

FCMPにおけるDictionary/Hash Object. SQLとハッシュを繋ぐもの

イーピーエス株式会社
森岡 裕

本発表は中級者向け以上です.

以下の前提知識については本発表では説明しません

初級者は一度に理解しようと聴くのではなく, イメージをつかむ感じで

【前提知識】

FCMPプロシジャ(ユーザー定義関数を作成するプロシジャ)

ハッシュオブジェクト

SQL

```
%macro get(master=,key=,var=);
%let name = &sysindex;
%let qkey = %sysfunc( tranwrd( %str("&key") , %str( ) , %str(",") ) );
if 0 then set &master(keep= &key &var);
if _N_=1 then do;
declare hash h&name.(dataset:"&master(keep= &key &var)", duplicate:'E');
h&name..definekey(&qkey);
h&name..definedata(all:'Y');
h&name..definedone();
end;
if h&name..find() ne 0 then do;
call missing(of &var);
end;
%mend get;
```

%Getマクロ

結合と存在確認の2機能

```
%macro chk(master=,key=,flg=);
%let name = &sysindex;
%let qkey = %sysfunc( tranwrd( %str("&key") , %str( ) , %str(",") ) );
if 0 then set &master(keep= &key);
if _N_=1 then do;
declare hash h&name.(dataset:"&master(keep= &key)", multidata:'Y');
h&name..definekey(&qkey);
h&name..definedone();
end;
&flg = ifc(h&name..check()=0,"Y","N");
%mend chk;
```

%Chkマクロ

keyが一意になっていることを前提として、他のデータセットから値を取得するマクロ %get

keyが一意になってなくても動き、keyが存在するかをYNで返すマクロ %chk

この2つのマクロがあれば、例えばADSLなどはほぼ1ステップで作成可能。

```
data adsl(label="");  
set sdtm.dm;
```

```
VISIT="Day 1";
```

```
%get(master=EX,key=USUBJID VISIT,var=EXDAT)
```

```
TRTSDT=input(EXDAT,?? Yymmdd10.);
```

```
CMCAT="CONCOMITANT MEDICATION";
```

```
%chk(master=cm,key=usubjid cmcat,fl=CMFL);
```

```
.....
```

```
run;
```

findメソッドと

Checkメソッドをマクロ化してしまえばSDTM/ADaM
のデータハンドリングは、(乱暴に言って)ほぼそ
れで勝負がつく

ハッシュオブジェクトを綺麗にマクロ化すると
ハンドリング周りのだいたいの問題はクリアされるが全てがそれで解決するわけではない。
たとえば...

- 通常のハッシュオブジェクトはデータステップ内の記述のため
SQLプロシジャの中で利用できないことの不便さ
- whereステートメント内で作動させれない。
例えば1億オブザベーションのデータセットAについて,
数百obsのデータセットBにIDがある症例だけをwhereで絞りたいといった状況
- マクロを減らしたい. あるいは複雑なネスト構造のマクロ内での使用時に,
見通しをよくするためにマクロに依らずにハッシュオブジェクトを
使いたいといった状況

FCMPプロシジャでユーザー定義関数を作成できるが

その中で、 ハッシュオブジェクトとディクショナリオブジェクトが利用できる

今日はその話をしてみたいです

```
data dm;
```

```
USUBJID="A-001";SUBJID="001";RFSTDTC="2024-09-01";RFENDTC="2024-10-01";output;
```

```
USUBJID="A-002";SUBJID="002";RFSTDTC="2024-09-02";RFENDTC="2024-10-02";output;
```

```
USUBJID="A-003";SUBJID="003";RFSTDTC="2024-09-03";RFENDTC="2024-10-03";output;
```

```
run;
```

VIEWTABLE: Work.Dm

	USUBJID	SUBJID	RFSTDTC	RFENDTC
1	A-001	001	2024-09-01	2024-10-01
2	A-002	002	2024-09-02	2024-10-02
3	A-003	003	2024-09-03	2024-10-03

```
data raw_ae;
```

```
SUBJID="001";AETERM="Vomit";output;
```

```
SUBJID="003";AETERM="Vomit";output;
```

```
SUBJID="005";AETERM="Vomit";output;
```

```
run;
```

VIEWTABLE: Work.Raw_ae

	SUBJID	AETERM
1	001	Vomit
2	003	Vomit
3	005	Vomit

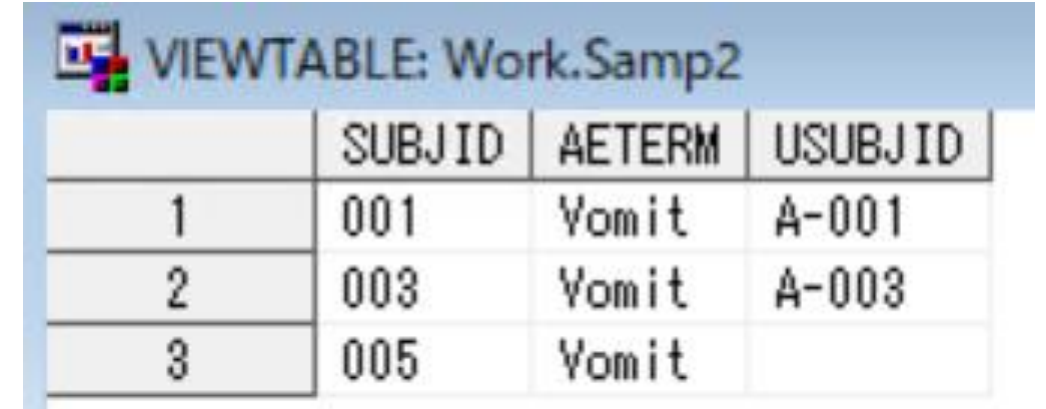
VIEWTABLE: Work.Wk1

	USUBJID	SUBJID	AETERM
1	A-001	001	Vomit
2	A-003	003	Vomit
3		005	Vomit

DMドメインからSUBJIDをキーにして
USUBJIDを取得するような処理を考
えてみる

素直に平文で書くなら

```
data samp1;  
set raw_ae;  
if 0 then set dm(keep=SUBJID USUBJID);  
if _N_ =1 then do;  
    declare hash h1(dataset:"dm");  
    h1.definekey("SUBJID");  
    h1.definedata("USUBJID");  
    h1.definedone();  
end;  
if h1.find() ne 0 then call missing(of  
USUBJID);  
run;
```



	SUBJID	AETERM	USUBJID
1	001	Vomit	A-001
2	003	Vomit	A-003
3	005	Vomit	

先に紹介したマクロを使えば1行

```
data samp2;  
    set raw_ae;  
    %get(master=dm, key=SUBJID, var=USUBJID);  
run;
```

これを関数で考える

```
proc fcmp outlib=work.functions.common;  
/*SUBJIDをいれるとDMドメインからUSUBJIDを返す関数*/  
function usubjid(SUBJID $) $;  
  declare hash h1(dataset: "DM");  
  rc=h1.definekey("SUBJID");  
  rc=h1.definedata("USUBJID");  
  rc=h1.definedone();  
  if h1.find() eq 0 then return(USUBJID);  
  else return("");  
endsub;  
quit;
```

👉 え... 簡単すぎ...

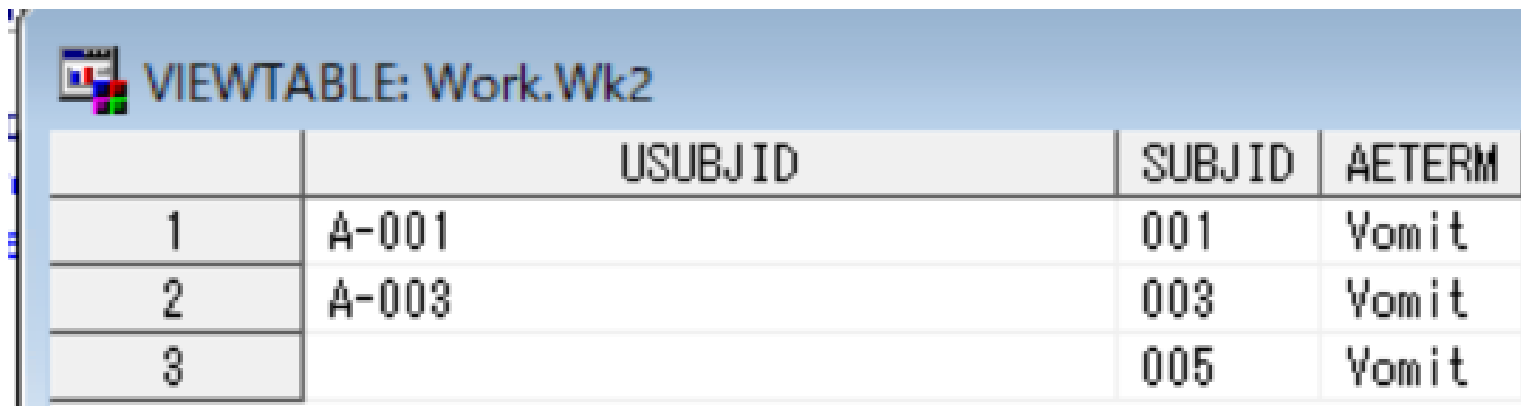
/*使ってみる */

```
data wk1;  
length SUBJID AETERM USUBJID $200.;  
set raw_ae;  
USUBJID=usubjid(SUBJID);  
run;
```

VIEWTABLE: Work.Wk1			
	SUBJID	AETERM	USUBJID
1	001	Vomit	A-001
2	003	Vomit	A-003
3	005	Vomit	

え, えすきゅーえるの中で ハッシュオブジェクトがうごいとる————

```
proc sql noprint;  
  create table wk2 as  
    select SUBJID,  
           usubjid(SUBJID) as USUBJID  
           , AETERM  
  from raw_ae;  
quit;
```



	USUBJID	SUBJID	AETERM
1	A-001	001	Vomit
2	A-003	003	Vomit
3		005	Vomit

```
data wk3;  
set raw_ae;  
where usubjid(SUBJID) ="A-001";  
run;
```



VIEWTABLE: Work.Wk3

	SUBJID	AETERM
1	001	Vomit

データセットに存在しない, USUBJIDで絞り込みを行うという
whereステートメントにおいて, びっくりなことが起きている

AEドメインの中にSUBJIDがあるかどうかでYNのフラグを返す処理を考えてみる

素直に平文で書いたら

```
data samp3;  
set dm;  
if _N_=1 then do;  
  declare hash  
h1 (dataset:"raw_ae",multidata:"Y");  
  h1.definekey("SUBJID");  
  h1.definedone();  
end;  
AEFL = ifc( h1.check() = 0 , "Y", "N");  
run;
```

先に紹介したマクロを使うなら

```
data samp4;  
set dm;  
%chk(master=raw_ae,key=SUBJID,flg=AEFL)  
run;
```

VIEWTABLE: Work.Samp3					
	USUBJID	SUBJID	RFSTDTC	RFENDTC	AEFL
1	A-001	001	2024-09-01	2024-10-01	Y
2	A-002	002	2024-09-02	2024-10-02	N
3	A-003	003	2024-09-03	2024-10-03	Y

```
proc fcmp outlib=work.functions.common;  
/*SUBJIDをいれるとAEドメインにあるかどうかでYNが返る関数*/  
function ae_chk(SUBJID $) $;  
  declare hash h1(dataset: "raw_ae");  
  rc=h1.definekey("SUBJID");  
  rc=h1.definedone();  
  if h1.check() eq 0 then return("Y");  
  else return("N");  
endsub;  
quit;
```

これも当然, SQLでもwhere句でも動く

```
data wk5;  
set dm;  
AEFL= ae_chk(SUBJID) ;  
run;
```

```
data wk6;  
set dm;  
where ae_chk(SUBJID) ="Y";  
run;
```

通常、データステップのwhereステートメントでは
SQLにおける in サブクエリや
Exists句をつかったような処理が書けないというのが
文法だったが

FCMPのハッシュオブジェクトを使うことで可能となること
が
わかった

```
proc sql;  
  create table wk8 as  
    select *  
    from dm  
    where exists  
      (select 1 from raw_ae where  
        dm.SUBJID=raw_ae.SUBJID);  
quit;
```

```
proc sql;  
  create table wk7 as  
    select *  
    from dm  
    where SUBJID in  
      (select SUBJID from raw_ae);  
quit;
```

どんな技術にも必ず弱点がある

ハッシュオブジェクトの関数化も

関数定義時にハッシュオブジェクトに格納のデータセットが固定



引数にできない

```
proc fcmp outlib=work.functions.common;  
/*SUBJIDをいれるとDMドメインからUSUBJIDを返す関数*/  
function usubjid(SUBJID $) $;  
  declare hash h1(dataset: "DM");  
  rc=h1.definekey("SUBJID");  
  rc=h1.definedata("USUBJID");  
  rc=h1.definedone();  
  if h1.find() eq 0 then return(USUBJID);  
  else return("");  
endsub;  
quit;
```

関数の場合は一つの戻り値しかとれない。

コールルーチンにするとSQLの中で動かない。

サブルーチンもできる

```

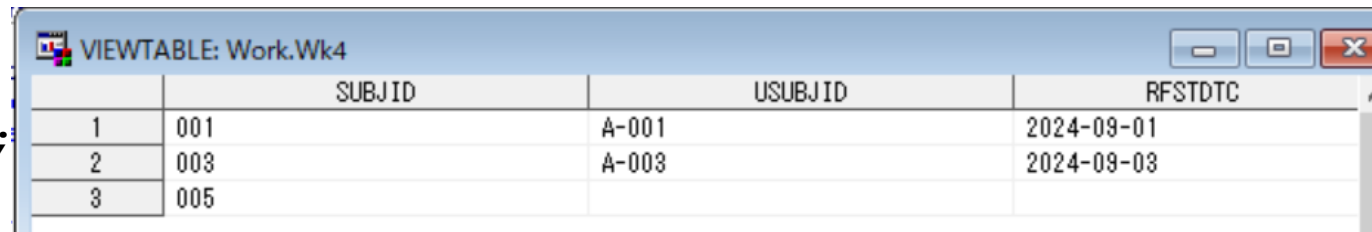
proc fcmp outlib=work.functions.common;
/*USUBJIDをキーにRFSTDTCとRFENDTCを返すサブルーチン*/
subroutine rfdtc(USUBJID $ , RFSTDTC $, RFENDTC $) ;
    outargs RFSTDTC, RFENDTC;
    declare hash h1(dataset: "DM");
    rc=h1.definekey("USUBJID");
    rc=h1.definedata("RFSTDTC");
    rc=h1.definedata("RFENDTC");
    rc=h1.definedone();
    if h1.find() ne 0 then call missing(of RFSTDTC
RFENDTC);
endsub ;
quit;

```

```

data wk4;
length SUBJID USUBJID RFSTDTC RFENDTC $200.;
set raw_ae;
USUBJID=usubjid(SUBJID);
call missing(of RFSTDTC RFENDTC);
call rfdtc(USUBJID, RFSTDTC, RFENDTC);
run;

```



	SUBJID	USUBJID	RFSTDTC
1	001	A-001	2024-09-01
2	003	A-003	2024-09-03
3	005		

FCMPの中のハッシュオブジェクトで使えるメソッドは以下の通り
DO_OVERメソッドやSUMメソッドなどは, 使えない. 通常のハッシュオブジェクトに比べて制限がかかることをに注意

ADD メソッド

CHECK メソッド

CLEAR メソッド

DEFINEDATA メソッド

DEFINEDONE メソッド

DEFINEKEY メソッド

DELETE メソッド: ハッシュオブジェクトとハッシュ反復子オブジェクト

EQUALS メソッド

FIND メソッド

DECLARE ステートメント: ハッシュオブジェクトとハッシュ反復子オブジェクト

NUM_ITEMSメソッド 属性

REMOVE メソッド

REPLACE メソッド

FIRST メソッド

LAST メソッド

NEXT メソッド

PREV メソッド

SETCUR メソッド

Dictionaryオブジェクト

FCMPプロシジャのDICTIONARYオブジェクト(9.4M5～)

基本的にkeyとdataで管理するという点ではHASHオブジェクトに非常に似ているが、大きな違いとして、HASHオブジェクトのようにdefinekeyやdefinedataで構造を事前定義する必要がない

declare dictionary 任意の名前XX;

XX[key1,key2,.....]=値 (格納・更新)

変数=XX[key1,key2,.....] (値の取得)

keyやdataが型に縛られず、文字や数値を混在して格納したり、配列などもdataに格納できる。

次スライドで表現する辞書のイメージ

key1	key2	key3	key . . .	data
1				100
2				200
A				AAA
B				BBB
1	1			1000
A	B	1		CCC

```

proc fcmp;
declare dictionary dc;
length data2 data3 data5 $200;
dc[1] = 100;
dc[2] = 200;
dc["A"]="AAA";
dc["B"]="BBB";
dc[1,1] = 1000;
dc["A","B",1] = "CCC";
key1 = 1;
data1 = dc[key1];
put key1= data1= ;
key2 = "A";
data2 = dc[key2];
put key2= data2=;
dc[1] = "ZZZ";
data3 = dc[key1];
put key1= data3=;

```

```

key4_1 = 1;
key4_2 = 1;
data4 = dc[key4_1,key4_2];
put key4_1= key4_2= data4=;
key5_1 = "A";
key5_2 = "B";
key5_3 = 1;
data5 = dc[key5_1,key5_2,key5_3];
put key5_1= key5_2= key5_3= data5=;
run;
quit;

```

key1	key2	key3	key . . .	data
1				100
2				200
A				AAA
B				BBB
1	1			1000
A	B	1		CCC

```

key1=1 data1=100
key2=A data2=AAA
key1=1 data3=ZZZ
key4_1=1 key4_2=1 data4=1000
key5_1=A key5_2=B key5_3=1 data5=CCC

```

FCMPの中のDcitionaryオブジェクトで使えるメソッドは以下の通り
ハッシュオブジェクトに似ている。ただ、ハッシュオブジェクトと違って
単純な出し入れだけの場合は メソッドを必要とせず 辞書名[key] =値で操作ができ、それが一番多く使う

あと、上級者向けになりすぎるので、言及しませんが、実は辞書には、ハッシュオブジェクトや配列など
通常の単一値以外のものを格納することができる... なんと..

CLEAR メソッド

CLONE メソッド

DECLARE ステートメント: ディクショナリオブジェクト

DESCRIBE メソッド

FIRST メソッド

HASNEXT メソッド

HASPREV メソッド

LAST メソッド

NEXT メソッド

NUM_ITEMS メソッド

PREV メソッド

REF メソッド

REMOVE メソッド

SKIPNEXT メソッド

SKIPPREV メソッド

テストデータ

```
data a;  
SEX="M";COUNTRY="JPN";output;  
SEX="F";COUNTRY="CHN";output;  
SEX="M";COUNTRY="JPN";output;  
run;
```

VIEWTABLE: Work.A		
	SEX	COUNTRY
1	M	JPN
2	F	CHN
3	M	JPN

out1	out2
Male	Japan
Female	China
Male	Japan

out1	out2
男性	日本
女性	中国
男性	日本

値から、それぞれ 英語用・日本語用の表示が欲しい場合など、SASフォーマットではなく 今回の技を使う

```
proc fcmp outlib=work.functions.common;
function f1(key1 $,key2 $,key3 $) $ ;
length key1 key2 rvalue dummy $200 ;
_key1=strip(key1);
_key2=strip(key2);
_key3=strip(key3);
declare dictionary d1 ;
d1['E','SEX','M']='Male';
d1['J','SEX','M']='男性';
d1['E','SEX','F']='Female';
d1['J','SEX','F']='女性';
d1['E','COUNTRY','JPN']='Japan';
d1['J','COUNTRY','JPN']='日本';
d1['E','COUNTRY','CHN']='China';
d1['J','COUNTRY','CHN']='中国';
rvalue=d1[_key1,_key2,_key3];
return(rvalue);
endsub;
run;
quit;
```

①英語or日本語

②カテゴリ名

③カテゴリ値

の 3つのキーで管理される辞書オブジェクトをつくる

```
options cmplib=work.functions ;
```

```
data output1;  
length out1 out2 $200.;  
set a;  
out1=f1("E","SEX",SEX);  
out2=f1("E","COUNTRY",COUNTRY);  
run;
```

	out1	out2	SEX	COUNTRY
1	Male	Japan	M	JPN
2	Female	China	F	CHN
3	Male	Japan	M	JPN

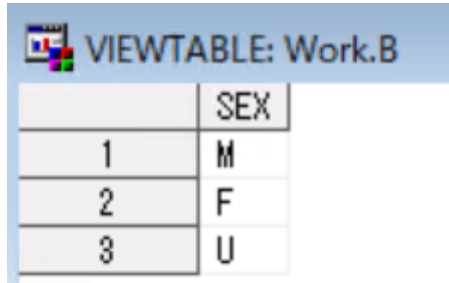
```
proc sql;  
select  
    f1("E","SEX",SEX) as out1,  
    f1("E","COUNTRY",COUNTRY) as out2  
from output1;  
quit;
```

out1	out2
Male	Japan
Female	China
Male	Japan

```
proc sql;  
select  
    f1("J","SEX",SEX) as out1,  
    f1("J","COUNTRY",COUNTRY) as out2  
from output1;  
quit;
```

out1	out2
男性	日本
女性	中国
男性	日本

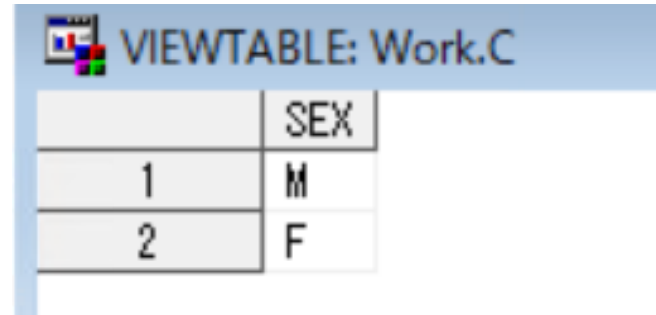
```
data b;  
SEX="M";output;  
SEX="F";output;  
SEX="U";output;  
run;
```



VIEWTABLE: Work.B

	SEX
1	M
2	F
3	U

```
data c;  
set b;  
where f1 ("E", "SEX", SEX) ne "";  
run;
```



VIEWTABLE: Work.C

	SEX
1	M
2	F

これもまたwhereステートメントで作動するため、辞書にあるもの(ないもの)のみを抽出するといったことが可能になる

余談

マニアックすぎて、過去の私の総会論文発表の中で、一番ウケが悪かったものとして
SQLの中で、LAG関数が使えない制限を、FCMPの ハッシュオブジェクトやDictionaryで無理やり突破するというの
があるので、気になる人はぜひ...



FCMPのSTATIC statement, HASH object, DICTIONARY objectそれぞれによるLAG関数機能の定義

○森岡 裕

(イーピーエス株式会社 統計解析1部)

FCMP-HASH objectによるLAG関数定義

```
proc fcmp outlib=work.functions.common;
```

```
function h_lagn(var,n);
```

```
declare hash h1;
```

```
rc=h1.definekey("mono");
```

```
rc=h1.definedata("val");
```

```
rc=h1.definedone();
```

```
_mono=monotonic();
```

```
mono=_mono-n;
```

```
rc=h1.find();
```

```
rv=val;
```

```
val=var;
```

```
mono=_mono;
```

```
rc=h1.add();
```

```
return(rv);
```

```
endsub;
```

```
run; options cmplib = work.functions;
```

```
quit; data OUT3;
```

```
set sashelp.class;
```

```
L_age=d_lagn(age,1);
```

```
run;
```

Monotonic関数の生成する処理連番を
キーとしてハッシュに値を格納する。

呼び出された行の処理連番から、関数の
第二引数で指定した数字(n)を引いて、そ
れをキーとしてfindすることで、n個前の値
を取得できる

Name	Age	L_age
アルフレッド	14	.
アリス	13	14
バーバラ	13	13
キャロル	14	13
ヘンリー	14	14
ジェームズ	12	14
ジェーン	12	12
ジャネット	15	12
ジェフリー	13	15
ジョン	12	13
ジョイス	11	12
ジュディー	14	11
ルイーズ	12	14
メアリー	15	12
フィリップ	16	15
ロバート	12	16
ロナルド	15	12
トーマス	11	15
ウィリアム	15	11

ちなみにSTATICでの欠点はこちらの方法でも同様

FCMP-HASH objectによるLAG関数 (in proc SQL)

```
proc sql;
select name, age, h_lagn(age, 2) as _age2
from sashelp.class;
quit;
```

できた

名前	年齢	_age2
アルフレッド	14	.
アリス	13	.
バーバラ	13	14
キャロル	14	13
ヘンリー	14	13
ジェームズ	12	14
ジェーン	12	14
ジャネット	15	12
ジェフリー	13	12
ジョン	12	15
ジョイス	11	13
ジュディー	14	12
ルイーズ	12	11
メアリー	15	14
フィリップ	16	12
ロバート	12	15
ロナルド	15	16
トーマス	11	12
ウィリアム	15	15

FCMP-DICTIONARY objectによるLAG関数定義

```
proc fcmp outlib=work.functions.common;
  function d_lagn(var,n);
    declare dictionary d1;
    rv=d1[n+1];
    d1[1]=var;
    do i=100 to 2 by -1;
      d1[i]=d1[i-1];
    end;
    d1[1]=.;
    return(rv);
  endsub;
quit;
options cmplib = work.functions;
data OUT3;
  set sashelp.class;
  L_age=d_lagn(age,1);
run;
```

保持数を適当に100として、
呼び出しごとに辞書内のデータを
スライドさせていっているイメージ

Name	Age	L_age
アルフレッド	14	.
アリス	13	14
バーバラ	13	13
キャロル	14	13
ヘンリー	14	14
ジェームズ	12	14
ジェーン	12	12
ジャネット	15	12
ジェフリー	13	15
ジョン	12	13
ジョイス	11	12
ジュディー	14	11
ルイーズ	12	14
メアリー	15	12
フィリップ	16	15
ロバート	12	16
ロナルド	15	12
トーマス	11	15
ウィリアム	15	11

ちなみにSTATICでの欠点はこちらの方法でも同様

FCMP-DICTIONARY objectによるLAG関数定義 (in proc SQL)

```
proc sql;
select name, age, d_lagn(age, 2) as _age2
from sashelp.class;
quit;
```

できた

名前	年齢	_age2
アルフレッド	14	.
アリス	13	.
バーバラ	13	14
キャロル	14	13
ヘンリー	14	13
ジェームズ	12	14
ジェーン	12	14
ジャネット	15	12
ジェフリー	13	12
ジョン	12	15
ジョイス	11	13
ジュディー	14	12
ルイーズ	12	11
メアリー	15	14
フィリップ	16	12
ロバート	12	15
ロナルド	15	16
トーマス	11	12
ウィリアム	15	15

まー そんなこんなで、自分はハッシュオブジェクトを通常のマクロベースで使っていますが、毎度かき捨てで関数化したり
ADSLやADTTEのような構造固定のADaMの処理を関数にしたり、
色々有用かも。あと自分、SASフォーマット嫌いなので
それならむしろ
全部Dictionaryオブジェクトにしたい

ご清聴ありがとうございましたー