information for Drug Submissions in the CTD Format" (2004) https://www.canada.ca/content/dam/hc-sc/migration/hc-sc/dhp-mps/alt_formats/hpfb-dgpsa/pdf/prodpharma/draft_ebauche_ctdbe-eng.pdf

CONTACT INFORMATION

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

Michael S Rimler
GlaxoSmithKline
1250 S. Collegeville Road
Collegeville, Pennsylvania, US, 19426-0989
Email: michael.s.rimler@gsk.com

APPENDIX A – SAMPLE CODE FOR DAT FILE (ru_cr8_dat.sas)

```
/*************************************************************************
| Program Name:    ru_cr8_dat.sas
| Program Version: 1
| Protocol ID:     mgr-140627
| Program Purpose: Create .DAT file for Health Canada CSBE requirement
| SAS Version:     9.4
| Created By:      Michael Rimler
| Date:            10-Mar-2020
*************************************************************************/
%macro ru_cr8_dat(pathin=
                 ,pathout=
                 ,convfact=
                 ,fname=
                  );

%let prefix=_dat_;

options nofmterr;

libname admlib "&pathin./adamdata" access=readonly;
libname outlib "&pathout./output";
libname reflib "&pathout./refdata";
libname dddlib "&pathout./dddata";

data &prefix.DS1;
   set admlib.adpc;
   where paramcd="CONC_P1C";                      *** Assumed only one analyte to output ***;
   if find(avalc,"NQ")>0 and aval=. then aval=0;  *** Not quantifiable results converted to 0 per guidance ***;
   if aval ne . then do;
      aval_conv = aval*%eval(&convfact.);         *** Convert from analysis units to collected units ***;
      avalc_conv = strip(put(aval_conv,best.));
      end;
   else do;
      aval_conv=.;
      avalc_conv="";
   end;

   *** Generate Numeric Timepoint from analysis timepoint ***;
   if ATPT="PREDOSE" then ATMPTN = 0;
   else ATMPTN = input(substr(ATPT,1,find(ATPT,"H")-1),best.);

   *** Assess precision and Total Length of concentration result ##.### ***;
   if find(avalc_conv,".")>0 then Stem = length(substr(avalc_conv,1,find(avalc_conv,".")-1));
   else Stem = length(avalc_conv);
```

```
   if find(avalc_conv,".")>0 then NumDPS = length(substr(avalc_conv,find(avalc_conv,".")+1));
   else NumDPS = 0;

   if NumDPS > 0 then TotalLength = Stem + NumDPS + 1;
   else TotalLength = Stem;
run;

proc sql noprint;
   select max(TotalLength)     into : len from &prefix.DS1;      *** Longest concentration ***;
   select max(NumDPS)          into : dps from &prefix.DS1;      *** Greatest # decimals   ***;
   select count(distinct ATPT) into : num_tpts from &prefix.DS1; *** Number of timepoints  ***;
quit;

*** Concentration format X.D ***;
data _null_;
   x = (&len. + 1 + &dps.) + &dps./10;
   call symput('xfmt',strip(put(x,best.)));
run;


data &prefix.DS2;
   set &prefix.DS1;

   length SUBJECT $6 SEQUENCE $3 PERIOD $1 TREATMENT $1;
   length AVALC2 $%eval(&len. + 1 + &dps.);

   *** Create formatted values with leading zeroes to required standard length ***;
   if aval_conv=. then AVALC2 = tranwrd(put(0,z&xfmt.),"0"," ");
   else AVALC2 = put(aval_conv,z&xfmt.);

   *** Record identifying fields ***;
   SUBJECT = strip(SUBJID);                                 *** Subject ID       ***;
   if find(TRTSEQA,"/")>0 then SEQUENCE = strip(TRTSEQA); *** Actual Sequence  ***;
   else SEQUENCE = strip(TRTSEQA)||"/.";                    *** Early w/d        ***;
   PERIOD = strip(put(APERIOD,best.));                      *** Period           ***;
   TREATMENT = substr(TRTA,1,1);                            *** Actual Treatment ***;
run;

*** Unique records ***;
proc sort data=&prefix.DS2
          out=&prefix.DS3
          nodupkey dupout=d;
   by SUBJECT SEQUENCE PERIOD TREATMENT ARELTM;
run;
```

```
*** Transpose formatted concentrations for horizontal layout per guidance ***;
proc transpose data=&prefix.DS3
               out=&prefix.DS4(drop=_:)
               prefix=C;
   by SUBJECT SEQUENCE PERIOD TREATMENT;
   var AVALC2;
run;

*** Create records for all subject and all planned timepoints ***;
proc sort data=&prefix.DS4 out=&prefix.SUBJECT(keep=SUBJECT SEQUENCE) nodupkey;
   by SUBJECT SEQUENCE;
run;

proc sort data=&prefix.DS4 out=&prefix.PERIOD(keep=PERIOD) nodupkey;
   by PERIOD;
run;

proc sql noprint;
   create table &prefix.DUM1 as
   select *
   from &prefix.SUBJECT, &prefix.PERIOD
   order by SUBJECT, SEQUENCE, PERIOD;
quit;

data &prefix.DS5;
   merge &prefix.DS4(in=a)
         &prefix.DUM1(in=b);
   by SUBJECT SEQUENCE PERIOD;
   if TREATMENT="" then do;
      if PERIOD="1" then TREATMENT = substr(SEQUENCE,1,1);
      else if PERIOD="2" then TREATMENT = substr(SEQUENCE,3);
      else TREATMENT="";
      end;

   %do ii = 1 %to &num_tpts.;
      if C&ii. = "" then C&ii. = tranwrd(put(0,z&xfmt.),"0"," ");
   %end;

   *** Added for sorting missing treatments ***;
   if TREATMENT="." then SORT_BY_TRT = "B";
   else SORT_BY_TRT = TREATMENT;
run;

proc sort data=&prefix.DS5;
```

```
   by SORT_BY_TRT SUBJECT;
run;

*** Create DAT file for submission ***;
data _null_;
   set &prefix.DS5;
   by SORT_BY_TRT SUBJECT;
   file "&pathout./&fname..dat" ;
   put SUBJECT SEQUENCE PERIOD TREATMENT
   %do ii = 1 %to &num_tpts.;
       C&ii. $char%eval(&len. + 1 + &dps. + 1).
   %end;
    ;
run;

*** Output dataset for validation compare purposes ***;
proc sort data=&prefix.DS5 out=dddlib.DAT_FILE(drop=SORT_BY_TRT);
   by TREATMENT SUBJECT;
run;

*** Create data definition information to be input into INF file ***;
proc contents data=work.&prefix.DS5
              out=&prefix.DATA1(keep=NAME TYPE LENGTH VARNUM)
              directory details varnum noprint;
run;

data &prefix.DATA2;
   set &prefix.DATA1;
   if substr(NAME,1,1)="C" then NAME = tranwrd(NAME,"C","Concentration ");
   else NAME = propcase(NAME);
run;

proc sort data=&prefix.DATA2;
   by VARNUM;
run;

data &prefix.DATA3(rename=(length=LENGTH name=DESCRIPTOR));
   retain IDX 0 EndPos 0;
   set &prefix.DATA2(rename=(TYPE=TYPEN));
   by VARNUM;
   where upcase(NAME) ne "SORT_BY_TRT";

   *** Start and stop positions ***;
   if _n_ = 1 then StartPos = 1;
   else StartPos = EndPos + 2;
```

```
   EndPos = StartPos + Length - 1;
   IDX = IDX + LENGTH;

   length POSITION $10;
   POSITION = compress(put(StartPos,best.)||"-"||put(EndPos,best.));

   length TYPE $12;
   if TYPEN=1 then TYPE = "NUMERIC";
   else if TYPEN=2 then TYPE = "ALPHANUMERIC";
   else TYPE = "";

   length EXAMPLE $30;
   if upcase(NAME) = "SUBJECT"   then EXAMPLE = "011818";
   if upcase(NAME) = "SEQUENCE"  then EXAMPLE = "A/B or B/A";
   if upcase(NAME) = "PERIOD"    then EXAMPLE = "1 or 2";
   if upcase(NAME) = "TREATMENT" then EXAMPLE = "A or B; missing treatment=.";

   label name= length=;
run;

proc sql noprint;
   create table reflib.INF_DATA_STRUCTURE as
   select DESCRIPTOR, POSITION, LENGTH, TYPE, EXAMPLE
   from &prefix.DATA3;
quit;

*** Clear Library ***;
proc datasets library=work nodetails nolist;
   delete &prefix.: d;
run;
quit;
%mend ru_cr8_dat;
```

APPENDIX B – SAMPLE CODE FOR INF FILE (ru_cr8_inf.sas)

```
/************************************************************************
| Program Name:    ru_cr8_inf.sas
| Program Version: 1
| Protocol ID:     mgr-140627
| Program Purpose: Create .INF file for Health Canada CSBE requirement
| SAS Version:     9.4
| Created By:      Michael Rimler
| Date:            10-Mar-2020
************************************************************************/
%macro ru_cr8_inf(pathin=
                 ,pathout=
                 ,in_file=
                 ,in_data=
                 ,fname=
                  );
%let prefix=_inf_;

options nofmterr;

libname admlib "&pathin./adamdata" access=readonly;
libname reflib "&pathout./refdata";

filename _in_file "&pathout./refdata/&in_file.";

*** Get run date for INF execution ***;
data _null;
   x = put(today(),worddate.);
   call symput("today",strip(x));
run;

*** Count number of subjects in PK population and get treatment labels ***;
proc sql noprint;
   select count(distinct USUBJID) into : bigN from admlib.adsl where PKCFL="Y";
   select TRTP                    into : trta from admlib.adpc where TRTPN=1;
   select TRTP                    into : trtb from admlib.adpc where TRTPN=2;
quit;

options ls=max;

*** Read in templated INF File with place holders ***;
data &prefix.INF1;
   length _line $200;
   infile _in_file lrecl=32767 truncover;
   input _line $1-200;
```

```
   _line = substr(_line,2); *** Ignore character in column 1 ***;
run;

*** Set templated INF File on DAT data definition file ***;
data &prefix.INF2;
   set &prefix.INF1(in=a)
       reflib.&in_data.(in=b);
   format line $char200.;
   x = " ";

   *** For placeholders, replace with data-driven values ***;
   if a then do;
      *** Creation Date ***;
      if find(_line,"<insert date>")>0 then line = tranwrd(_line,"<insert date>","&today.");
      *** Number of subjects ***;
      else if find(_line,"<bigN>")>0 then line = tranwrd(_line,"<bigN>","%eval(&bigN.)");
      *** Treatment label 1 ***;
      else if find(_line,"<TRTA>")>0 then line = tranwrd(_line,"<TRTA>","&trta.");
      *** Treatment label 2 ***;
      else if find(_line,"<TRTB>")>0 then line = tranwrd(_line,"<TRTB>","&trtb.");
      else line = _line;
      end;
   *** else format data definition information for INF File ***;
   else line = cat(DESCRIPTOR,x,POSITION,x,x,x,x,LENGTH,x,x,x,x,x,TYPE,x,x,EXAMPLE);
   keep line;
run;

*** Output INF File ***;
data _null_;
  set &prefix.INF2 end=eof;
  file "&pathout./&fname..inf" ;
  put LINE $char200.;
run;

*** Clear Library ***;
proc datasets library=work nodetails nolist;
   delete &prefix.:;
run;
quit;
%mend ru_cr8_inf;
```