初/中級者が陥りやすいプログラミングエラー

井上 貴博
（ノバルティス ファーマ株式会社）

Programming errors that easily occur on beginner/intermediate SAS programmers

Takahiro Inoue
Data Sciences & Scientific Operations Department, Novartis Pharma K.K.

要旨

初/中級者には理解し難く陥りやすいプログラミングエラーは多数ある。エラー発見・解消テクニックを紹介し、浮動小数点の丸め誤差、PDV など、理解していないと意図した結果が得られない事例を紹介する。

キーワード：プログラミング、テクニック、エラー、浮動小数点、丸め誤差、PDV、Program Data Vector

1. はじめに

SAS のパワーユーザーには当たり前になっているプログラミングエラーでも、SAS を構文を覚え慣れ始めた初/中級者に理解し難く、陥りやすいエラーは多数ある。しかし、そのテクニックや内部処理が体系的にまとめられた資料は少ない。そこで、本稿ではエラー発見・解消に役立つテクニックを紹介し、その後、浮動小数点の丸め誤差、Program Data Vector など、内部処理を理解していないと意図した結果が得られない事例を紹介する。なお、本稿は、SAS OnDemand for Academics を使用した。処理結果を再現できるようにプログラムも記載しているが、他の動作環境を使用する場合、結果が異なる場合がある。

2. ログの確認

SAS は、プログラムを実行するとログを生成する。このログには以下の情報が含まれている。

- 実行されたプログラム
- SAS データセットに関する情報(変数やオブザベーション数など)
- プログラム実行中に発生した警告・エラーメッセージなど
- プログラム実行に要した時間

プログラムが正常に実行されてない場合、ログに WARNING や ERROR が表示される。その原因やプログラム修正のためのヒントは、ログから見つけることができ、プログラムの実行後は、意図した通りに実行され結果が得られているかログを確認する必要がある。
詳細は後項に記載するが、自動マクロ変数で ERROR、WARNING の有無を確認できる。しかし、ERROR、WARNING ではない NOTE でも注意すべきメッセージが多数ある。ログに確認する際のキーワードの一例を以下に示す。

ERROR	WARNING	BY 値を繰り返すデータセット (バージョンにより「BY 値」とスペースがないものもある)		
欠損値を含んだ	初期化され	変換	範囲外	による割り算がありました
ゼロ割り	無効な	LOST CARD	処理を中止	文字を超えています
実行を中止	切り捨て	表示が制限	ステートメントの余分な情報を無視します	
既にライブラリ	出力形式によって小数点がシフト			変数がありません
存在しません	演算式を計算できなかった			

3. エラーのタイプ

エラーが発生したプログラムを修正するには、その発生したエラーのタイプを知ることで、そのエラーのタイプに合わせたプログラム修正ができる。SAS 9.4 Language Reference [1] の Types of Errors in SAS の項では、以下のエラーのタイプが紹介されている。

- 構文エラー(Syntax errors)
- セマンティックエラー(Semantic errors)
- 実行時エラー(Execution-time errors)
- データエラー(Data errors)
- マクロ関連エラー(Macro-related Errors)※

  ※マクロコンパイル時および実行時に発生する。または、マクロから生成されたステートメント構文等のエラーだが、本稿での解説は割愛する。

また、Language Reference では紹介されていないが、論理エラー (Logic errors) と呼ばれるエラーがあり、以下にそれぞれの具体例を示す。

3.1. 構文エラー(Syntax errors)

SAS 構文エラーに沿っていない場合に発生するエラーで、以下のようなエラーがある。

- スペルミス
- 引用符(“ ”,’ ’)の不一致
- セミコロン(;)がない
- 無効なステートメントオプション
- 無効なプロシジャオプション

サンプル プログラム	ログ
date temp ; 　set a ; run ;	60　　date temp ; 　　____ 　　14 WARNING 14-169: dateはシンボルDATAのスペルミスであると判断しました。 61　　set a ; 62　　run ; NOTE: データセットWORK.Aから1オブザベーションを読み込みました。 NOTE: データセットWORK.TEMPは1オブザベーション、1変数です。 NOTE: DATAステートメント処理(合計処理時間):
data を date とスペルミス。SAS では一部のスペルミスは自動的に変更し実行する。	
data temp 　set a ; run ;	60　　data temp 61　　set a ; 　　___ 　　56 ERROR 56-185: オプションDATASTMTCHK=COREKEYWORDSの時、DATAステートメントにSETは使用できません。 　　DATAステートメントにセミコロンが抜けていないか、 　　またはDATASTMTCHK=NONEを使用してください。 62　　run ; NOTE: エラーが発生したため、このステップの処理を中止しました。 NOTE: DATAステートメント処理(合計処理時間):
data temp ;のセミコロン(;)が抜けていた。SAS のバージョンによっては、temp, set, a の 3 データセットが作成されることがあり、SET 文の参照データ a が上書きされる場合もある。	

3.2. セマンティックエラー(Semantic errors)

構文規則上は正しくても、使用意図として間違っているような場合に発生する。

サンプルプログラム	ログ
data temp ; 　array all(*) x1 - x5 ; 　all = 3 ; run ;	61　　data temp ; 62　　array all(*) x1 - x5 ; 63　　all = 3 ; ERROR: 配列allへの参照が正しくありません。 64　　run ; NOTE: エラーが発生したため、このステップの処理を中止しました。 WARNING: データセットWORK.TEMPは未完成です。このステップは、0オブザベーション、 　　5変数で停止しました。 WARNING: このステップを中止したため、データセットWORK.TEMPを置き換えていません。
array all(*) x1 - x5 ;および all = 3 ;を各行で見れば、構文規則上、問題はない。 しかし、all は配列として定義されているため、変数として all は使用できない。	

3.3. 実行時エラー(Execution-time errors)

コンパイル済みのプログラムを実行する際に発生する以下のようなエラーがある。

- 関数の無効な引数
- 無効な数学演算(0 除算)
- BY 変数が Sort されていない
- ARRAY 配列の引数が範囲外
- INFINE/FILE ステートメントで指定したファイルのオープンおよびクローズエラー
- INPUT 文での無効な引数(文字変数を数値変数として抽出)

3.4. データエラー(Data errors)

ステートメントは正しいが、データ値がステートメントに適さない場合に発生する。

サンプルプログラム	ログ
data temp ; dtc = '2017-02-31' ; dt = input(dtc, yymmdd10.) ; run ;	61 data temp ; 62 dtc = '2017-02-31' ; 63 dt = input(dtc, yymmdd10.) ; 64 run ; NOTE: 関数INPUT(行 63 カラム 7)の引数は無効です。 dtc=2017-02-31 dt=. _ERROR_=1 _N_=1 NOTE: 以下の箇所で演算式を計算できなかったため、結果を欠損値に設定しました。 (回数)(行:カラム) 1 63:7 NOTE: データセットWORK.TEMPは1オブザベーション、2変数です。
構文に問題はない。しかし、2017-02-31 は日付として存在しないため、引数が無効となり Null 値が設定される。ログには ERROR, WARNING ではなく NOTE として表示されないため、注意が必要である。	

なお、2050-02-01 のように日付として存在するが、未来の日付が入力された場合、SAS のログには表示されない。そのため、必要に応じて、入力データが適切であるかデータレビュー等を行う。

3.5. 論理エラー(Logic errors)

ここまで紹介したエラー以外に SAS のログからは発見できない論理エラーがある。例えば、以下のように年齢(age)を'10 代'、'20 代'、'30 代以上'と分類するプログラムを作成したとする。プログラム(誤)では比較演算子に「=」がなく、age = 10, 20, 30 は、agec が Null 値となってしまう。

プログラム(誤)	プログラム(正)
if 10 < age < 20 then agec = '10 代' ; else if 20 < age < 30 then agec = '20 代' ; else if 30 < age then agec = '30 代以上' ;	if 10 <= age < 20 then agec = '10 代' ; else if 20 <= age < 30 then agec = '20 代' ; else if 30 <= age then agec = '30 代以上' ;

また、数値変数の Null 値は最小値として扱われるため、IF 文の比較演算には注意が必要である。

サンプルプログラム	ログ
data temp ; input age ; if age <= 1 then agec = 1 ; else if 2 < age then agec = 2 ; put age = agec = ; cards; . 1 2 ; run ;	60 data temp ; 61 input age ; 62 if age <= 1 then agec = 1 ; 63 else if 2 < age then agec = 2 ; 64 put age = agec = ; 65 cards; age=. agec=1 age=1 agec=1 age=2 agec=. NOTE: データセットWORK.TEMPは3オブザベーション、2変数です。 NOTE: DATAステートメント処理(合計処理時間): 処理時間 0.00 秒 ユーザーCPU時間 0.00 秒
age=. を agec=. とする場合 if age <= 1 then agec = 1 ;を if . < age <= 1 then agec = 1 ;とする。	

論理エラーとして紹介した 2 つの例では、ログに ERROR, WARNING は表示されず、通常の NOTE のみである。そのため、作成したプログラムおよびデータが意図した結果になっているか確認の必要がある。

4. エラー発見・解消のためのテクニック

SAS では、次のコンポーネントをデバッグに利用することができる。これを組み合わせることにより、検証時のデータ損失の予防や、エラー発生箇所の特定を容易にすることができる。

- オプション(システム/データセット)
- ステートメント
- 関数および CALL ルーチン
- 自動変数

以下にそれぞれの具体例を示す。

4.1. オプション(システム/データセット)を利用する

obs = 0 と noreplace オプションを同時に利用することで、データセットや外部データを読み込まず Data step を実行し、構文エラーの確認ができ、存在するデータセットへの上書きを防げる。設定をデフォルトに戻すには「options obs = max replace ;」と記述する。

サンプルログ(およびプログラム)

```
60          option obs = 0 noreplace ;
61          data temp.b ;
62              set a ;
63          run ;

NOTE: データセットWORK.Aから0オブザベーションを読み込みました。
NOTE: データセットTEMP.Bは0オブザベーション、1変数です。
WARNING: NOREPLACE オプションのため、データセットTEMP.Bを置き換えていません。
NOTE: DATAステートメント処理(合計処理時間):
```

大量の処理する場合、必然的に実行時間が長くなる。そのため、FISRTOBS=および OBS=データセットオプションを利用しデータセットをサブセット化することで、実行時間を短くしコードレビューを容易にすることができる。また、WHERE=データセットオプションを利用することにより、サブグループごとにサブセット化することも可能である。

サンプルログ(およびプログラム)

```
61          data b ;
62              set a(firstobs = 10 obs = 15) ;
63          run ;

NOTE: データセットWORK.Aから6オブザベーションを読み込みました。
NOTE: データセットWORK.Bは6オブザベーション、1変数です。
NOTE: DATAステートメント処理(合計処理時間):

64          data b ;
65              set a(where = (10 < age <20)) ;
66          run ;

NOTE: データセットWORK.Aから9オブザベーションを読み込みました。
      WHERE (age>10 and age<20);
NOTE: データセットWORK.Bは9オブザベーション、1変数です。
NOTE: DATAステートメント処理(合計処理時間):
```

4.2. ステートメントを利用する

前項の方法ではデータに偏りが発生する場合がある。そのため、ステートメントを利用したサブセット化も有用である。

サンプルプログラム

```
data temp ;
    do pickup=1 to totobs by 100;
        set a point=pickup nobs=totobs;
        output;
    end;
    stop;
run;
```

また、プログラムが正しくても、不正なデータにより処理に悪影響を受けることがある。このようなエラーを防ぐために Defensive coding が必要となる。例えば、以下のプログラムは、未来日が処理に悪影響があるとして ERROR を表示している。

サンプルプログラム

```
data temp ;
    length dtc $10. ; input dtc $ ;
    dt = input(dtc, yymmdd10.) ;
    if today() < dt then
      put 'ER' 'ROR: 未来日あり' dt = yymmdd10.;
cards;
2017-02-28
2050-02-01
;
run ;
```

ログ

```
60          data temp ;
61              length dtc $10. ; input dtc $ ;
62              dt = input(dtc, yymmdd10.) ;
63              if today() < dt then
64                put 'ER' 'ROR: 未来日あり' dt = yymmd
65          cards;

ERROR: 未来日あり dt=2050-02-01
NOTE: データセットWORK.TEMPは2オブザベーション、2変数です
NOTE: DATAステートメント処理(合計処理時間):
```

4.3. 関数を利用する

関数を利用して、データセットの存在を確認することができる。以下のプログラムは、データの存在を確認し ERROR が表示している。

サンプルログ(およびプログラム)

```
61          data _null_ ;
62              zaibuzai = exist('work.no') ;
63              if zaibuzai ne 1 then
64                put 'ER' 'ROR: データセットがありません' ;
65          run ;

ERROR: データセットがありません
NOTE: DATAステートメント処理(合計処理時間):
```

また、システムオプションの設定値によっては、プログラムが意図した結果にならない。この場合、GETOPTION 関数を利用して設定値を取得する。

サンプルログ(およびプログラム)

```
63          %LET p_size=%SYSFUNC(GETOPTION(PS));
64          %PUT 現在の PAGESIZE:  &p_size;
現在の PAGESIZE:  75
65          /* PAGESIZE が 100 未満なら 100 へ変更 */
66          %MACRO ps;
67             %IF &p_size < 100 %THEN %DO;
68
69                OPTIONS ps=100;
70             %END;
71          %MEND;
72          %ps;
73          %LET p_size=%SYSFUNC(GETOPTION(PS));
74          %PUT 現在の PAGESIZE:  &p_size;
現在の PAGESIZE:  100
```

4.4. 自動変数を利用する

SAS データセットでは、いくつかの自動変数が用意されている。以下は、文字変数が不正日付で数値に変換できなかった場合、_error_を使用して_n_(データ行)を表示するプログラムである。

サンプルプログラム

```
data date ;
    length dtc $10. ;
    input dtc $ ;
cards;
2017-02-01
2017-02-30
;
run ;

data temp ;
    set date ;
    dt = input(dtc, yymmdd10.) ;
    if _error_ = 1 then
      put 'ER' 'ROR: '
          _n_ '行目に不正日付あり' dtc = ;
run ;
```

ログ

```
～ 省略 ～

68          data temp ;
69              set date ;
70              dt = input(dtc, yymmdd10.) ;
71              if _error_ = 1 then
72                put 'ER' 'ROR: '
73                    _n_ '行目に不正日付あり' dtc = ;
74          run ;

NOTE: 関数INPUT(行 70 カラム 10)の引数は無効です。
ERROR: 2 行目に不正日付ありdtc=2017-02-30
dtc=2017-02-30 dt=. _ERROR_=1 _N_=2
NOTE: 以下の箇所で演算式を計算できなかったため、結果を欠損値に
```

引数の前に「?」または「??」をつけることにより、_error_およびログの出力等を制御することができ、複数の INPUT 文を使用する際などに利用し特定の変数のみを確認することができる。

サンプルログ(およびプログラム)

```
61          data temp ;
62              set date ;
63              dt = input(dtc,? yymmdd10.) ;
64              if _error_ = 1 then
65                put 'ER' 'ROR: ' _n_ '行目に不正日付あり' dtc = ;
66          run ;

ERROR: 2 行目に不正日付ありdtc=2017-02-30
dtc=2017-02-30 dt=. _ERROR_=1 _N_=2
NOTE: データセットWORK.DATEから2オブザベーションを読み込みました。
NOTE: データセットWORK.TEMPは2オブザベーション、2変数です。
NOTE: DATAステートメント処理(合計処理時間):

68          data temp ;
69              set date ;
70              dt = input(dtc,?? yymmdd10.) ;
71              if _error_ = 1 then
72                put 'ER' 'ROR: ' _n_ '行目に不正日付あり' dtc = ;
73          run ;

NOTE: データセットWORK.DATEから2オブザベーションを読み込みました。
NOTE: データセットWORK.TEMPは2オブザベーション、2変数です。
NOTE: DATAステートメント処理(合計処理時間):
```

ログに出力される ERROR、WARNING メッセージを格納する自動マクロ変数が用意されている。

サンプルログ(およびプログラム)

```
60          dat temp
            ___
            14
61              set date ;
                ___
                56
WARNING 14-169: datはシンボルDATAのスペルミスであると判断しました。

ERROR 56-185: オプションDATASTMTCHK=COREKEYWORDSの時、DATAステートメントにSETは使用できません。
              DATAステートメントにセミコロンが抜けていないか、
              またはDATASTMTCHK=NONEを使用してください。

62          run;

NOTE: エラーが発生したため、このステップの処理を中止しました。
NOTE: DATAステートメント処理(合計処理時間):
      [illegible]時間          00 秒

63          %PUT &SYSWARNINGTEXT;
14-169: datはシンボルDATAのスペルミスであると判断しました。
64          %PUT &SYSERRORTEXT;
56-185: オプションDATASTMTCHK=COREKEYWORDSの時、DATAステートメントにSETは使用できません。
DATAステートメントにセミコロンが抜けていないか、          またはDATASTMTCHK=NONEを使用してください。
```

これにより、ログに ERROR、WARNING が存在するか確認することができる。

5. 浮動小数点の丸め誤差

5.1. 2 進算術演算と 10 進算術演算の差異

コンピュータは、有限精度の 2 進算術演算を使用している。通常、厳密な 2 進表現がない数値を扱う場合、コンピュータによって生成される結果は 10 進算術演算で生成される結果とわずかに異なる。例えば、10 進値の 0.1 および 0.3 には厳密な 2 進表現はない。10 進算術演算の 3*0.1 は厳密に 0.3 になるが、2 進算術演算ではこの等式は成り立たない。次の例に示すように、これらの 2 つの値を SAS に書き込むと、同じ値のように見える。ただし、差異を計算すると、値が異なることが分かる。

サンプルログ(およびプログラム)

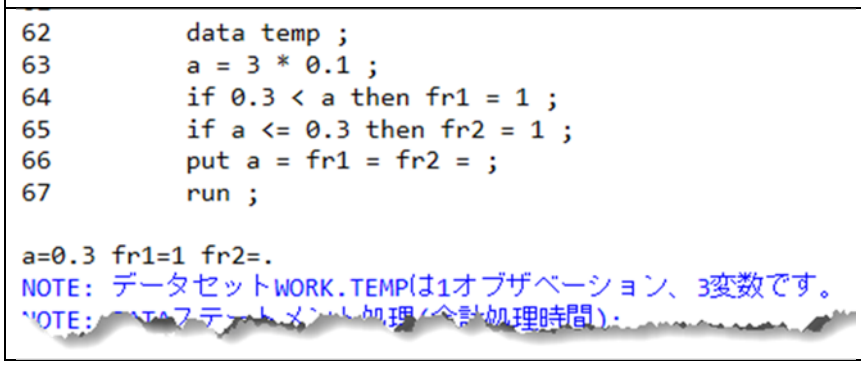
```
61          data _null_;
62             point_three=0.3;
63             three_times_point_one=3*0.1;
64             difference=point_three - three_times_point_one;
65             put point_three= ;
66             put three_times_point_one= ;
67             put difference= ;
68          run;

point_three=0.3
three_times_point_one=0.3
difference=-5.55112E-17
NOTE: DATAステートメント処理(合計処理時間):
      処理時間          0.00 秒
      [illegible]          00 秒
```

SAS では、IEEE 準拠の浮動小数点表現 (倍精度浮動小数点表現) を用いて数値を取り扱っている。この浮動小数点表現は、効率よく実数を表現でき、また表現可能なレンジが広いという利点があるが、10 進数を 2 進数で格納するため、10 進数と 2 進数のマッピングに誤差が生じる場合がある。とくに 10 進数では有限小数点であっても、2 進数では無限小数点(循環小数)となる場合があり、コンピュータ内部では、物理的な格納つまり有限桁に変換され、精度が失われる。

5.2. 丸め誤差の及ぼす影響

前項では、3*0.1 が厳密に 0.3 にならない例を示した。このような差異がある場合、IF 文で意図した結果にならない。例えば、以下のプログラムのように 0.3 超のとき fr1 に 1 を、0.3 以下のとき fr2 に 1 を設定する IF 文を作成した場合、意図した結果が得られないことがある。

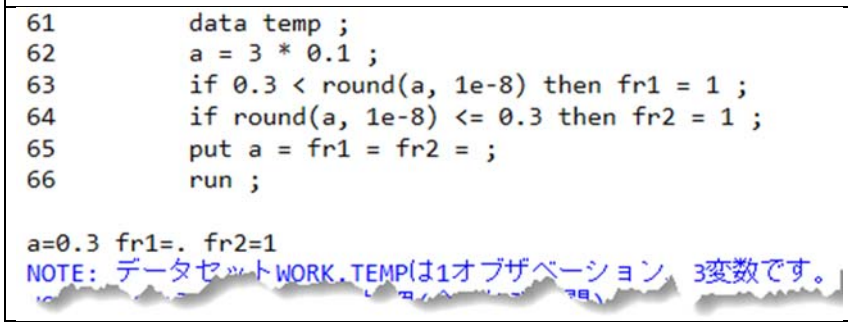
サンプルログ(およびプログラム)



5.3. 回避策

SAS は、プログラムで"0.3"を定数として示した場合、値は標準の入力形式で 3/10 として計算される。変数は前項で示した丸め誤差が生じるため、ROUND 関数を使用し丸め処理を行う必要がある。

サンプルログ(およびプログラム)



本稿では 3*0.1 の算術式を例としたが、SAS が別の入力形式を使用して変数を読み込む場合や、SAS 以外のプログラムで変数を読み込む場合も、10 進値の厳密な値が生成されない可能性がある。また、厳密な 2 進表現ではない数値が計算に含まれている場合や、浮動小数点表現が異なる別の動作環境にデータセットを移植する場合も、正確な結果を得られない可能性がある。

5.4. ROUND 関数の注意点

通常、ROUND 関数は、結果の有効桁数が 9 以下で、次のどの条件にも該当しない場合、10 進算術演算で期待される結果を生成する。

- 丸めの単位が整数である。
- 丸め単位が 1e-15 以上の 10 のべき乗である。
  (丸め単位が 1 よりも小さい場合、ROUND は、丸め単位の逆数が 10 のべき乗との違いが最下位から第 3 または第 4 ビッドまででであれば、丸め単位を 10 のべき乗として扱う)
- 10 進算術演算で期待される結果が小数第 4 位以下である。

多くのパッケージソフトウェアや表計算ソフトウェアでは、内部的に暗黙の FUZZ 値を用いてこの問題を回避しているが、以前の SAS の ROUND 関数では引数の値をそのまま用いて処理を行うため、意図した結果が得ないことが多かった。このような問題は、FUZZ 関数を利用した構文を作成することにより、回避できた。[例 round(a, 1e-5)： round((a+(sign(a)*1e-10)),1e-5)]

6. Program Data Vector (PDV)

SAS は、Program Data Vector (以下 PDV) に一度新しいデータセットを作成する。どのように PDV で初期設定され、情報が保持されているかを理解することが正確なプログラムを書く上で必要不可欠となる。本項では、事例を使って PDV でどのような処理されているか紹介する。

6.1. 使用するデータセット

以下のデータを使用し、PDV が影響する事例を示す。

Dataset: A	
id	var1
a1	1
a2	2

```
/* プログラム */
data A ;
    input id $ var1 ;
cards;
a1 1
a2 2
;
run ;
```

Dataset: B		
id	var1	var2
b1	1	101
b2	2	202

```
/* プログラム */
data B ;
    input id $ var1 var2 ;
cards;
b1 1 101
b2 2 202
;
run ;
```

Dataset: C		
id	var1	var2
a1	1	.
a2	2	.
b1	1	101
b2	2	202

```
/* プログラム */
data C ;
    set A B ;
run ;
```

6.2. PDV が影響する事例

Dataset: C に対して、if var1 = 1 then var2 = 0 ； という IF 文を使用し、Dataset: D を作成した例を示す。この例は意図通りの結果を得られている。

Dataset: D			
id	var1	var2	var3
a1	1	0	0
a2	2	.	.
b1	1	0	0
b2	2	202	.

```
/* プログラム */
data D ;
    set C ;
    if var1 = 1 then var2 = 0 ;
    if var1 = 1 then var3 = 0 ;
run ;
```

次に Dataset: C を使わず、Dataset: A と B を SET 文で読み込み、同様の IF 文を使用した Dataset: E を作成した例を示す。

Dataset: E			
id	var1	var2	var3
a1	1	0	0
a2	2	0	.
b1	1	0	0
b2	2	202	.


```
/* プログラム */
data E ;
    set A B ;
    if var1 = 1 then var2 = 0 ;
    if var1 = 1 then var3 = 0 ;
run ;
```

Dataset: E では、id = a2 の var2 が 0 になっていることに注目していただきたい。PDV の処理を理解されていない方にとっては、意図していない結果になっていると思う。

6.3. 処理プロセス・結果

PDV では、1 オブザベーション毎にデータが保持され処理される。次のオブザベーションの処理に移った際、新規変数はリセットされ(Null 値に変換)され、読み込み変数は読み込みデータで上書きされる。

前項で示した Dataset: E の場合、var2 は読み込み変数、var3 は新規変数である。新規変数である var3 は、オブザベーションが変わるたびに Null 値にリセットされる。その上で、IF 文の条件に合致した id=a1, a2 は var3 に 0 が代入され、条件が合致しない id=a2, b2 は Null 値が残る。

var2 は読み込み変数だが、Dataset: A に var2 がない。そのため、Dataset: B 由来の id = b1, b2 では変数が読み込みデータで上書きされるが、Dataset: A 由来の id = a1, a2 は読み込みされず上書きされない。そのため、id = a2 では上書きされず、前オブザベーションのデータが残り、0 となった。

6.4. PDV が影響した MERGE 文の事例

前項では SET 文の事例を示したが MERGE 文でも同様に PDV が影響し、意図しない結果が出る場合がある。

Dataset: F	
id	var1
1	XXXX
1	XXXX
2	YYYY
2	YYYY

```
/* プログラム */
data F ;
    input id var1 $ ;
cards ;
1 XXXX
1 XXXX
2 YYYY
2 YYYY
;
run ;
```

Dataset: G		
id	var1	var2
1	ZZZZ	7777
2	ZZZZ	9999

```
/* プログラム */
data G ;
    input id var1
$ var2 ;
cards ;
1 ZZZZ 7777
2 ZZZZ 9999
;
run ;
```

Dataset: H		
id	var1	var2
1	ZZZZ	7777
1	XXXX	7777
2	ZZZZ	9999
2	YYYY	9999

```
/* プログラム */
data H ;
    merge F G ;
    by id ;
run ;
```

var2 は Dataset: F になく Dataset: G にのみ存在するため、全てのオブザベーションで読み込みが行われる。Dataset: F, G の両方に存在する var1 は、Datasets: F のデータが読み込まれた後、Dataset: G のデータで上書きされる。その際、Dataset: G のオブザベーションのみ上書きされるため、var1 は、各 ID の 1 行目が Dataset: G、2 行目が Dataset: F のデータとなる。

6.5. PDV に対する対策

通常、前項の事例で示したような IF 文で存在する変数の上書き、変数が重複する状態での MERGE は行わないほうが適切である。今回は、PDV の処理プロセスの説明のため、このようなプログラムを事例として紹介した。

もし、Dataset: E で示したように SET 文と IF 文を同一 Data step 内で処理する場合、以下のプログラムで Dataset: D と同様の結果を得られることができる。

```
/* プログラム */
data E_ ;
    set A(in = A) B ;
    if var1 = 1 then var2 = 0 ;
      else if A then var2 = . ;  /* var2 がない dataset:A は var2 を初期化 */
    if var1 = 1 then var3 = 0 ;
run ;
```

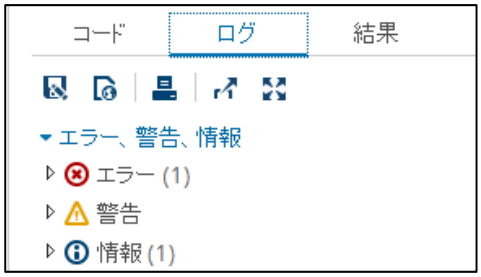
変数が重複する状態での MERGE に関しては、「options msglevel = i ;」を指定することにより、ログに重複した変数の情報を表示できる。設定をデフォルトに戻すには「options msglevel = N ;」と記述する。

サンプルログ(およびプログラム)

```
79          options msglevel=i;
80          data H ;
81          merge F G ;
82          by id ;
83          run ;

INFO:  変数 var1 (データセット WORK.F) はデータセットWORK.Gによって上書きされます。
NOTE: データセットWORK.Fから4オブザベーションを読み込みました。
NOTE: データセットWORK.Gから2オブザベーションを読み込みました。
```

7. おわりに

SAS のログでは、ERROR、WARNING が赤や緑などで目立つように表示される。また、SAS OnDemand for Academics などをはじめとする最新の SAS では、エラー、警告などの数をカテゴリで表示できるようになった。しかし、本稿で紹介した事例にようにログの NOTE でも注意すべきメッセージでも多くある。意図した結果を得るには、まずはログの確認が重要である。また、どういったエラーがあるか内容を理解し、そのエラーに合わせたプログラミング、得られた結果が意図したものであるか、実際に入力されたデータ、出力されたデータを確認することが必要である。



また、SAS 初心者が SAS を学習する際、構文を学習することに重点を置くことが多い。しかし、本稿で紹介した事例のように、SAS がコンパイルと実行段階で、どのようにデータを処理するかを理解せずプログラミングした場合、意図した結果を得られないことがある。SAS の構文を覚えた初/中級者は、正しい結果を作成できるように本稿で紹介したような SAS 内部のデータ処理に対しても、理解を深めていただきたい。

8. 参考文献

[1] SAS 9.4 Language Reference: Concepts, Sixth Edition
< https://support.sas.com/documentation/cdl/en/lrcon/69852/HTML/default/viewer.htm#titlepage.htm>
(参照: 2017-6-19)

[2] SAS Technical News
< http://www.sas.com/offices/asiapacific/japan/periodicals/technews/>
(参照: 2017-6-19)

[3] SAS 9.4 関数と CALL ルーチン　リファレンス, 第 4 版
<http://support.sas.com/documentation/cdl_alternate/ja/lefunctionsref/67960/HTML/default/titlepage.htm>
(参照: 2017-6-19)

[4] SAS FAQ
<https://www.sas.com/ja_jp/support/technical/faq.html>
(参照: 2017-6-19)

[5] Essentials of the Program Data Vector (PDV): Directing the Aim to Understanding the DATA Step!
Arthur Xuejun Li, City of Hope National Medical Center, Duarte, CA
SAS Global Forum 2013, Paper 125-2013

[6] The Use and Abuse of the Program Data Vector
Jim Johnson, Ephicacy Corporation, North Wales, PA, USA
SAS Global Forum 2012, Paper 255-2012