解析用に作成したデータから CDISC SDTM xx と SUPPxx を 自動的に作成するプログラムの紹介

クライミングコンフロントシステム株式会社
鈴木 則之

紹介するプログラムの概略

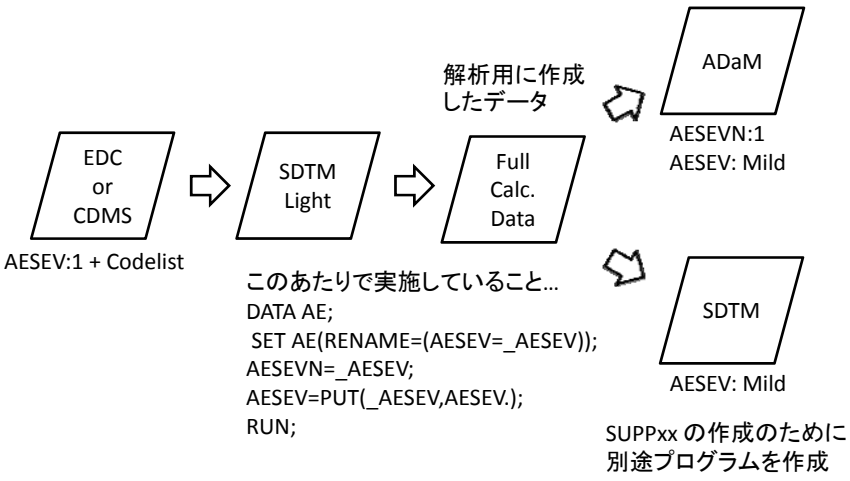
- To Main
  - SDTM ひな形データセット（モデルデータセット）にも変換元のデータセットにも存在する変数をMainへの変換対象にする
  - 同一変数名であっても変数型が異なる可能性があるのでいったんRENAMEを行ったうえで同一変数名へのマッピングを行う
  - フォーマットが設定されていたらそれを用いて文字列変換を行う
- To SUPPxx
  - モデルデータセットには存在しない変数を変換対象にする
  - フォーマットが設定されていたらそれを用いて文字列変換を行う
  - Null の場合は変換しない

解析用データ vs. SDTM

解析用データ	SDTM
主に解析を目的としたもの	主に医学的レビューを目的としたもの
順序のあるコード値は数値データが扱いやすい（有害事象程度など）	順序のあるコード値はそのデコードであったが評価しやすい
必要な値を SUPPxx から取得するとなると使いにくい	主要変数以外の必要な値は SUPPxx へ

ここでの解析用データとは CDISC ADaM や SDTM を作成するための前段階のデータで、一通りの必要な計算が行われたものをさしています
どこかで「数値＋フォーマット」データを文字列変換する必要が生じます

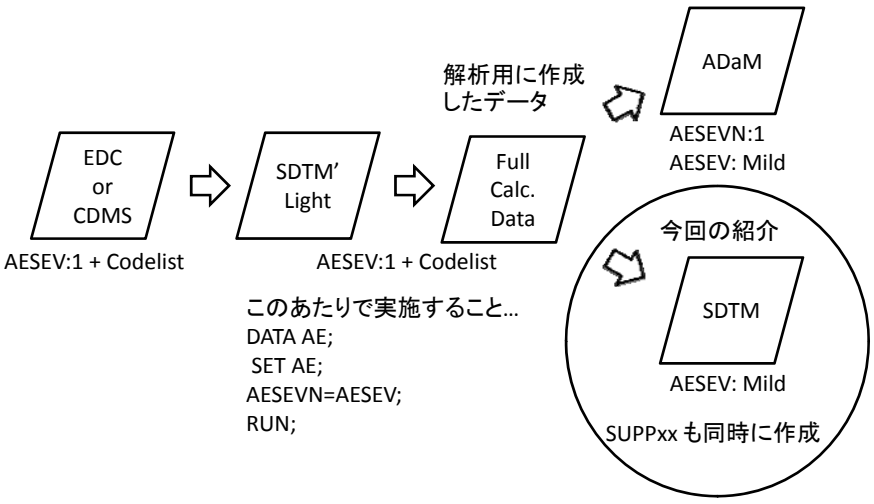
データの流れ（例）



改善したいこと

- To Main
  - 「値＋フォーマット」データからの変換を自動化する
  - CDISC SDTM の（文字列長さを含めた会社）定義が決まっているのだからそれに強制的に合わせる
- To SUPPxx
  - SUPPxxへ出力すべきものはSDTM中に定義されていない変数なのだから、それを自動的に検出し変換する

データの流れ（今回の紹介）



前提

- CDISC SDTM Main の会社定義が決定していてそのひな形となるデータセット（モデルデータセットといいます）が作成されている
- CDISC SDTM SUPPxx の会社定義が決定していてそのひな形となるデータセットが作成されている
- 必要なフォーマットが作成されている
- 元データは変数ラベルが設定されている
- Main, SUPPxx のいずれにも変換しない変数は元データの段階でDROP されている
- IDVAR は xxSEQ である
- 元データにアンダーバー(_)から始まる変数は存在しない
- 元データの Null 値は変換後も Null 値である

Main への変換

- モデルデータセットと元データセットの双方に同一変数名の変数が存在する場合に変換を行う
  - 元データセットの変数を RENAME
    - RENAME=(… AESEV = _0000016 AEREL = _0000020 …);
  - KEEP 対象変数とする
    - KEEP … AESEV AEREL … ;
  - 元データセットのFORMATにより文字列に一旦変換したものをモデルデータセットのINFORMATを用いて変換
    - IF _0000016 NE . THEN AESEV = INPUT(TRIM(LEFT(PUT(_0000016,AESEV.))),??$CHAR200.); ELSE AESEV = '';
    - IF _0000020 NE . THEN AEREL = INPUT(TRIM(LEFT(PUT(_0000020,AEREL.))),??$CHAR200.); ELSE AEREL = '';

Main への変換

- モデルデータセットと元データセットの双方に同一変数名の変数が存在する場合に変換を行う
  - 変換にエラーがあったかどうかを確認するプログラムを作成
    - IF _0000016 NE . AND AESEV NE INPUT(TRIM(LEFT(PUT(_0000016,AESEV.))),??$CHAR200.) THEN _CHK_0000016='BAD';
    - IF AESEV EQ '' AND _0000016 NE . OR AESEV NE '' AND _0000016 EQ . THEN _CHK_0000016='BAD';
  - 前項の確認するための変数を RETAIN

SUPPxx への変換

- モデルデータセットには存在しないが元データセットには存在する変数でNullでない場合に変換
  - 1レコードづつ FETCH し、STUDYID, RDOMAIN, USUBJID, xxSEQ の値を取得
  - 1変数づつMainに変数が存在するかを確認し、ない場合は変数名を QNAM へ、変数ラベルをQLABEL へ持たせる
  - 変数の値を取得し、元データのFORMATにより文字列に変換しQVAL に持たせる
  - 変換にエラーがあったかどうかを確認
  - OUTPUT

SUPPxx への変換

- 今回の紹介プログラムでは行っておりませんが、例えばSTUDYID, RDOMAIN, QNAM と QORIG, QEVAL のマッピングがされている QNAMINF というデータセットが存在する場合 MERGE やSQL UPDATE により値を持たせることができます
- SQL UPDATE による QORIG, QEVAL 変換例

```
PROC SQL;
  UPDATE SUPPxx AS A
  SET QORIG=(SELECT B.QORIG FROM QNAMINF AS B WHERE B.STUDYID=A.STUDYID AND
B.RDOMAIN=A.RDOMAIN AND B.QNAM=A.QNAM)
  ;
 UPDATE SUPPxx AS A
  SET QEVAL=(SELECT B.QEVAL FROM QNAMINF AS B WHERE B.STUDYID=A.STUDYID AND
B.RDOMAIN=A.RDOMAIN AND B.QNAM=A.QNAM)
  ;
QUIT;
```

連絡先

- ご質問などがございましたら下記までご連絡ください

クライミングコンフロントシステム株式会社

代表取締役 鈴木 則之

℡ 03-4530-6041

Mailto : suzuki@climbingconfront.co.jp

HP : http://www.climbingconfront.co.jp

```
/*****************************************************************************/
/* AutoSDTM_Program.SAS                                                      */
/*                                                                           */
/* SDTM へ変換する直前のデータ（解析用データセットなど）の形式が SDTM と異な  */
/* る（SDTM 文字型であるのに元データは数値＋FORMATの場合など）ことがあること  */
/* も考慮して元データから SDTM 形式へのデータ変換を行います                  */
/* モデルデータセットに同一の変数名が存在する場合は Main (AE など) のデータセ */
/* ットへ、存在しない場合は SUPPxx (SUPPAE など) のデータセットに変換します   */
/* 変換元の段階で必須変数は存在しており、逆に SUPPxx への変換も行わない変数は */
/* DROP されている必要があります                                             */
/* また、STUDYID, USUBJID, DOMAIN, xxSEQ には FORMAT, INFORMAT は付されていな */
/* いこととします                                                            */
/*                                                                           */
/*           2011/07/07  Climbing Confront System Co., Ltd.  Suzuki Noriyuki */
/*                 Developed on SAS 9.2 Japanese Version For Windows 32 Bits */
/*****************************************************************************/

/* SDTM への変換元となるデータセットの保存場所     */
LIBNAME PREDS 'P:\Climbing Confront System\SASユーザ会\2011\Poster\CalcDs'  ACCESS=READONLY;
/* SDTM に従い定義したモデルデータセットの保存場所 */
LIBNAME MDLDS 'P:\Climbing Confront System\SASユーザ会\2011\Poster\ModelDs' ACCESS=READONLY;
/* SDTM への変換結果の保存場所                     */
LIBNAME RESDS 'P:\Climbing Confront System\SASユーザ会\2011\Poster\SDTMDs'                ;

OPTIONS FMTSEARCH=(WORK LIBRARY PREDS MDLDS RESDS) NOFMTERR MISSING=' ' COMPRESS=YES;

%MACRO AutoSDTM(_DOMAIN_);
  /* Var Check and Create Macro Strings */
  DATA    _NULL_;
  LENGTH  _KEEP_STR_
          _RETAIN_STR_
          _RENAME_STR_
          _TRANS_STR_
          _CHK_STR1_
          _CHK_STR2_
          _RESULT_STR_ $32767
          _TMP_STR_    $01024;
  dsidm=OPEN("MDLDS.&_DOMAIN_",'I');
  dsidp=OPEN("PREDS.&_DOMAIN_",'I');
  _mnvars_=ATTRN(dsidm,'NVARS');
  _pnvars_=ATTRN(dsidp,'NVARS');
  DO _mvarnum=1 TO _mnvars_;
    /* Model DS Info. */
    _mvarname =VARNAME (dsidm,_mvarnum);
    _mvartype =VARTYPE (dsidm,_mvarnum);
    _mvarlen  =VARLEN  (dsidm,_mvarnum);
    _mvarlabel=VARLABEL(dsidm,_mvarnum);
    _mvarfmt  =VARFMT  (dsidm,_mvarnum);
    _mvarinfmt=VARINFMT(dsidm,_mvarnum);
    /* Original DS Info. */
    _pvarnum=VARNUM(dsidp,_mvarname);
    /* 双方に変数が存在するときに実行 */
    IF _pvarnum>0 THEN DO;
      _pvarname =VARNAME (dsidp,_pvarnum);
      _pvartype =VARTYPE (dsidp,_pvarnum);
      _pvarlen  =VARLEN  (dsidp,_pvarnum);
      _pvarlabel=VARLABEL(dsidp,_pvarnum);
      _pvarfmt  =VARFMT  (dsidp,_pvarnum);
      _pvarinfmt=VARINFMT(dsidp,_pvarnum);
    /* KEEP 変数の対象とする */
      /* Temporary String */
      _TMP_STR_=TRIM(LEFT(_mvarname));
      /* Join */
      IF _KEEP_STR_  ='' THEN _KEEP_STR_  =                        TRIM(LEFT(_TMP_STR_));
                         ELSE _KEEP_STR_  =TRIM(_KEEP_STR_)  ¦¦' '¦¦TRIM(LEFT(_TMP_STR_));
    /* Check 変数を RETAIN 変数の対象とする */
      /* Temporary String */
      _TMP_STR_='_CHK_'¦¦PUT(_mvarnum,Z7.);
      /* Join */
      IF _RETAIN_STR_='' THEN _RETAIN_STR_=                        TRIM(LEFT(_TMP_STR_));
                         ELSE _RETAIN_STR_=TRIM(_RETAIN_STR_)¦¦' '¦¦TRIM(LEFT(_TMP_STR_));
    /* RENAME 変数の対象とする */
      /* Temporary String */
      _TMP_STR_=TRIM(LEFT(_mvarname))¦¦' = _'¦¦PUT(_mvarnum,Z7.);
      /* Join */
      IF _RENAME_STR_='' THEN _RENAME_STR_=                        TRIM(LEFT(_TMP_STR_));
                         ELSE _RENAME_STR_=TRIM(_RENAME_STR_)¦¦' '¦¦TRIM(LEFT(_TMP_STR_));
    /* 変換式を作成する */
      /* Format Set */
      _varfmt  =_pvarfmt  ;
      IF _varfmt  ='' THEN DO;
        IF _pvartype='N' THEN _varfmt  ='BEST32.'; ELSE _varfmt  ='$CHAR200.';
      END;
      /* Informat Set */
      _varinfmt=_mvarinfmt;
      IF _varinfmt='' THEN DO;
        IF _mvartype='N' THEN _varinfmt='BEST32.'; ELSE _varinfmt='$CHAR200.';
      END;
      /* Temporary String */
      _TMP_STR_='IF  '¦¦PUT(_mvarnum,Z7.)¦¦' NE '¦¦IFC(_pvartype='N',' .','''''')¦¦' THEN '¦¦
                TRIM(LEFT(_mvarname))¦¦' = INPUT(TRIM(LEFT(PUT(_'¦¦PUT(_mvarnum,Z7.)¦¦','¦¦TRIM(LEFT(_varfmt))¦¦'))),??'¦¦
                TRIM(LEFT(_varinfmt))¦¦'); ELSE '¦¦TRIM(LEFT(_mvarname))¦¦' = '¦¦IFC(_mvartype='N',' .','''''')¦¦';';
      /* Join */
      IF _TRANS_STR_ ='' THEN _TRANS_STR_ =                        TRIM(LEFT(_TMP_STR_));
```

```
                    ELSE _TRANS_STR_ =TRIM(_TRANS_STR_ )||' '||TRIM(LEFT(_TMP_STR_));
  /* 変換結果を Check する */
    /* Temporary String */
    _TMP_STR_='IF _'||PUT(_mvarnum,Z7.)||' NE '||IFC(_pvartype='N',' .','''''')||' AND '||TRIM(LEFT(_mvarname))||
              ' NE INPUT(TRIM(LEFT(PUT(_'||PUT(_mvarnum,Z7.)||','||TRIM(LEFT(_varfmt))||'))),??'||TRIM(LEFT(_varinfmt))||
             ') THEN _CHK_'||PUT(_mvarnum,Z7.)||'=''BAD'';';
    /* Join */
    IF _CHK_STR1_  ='' THEN _CHK_STR1_  =                        TRIM(LEFT(_TMP_STR_));
                       ELSE _CHK_STR1_  =TRIM(_CHK_STR1_  )||' '||TRIM(LEFT(_TMP_STR_));
    /* Temporary String */
    _TMP_STR_='IF '||TRIM(LEFT(_mvarname))||' EQ '||IFC(_mvartype='N',' .','''''')||' AND _'||PUT(_mvarnum,Z7.)||' NE '||
                    IFC(_pvartype='N',' .','''''')||' OR '||
                    TRIM(LEFT(_mvarname))||' NE '||IFC(_mvartype='N',' .','''''')||' AND _'||PUT(_mvarnum,Z7.)||' EQ '||
                    IFC(_pvartype='N',' .','''''')||' THEN _CHK_'||PUT(_mvarnum,Z7.)||'=''BAD'';';
    /* Join */
    IF _CHK_STR2_  ='' THEN _CHK_STR2_  =                        TRIM(LEFT(_TMP_STR_));
                       ELSE _CHK_STR2_  =TRIM(_CHK_STR2_  )||' '||TRIM(LEFT(_TMP_STR_));
  /* Check 結果を表示する */
    /* Temporary String */
    _TMP_STR_='IF _CHK_'||PUT(_mvarnum,Z7.)||' NE '''' THEN PUT '''||TRIM(LEFT(_mvarname))||''';';
    /* Join */
    IF _RESULT_STR_='' THEN _RESULT_STR_=                        TRIM(LEFT(_TMP_STR_));
                       ELSE _RESULT_STR_=TRIM(_RESULT_STR_)||' '||TRIM(LEFT(_TMP_STR_));
  END;
END;
rc=CLOSE(dsidm);
rc=CLOSE(dsidp);
/* Set Macro Values */
CALL SYMPUT('_PNVARS_',TRIM(LEFT(PUT(_pnvars_,BEST32.))));
CALL SYMPUT('_KEEP_STR_  ',TRIM(_KEEP_STR_  ));
CALL SYMPUT('_RETAIN_STR_',TRIM(_RETAIN_STR_));
CALL SYMPUT('_RENAME_STR_',TRIM(_RENAME_STR_));
CALL SYMPUT('_TRANS_STR_ ',TRIM(_TRANS_STR_ ));
CALL SYMPUT('_CHK_STR1_  ',TRIM(_CHK_STR1_  ));
CALL SYMPUT('_CHK_STR2_  ',TRIM(_CHK_STR2_  ));
CALL SYMPUT('_RESULT_STR_',TRIM(_RESULT_STR_));
RUN;
/* Create Main DS */
DATA     RESDS.&_DOMAIN_;
 SET     MDLDS.&_DOMAIN_(WHERE=(1=2))
         PREDS.&_DOMAIN_(KEEP=&_KEEP_STR_ RENAME=(&_RENAME_STR_)) END=_EOF;
/* Keep */
KEEP     &_KEEP_STR_  ;
/* Retain Variables for Check Result Stored */
RETAIN   &_RETAIN_STR_;
/* Transfer */
&_TRANS_STR_
/* Check1 */
&_CHK_STR1_
/* Check2 */
&_CHK_STR2_
/* Check Result */
IF _EOF THEN DO;
  PUT '*** Please Check following Variable(s) ***';
  &_RESULT_STR_
  PUT '**********************************************';
END;
RUN;
/* Create Temporary 1 Record Dataset */
DATA     _TMPDS_;
_TMPVAR_=.;
RUN;
/* Create Supp. DS */
DATA     RESDS.SUPP&_DOMAIN_;
 SET     MDLDS.SUPP(WHERE=(1=2)) _TMPDS_;
KEEP     STUDYID
         RDOMAIN
         USUBJID
         IDVAR
         IDVARVAL
         QNAM
         QLABEL
         QVAL
         QORIG
         QEVAL
        ;
/* Check Var. Array */
ARRAY _VAR_ERROR_(&_PNVARS_);
/* Length */
LENGTH _varc $200;
/* Fetch DS */
dsidp=OPEN("PREDS.&_DOMAIN_",'I');
dsidm=OPEN("MDLDS.&_DOMAIN_",'I');
DO UNTIL(frc NE 0);
  frc=FETCH(dsidp);
  IF frc = 0 THEN DO;
    STUDYID=GETVARC(dsidp,VARNUM(dsidp,'STUDYID'));
    IF STUDYID NE GETVARC(dsidp,VARNUM(dsidp,'STUDYID')) THEN _VAR_ERROR_(VARNUM(dsidp,'STUDYID'))=1;
    USUBJID=GETVARC(dsidp,VARNUM(dsidp,'USUBJID'));
    IF USUBJID NE GETVARC(dsidp,VARNUM(dsidp,'USUBJID')) THEN _VAR_ERROR_(VARNUM(dsidp,'USUBJID'))=1;
    RDOMAIN=GETVARC(dsidp,VARNUM(dsidp,'DOMAIN'));
    IF RDOMAIN NE GETVARC(dsidp,VARNUM(dsidp,'DOMAIN ')) THEN _VAR_ERROR_(VARNUM(dsidp,'DOMAIN '))=1;
    IF VARNUM(dsidp,"&_DOMAIN_.SEQ")>0 THEN DO;
      IDVAR   =UPCASE("&_DOMAIN_.SEQ");
```

```
        IDVARVAL=TRIM(LEFT(PUT(GETVARN(dsidp,VARNUM(dsidp,"&_DOMAIN_.SEQ")),BEST32.)));
        IF IDVARVAL NE TRIM(LEFT(PUT(GETVARN(dsidp,VARNUM(dsidp,"&_DOMAIN_.SEQ")),BEST32.)))
                                                          THEN _VAR_ERROR_(VARNUM(dsidp,"&_DOMAIN_.SEQ"))=1;
      END; ELSE
      DO;
        IDVAR   ='';
        IDVARVAL='';
      END;
      DO _I_ = 1 TO &_PNVARS_;
        _pvarnum  =_I_;
        _pvarname =VARNAME (dsidp,_pvarnum);
        _pvartype =VARTYPE (dsidp,_pvarnum);
        _pvarlen  =VARLEN  (dsidp,_pvarnum);
        _pvarlabel=VARLABEL(dsidp,_pvarnum);
        _pvarfmt  =VARFMT  (dsidp,_pvarnum);
        _pvarinfmt=VARINFMT(dsidp,_pvarnum);
        /* Format Set */
        _varfmt  =_pvarfmt  ;
        IF _varfmt  ='' THEN DO;
          IF _pvartype='N' THEN _varfmt  ='BEST32.'; ELSE _varfmt  ='$CHAR200.';
        END;
        _varfmt  =TRIM(LEFT(_varfmt));
        /* When Variable Not Exist on Main DS */
        IF VARNUM(dsidm,_pvarname)=0 THEN DO;
          /* Get Value */
          IF _pvartype='N' THEN DO;
            _varn=GETVARN(dsidp,_pvarnum);
            _varc='';
            IF _varn NE . THEN DO;
              /* Set Values */
              QNAM  =UPCASE(_pvarname);
              QLABEL=TRIM(LEFT(_pvarlabel));
              QVAL  =TRIM(LEFT(PUTN(_varn,_varfmt)));
              IF QNAM   NE UPCASE(_pvarname)
              OR QLABEL NE TRIM(LEFT(_pvarlabel))
              OR QVAL   NE TRIM(LEFT(PUTN(_varn,_varfmt))) OR QVAL EQ '' THEN _VAR_ERROR_(_pvarnum)=1;
              OUTPUT;
            END;
          END; ELSE
          DO;
            _varn= .;
            _varc=GETVARC(dsidp,_pvarnum);
            IF _varc NE '' THEN DO;
              /* Set Values */
              QNAM  =UPCASE(_pvarname);
              QLABEL=TRIM(LEFT(_pvarlabel));
              QVAL  =TRIM(LEFT(PUTC(_varc,_varfmt)));
              IF QNAM   NE UPCASE(_pvarname)
              OR QLABEL NE TRIM(LEFT(_pvarlabel))
              OR QVAL   NE TRIM(LEFT(PUTC(_varc,_varfmt))) OR QVAL EQ '' THEN _VAR_ERROR_(_pvarnum)=1;
              OUTPUT;
            END;
          END;
        END;
      END;
    END;
  END;
  PUT '*** Please Check following Variable(s) ***';
  DO _I_ = 1 TO &_PNVARS_;
    IF _VAR_ERROR_(_I_)=1 THEN DO;
      _pvarname =VARNAME (dsidp,_I_);
      _pvarname_len=LENGTH(_pvarname);
      PUT _pvarname $VARYING200. _pvarname_len;
    END;
  END;
  PUT '*****************************************';
  /* Close DS */
  rc=CLOSE(dsidp);
  rc=CLOSE(dsidm);
  RUN;
  /* Delete Temporary 1 Record Dataset */
  PROC DATASETS NOLIST;
  DELETE  _TMPDS_/ MT=DATA;
  QUIT;
%MEND;


/* Execute */
OPTIONS MPRINT MLOGIC;
%AutoSDTM(AE);
DATA RESDS.SUPPAE;
 SET RESDS.SUPPAE;
QORIG='DERIVED';
QEVAL='SPONSOR';
RUN;
```