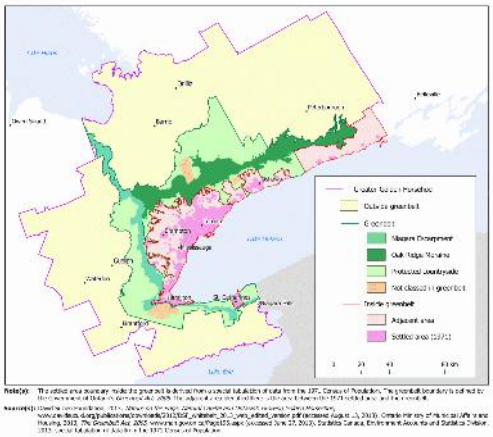
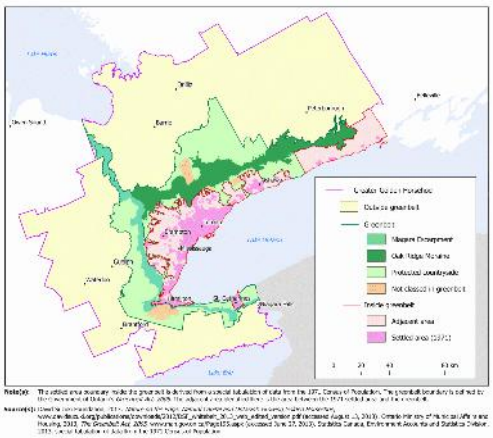
Golden Horseshoe SAS Users Group
October 21, 2016



I Object:
SAS® Does Objects with DS2

Peter Eberhardt
Fernwood Consulting Group Inc.
Xue Yao
Winnipeg Regional Health Authority

Golden Horseshoe SAS Users Group
October 21, 2016



I Object:
SAS® Does Objects with DS2

Peter Eberhardt
Fernwood Consulting Group Inc.
Xue Yao
Winnipeg Regional Health Authority

Copyright 2016 Peter Eberhardt, Fernwood Consulting Group Inc. All rights Reserved.

Agenda

- Overview of DS2
- Data types
- Scope
- Methods
- Packages
- Missing values and NULL values

Overview of DS2

- DS2 is new in SAS 9.4
- DS2 is based on object-oriented concepts
- DS2 has many distinguished benefits
  - » Data types
  - » User defined methods
  - » Packages
  - » Threads
  - » FedSQL

» **BETTER PROGRAMS**

Overview of DS2

```
PROC DS2 SCOND=ERROR ;

PACKAGE pkglib.pkgName / overwrite = yes;
DECLARE double x;
method one(<dataType> var1, <dataType> var1, ... ) returns <datatype>;
       DECLARE integer x;
       /* statements */
       return(<variable>);
end;
/* more methods */
ENDPACKAGE;

data <dsname> (overwrite=YES);
```

Overview of DS2

```
data <dsname> (overwrite=YES);
DECLARE integer x;
method calc1(<dataType> var1, <dataType> var1, ... ) returns <datatype>;
    DECLARE double x;
    /* more statements */
    return(<variable>);
end;
/* more methods */
method init();
    DECLARE double x;
    /* more statements */
end;
```

Overview of DS2

```
method init();
    DECLARE <dataType> var1;
    /* more statements */
end;
method run();
 DECLARE <dataType> var1;
 DECLARE <dataType> var2;
/* more statements */
end;
method term();
    /* more statements */
end;
run;
quit;
```

Agenda

- ✓ Overview of DS2
- Data types
- Scope
- Methods
- Packages
- Missing values and NULL values

Data Types

- DATA Step
  - Two data types
    - » Numeric
      - » Signed double precision
    - » Character
      - » Fixed length

Data Types

- DS2
  - Whack
    - » 4?
    - » 7?
    - » 9?
    - » 11?
    - » 13?
    - » 16?
    - » 23?

16

Data Types

- DS2
  - Numeric
    - » Seven types
      - » bigint, int, smallint, tinyint
      - » decimal(precision, scale)
      - » Double/float, real
  - Character
    - » Four types
      - » char(n), varchar(n)
      - » nchar(n), nvarchar(n)

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Data Types

- DS2
  - Date and Time
    - » Three types
      - » date, time, timestamp
  - Binary
    - » Two types
      - » binary(n), varbinary(n)

Data Types

- DECLARE
  - PROC DS2 *SCOND=ERROR* ;
  - declare <datatype> varName {having <attributes>};
    - » declare tinyint i;
      - » do i = 1 to 12;
    - » declare char(20) firstName;
      - » ~~firstname = "peter";~~
      - » firstname = 'peter';

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Data Types

- DECLARE
  - PROC DS2 *SCOND=ERROR* ;
  - declare <datatype> varName having <attributes>;
    - » label
    - » format
    - » informat
  - declare char(20) firstName having label 'First Name' format $20.;
  - declare int age i j;

Data Types

- DECLARE
  - declare date dob having format yymmdd10. label ‘Date of Birth’;
    - » Dob = date ’2014-01-01’;

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Data Types

- DECLARE
  - declare date “Date of Birth” having format yymmdd10.;
    - » “Date of Birth” = date ’2014-01-01’;

Data Types

- DECLARE
  - Temporary array
    - » Not part of the program data vector (PDV)
  - declare integer age[5] ;
    - » age[2] = 25;

Data Types

- DECLARE
  - Variable array
    - » Part of the program data vector (PDV)
  - **vararray** integer age[5] ;
    - » age[2] = 25;
    - » age2 = 25;
  - **vararray** integer age[*] age1-age5;
    - » age[2] = 25;
    - » age2 = 25;

Data Types

- DATA Step
  - Two data types
    - » Numeric
    - » Character

Agenda

- ✓ Overview of DS2
- ✓ Data types
- Scope
- Methods
- Packages
- Missing values and NULL values

Scope

- Scope is determined by where the identifier is declared
  - Local – Within a method
  - Global – Outside the methods
- Scope determines where an identifier is visible
  - Local – accessible only from within the method
  - Global – accessible from anywhere in the program

Scope

- Scope can be considered an attribute of identifiers
- Only global variables are added to the PDV
- Within a scope identifiers must be unique. Same identifier name can be used in different scopes

```
data <dsname>;
DECLARE integer x;
method calc1((<dataType> var1, <dataType> var1, ... ) returns <datatype>;
    DECLARE double x;
    /* more statements */
    return(<variable>);
end;
```

Scope

- When the global and local variables have same name the local variable will take precedence over the global variable

```
data <dsname>;
DECLARE integer x;
method calc1((<dataType> var1, <dataType> var1, ... ) returns <datatype>;
    DECLARE double x;
    x = 1.0; /* assigns to LOCAL x */
    /* more statements */
    return(<variable>);
end;
```

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Scope

- When the global and local variables have same name the local variable will take precedence over the global variable
  - » this.varName will use the GLOBAL value

```
data <dsname>;
DECLARE integer x;
method calc1((<dataType> var1, <dataType> var1, ... ) returns <datatype>;
    DECLARE double x;
    this.x = 1; /* assigns to GLOBAL x */
    /* more statements */
    return(<variable>);
end;
```

Scope

- **Tip: for variables that don't need to be output to the dataset use the local scope as much as possibleat**

Agenda

- ✓ Overview of DS2
- ✓ Data types
- ✓ Scope
- Methods
- Packages
- Missing values and NULL values

Methods

- Methods are the fundamental building blocks of DS2
- In DS2, all executable code must reside in methods

```
method Name({argument list}) {returns <datatype>};
... DS2 statements ...
end;
```

- Types of method
  - Predefined: INIT, RUN, TERM, SETPARMS
  - User-defined

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Methods

- INIT method is automatically called once at the beginning of the program
- After the INIT method runs, RUN is called and will iterate once for each input row
- TERM method is automatically called once at the end of the program
- SETPARMS method is used in a thread to initialize parameters

Methods

```
DATA example;
  IF _N_ = 1
  then
    do;
      /* some statements to do at start*/
    end;
  set bigdata.lotsOfData end=done;
  /* some statements */
  IF done
  then
    do;
      /* some statements to do at end */
    end;
run;
```

INIT()

RUN()

TERM()

Methods

- When INIT, RUN, TERM are explicitly defined, they must be defined without any parameters and without a return value
- User-defined method can take parameters/arguments and return values
  - Each parameter must have a data type
  - The scope of the parameter is local to the method
- Method needs to be called by method name and corresponding the ordered list of the method's parameter types

Methods

```
method C_to_F(INTEGER C) returns DOUBLE;
  /* convert degrees celsius to degrees fahrenheit */
  return 32 + (C *  (9 / 5));
end;
```

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Methods

```
246  proc DS2 scond=ERROR;
247  data winterSucks (overwrite=YES);
248  declare double degF having label 'Degrees Farenheit' format 6.;
249  method C_to_F(INTEGER C) returns DOUBLE;
250    /* convert degrees celsius to degrees fahrenheit */
251    return 32 + (C *  (9 / 5));
252  end;
253  method run();
254    declare int degC;
255    degC = 0;      degF = C_to_F(degC); put degC= degF=;
256  end;
257  enddata;
258  run;
degC=0 degF=    32
NOTE: Execution succeeded. One row affected.
```

Methods

```
260  proc DS2 scond=ERROR;
261  data winterSucks (overwrite=YES);
262  declare double degF having label 'Degrees Farenheit' format 6.;
263  method C_to_F(INTEGER C) returns DOUBLE;
264    /* convert degrees fahrenheit to degrees celsius */
265    return 32 + (C *  (9 / 5));
266  end;
267  method run();
268    declare int degC;
269    degC = 100; degF = C_to_F(degC); put degC= degF=;
270  end;
271  enddata;
272  run;
degC=100 degF=   132
NOTE: Execution succeeded. One row affected.
```

Methods

```
method C_to_F(INTEGER C) returns DOUBLE;
  /* convert degrees celsius to degrees fahrenheit */
 return 32 + (C *  (9  / 5 ));
 return 32 + (C *  (9. / 5.));
end;
```

Methods

```
274  proc DS2 scond=ERROR;
275  data winterSucks (overwrite=YES);
276  declare double degF having label 'Degrees Farenheit' format 6.;
277  method C_to_F(INTEGER C) returns DOUBLE;
278    /* convert degrees fahrenheit to degrees celsius */
279    return 32 + (C * (9. / 5.));
280  end;
281  method run();
282    declare int degC;
283    degC = 100; degF = C_to_F(degC); put degC= degF=;
284  end;
285  enddata;
286  run;
degC=100 degF=   212
```

Methods

- Method Overload

```
method concat(char(100) x, char(100) y) returns char(200);
        return "pre" || trim(x) || y;
end;

method concat(char(100) x, char(100) y, char(100) z) returns char(200);
        return trim(x) || trim(y) || z;
end;
```

Methods

- Methods are the fundamental building blocks of DS2
- In DS2, all executable code must reside in methods
- Types of method
  - Predefined: INIT, RUN, TERM, SETPARMS
  - User-defined
    - » As many as you need
- User-defined method overloaded
  - Same name but different signatures

Agenda

- ✓ Overview of DS2
- ✓ Data types
- ✓ Scope
- ✓ Methods
- Packages
- Missing values and NULL values

Packages

- Package is a collection of logically related methods
- Packages be re-used in DS2 programs and threads
- Pre-defined packages
- User defined packages

- DECLARE package instances

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Packages

- Package is a collection of logically related methods and variables and can be shared and re-used in DS2 programs and threads

```
PACKAGE package [/table-options];
        …DS2 statements…
ENDPACKAGE;
```

- DECLARE PACKAGE statement constructs an instance
- Method stored in a package can be accessed by creating an instance of the package and then using dot notation to call the method

Packages

- Predefined packages
  - FCMP supports calls to FCMP functions and subroutines from within the DS2 language
  - The Hash package enables you to quickly and efficiently store, search, and retrieve data based on unique lookup keys.
  - Logger provides a basic interface to the SAS logging facility
  - Matrix provides a powerful and flexible matrix programming capability
  - SQLSTMT provides a way to pass FedSQL statements to a DBMS for execution and to access the result set returned by the DBMS

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Packages

```
libname conv "C:\DS2\convert";
proc ds2 SCOND=ERROR ;
package conv.convert / overwrite = yes;
 method C_to_F(INTEGER C) returns DOUBLE;
  /* convert degrees fahrenheit to degrees celsius */
  return 32 + (C *  (9. / 5.));
 end;
 method IN_to_CM(DOUBLE inch) returns double;
  return inch * 2.54;
 end;
  method F_to_C(IN_OUT INTEGER F, IN_OUT DOUBLE C);
  /* convert degrees fahrenheit to degrees celsius */
  C = (F - 32) * (5. / 9.);
 end;
endpackage;
```

Packages

```
libname conv "C:\DS2\convert";
proc ds2 SCOND=ERROR;
package conv.convert / overwrite = yes;
  /* methods */
endpackage;
```

Packages

```
method C_to_F(INTEGER C) returns DOUBLE;
  /* convert degrees fahrenheit to degrees celsius */
  return 32 + (C *  (9. / 5.));
 end;

 method IN_to_CM(DOUBLE inch) returns double;
  return inch * 2.54;
 end;

 method F_to_C(IN_OUT INTEGER F, IN_OUT DOUBLE C);
  /* convert degrees fahrenheit to degrees celsius */
  C = (F - 32) * (5. / 9.);
 end;
```

Packages

```
data one (overwrite=yes);
DECLARE int C I;
DECLARE double F    having format 6.1;
DECLARE double cm   having format 8.3;
DECLARE double inch having format 8.3;
DECLARE char(1) type;
declare package conv.convert cnv();
```

Packages

```
  method run();
    type = 'C';
    do c = 0 to 100;
      f = cnv.C_to_F(c);
      output;
     end;
    type = 'I';
    do inch = 0. to 10. by 0.25;
      cm = cnv.IN_to_CM(inch);
      output;
    end;
end;
enddata;
run;
quit;
```

Packages

```
/*****************************************************************/
/* use a data step to create the average                         */
DATA avgAmt;
     retain total 0;
     drop amount;
     set amounts end = done;
    total = total + amount;
    if done
    then
       do;
         avgAmt = total / _n_;
         output;
       end;
run;
```

Packages

```
/*****************************************************************/
/* create a package                                              */
/*   package has more functionality than needed                  */
proc ds2;
package packlib.accumulate / overwrite = yes;
declare integer cnt;
declare numeric(10,2) amt;

/* default constructor                                           */
method accumulate();
     cnt = 0;
     amt = 0.00;
end;
```

Packages

```
/* constructor with initial amount                    */
method accumulate(numeric(10,2) inAmt);
    accumulate();
    if inAmt > 0.00
    then
      do;
        cnt = 1;
        amt = inAmt;
      end;
end;
```

Packages

```
/* initial add method                                 */
method add(numeric(10,2) inAmt);
    cnt = cnt + 1;
    amt = amt + inAmt;
end;
```

Packages

```
/* get methods to get count or amount                    */
method getCnt() returns integer;
    return (cnt);
end;

method getAmt() returns numeric(10,2);
    return (amt);
end;

/* get method for average                                 */
method getAvg() returns numeric(10,2);
    return (amt/cnt);
end;
```

Packages

```
/* our DS2 program using the package                     */
proc ds2;
data acc (overwrite=yes);
declare package packlib.accumulate fees();
declare numeric(10,2) total avgAmt having format comma15.2;
drop amount;
method run();
    set amounts;  fees.add(amount);
end;
method term();
  total  = fees.getamt();   avgAmt = fees.getAvg();   output;
end;
enddata;
run; quit;
```

Packages

```
DATA summaryAmts;
     retain total max min 0;
     drop amount;
     set amounts end = done;
     if _n_ = 1 then
     do;
        min = amount; max = amount;
     end;
```

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Packages

```
     total = total + amount;
     if amount < min then min = amount;
     else if amount > max then max = amount;
     if done then
     do;
         avgAmt = total / _n_;  output;
     end;
run;
```

Packages

```
/* default constructor   == UPDATED                              */
method accumulate();
    cnt = 0;
    amt = 0.00;
    min = 99999999.99;
    max = -99999999.99;
End;
/* add method       == UPDATED                                  */
method add(numeric(10,2) inAmt);
    cnt = cnt + 1; amt = amt + inAmt;
    if inAmt > max   then max = inAmt;
    else if inAmt < min then min = inAmt;
end;
```

Packages

```
/* our DS2 program using the package                             */
/*  note we only had to make two changes:                        */
/*   1. add min and max to DECLARE statement                     */
/*   2. add the get methods in term()                            */
proc ds2;
data acc2 (overwrite=yes);
declare package packlib.accumulate fees();
declare numeric(10,2) total avgAmt min max having format comma15.2;
drop amount;
method run();
     set amounts;
     fees.add(amount);
end;
```

Packages

```
method term();
  total = fees.getamt();
  avgAmt = fees.getAvg();
  min = fees.getMin();
  max = fees.getMAX();
  output;
end;
enddata;
run;
quit;
```

I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Packages

- Package can be invoked as a function in a FedSQL SELECT statement
- Package instance can be returned from a method.
  - The package instance needs to be created in a scope outside the scope of the method
  - Using _NEW_ operator and THIS scope keyword, the instance can be created as global scope in a program, package or thread
- Package instance also can be passed to methods

Packages

- Package is a collection of logically related methods
- Packages be re-used in DS2 programs and threads
- Pre-defined packages
- User defined packages

- DECLARE package instances

Agenda

- ✓ Overview of DS2
- ✓ Data types
- ✓ Scope
- ✓ Methods
- ✓ Packages
- Missing values and NULL values

Missing and NULL Values

- SAS Missing Values
  - Normal: ., ‘ ‘ (space)
  - Special: ._ .Z-.A
- ANSI SQL NULL Values
- SAS Mode
- ANSI Mode

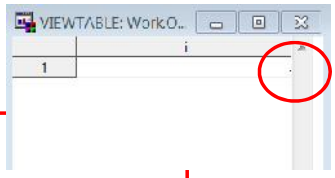
I Object: SAS® Does Objects with DS2
Peter Eberhardt and Xue Yao

Missing and NULL Values

- SAS Missing Values
  - Only datatypes DOUBLE and CHAR(n)

Missing and NULL Values

```
169  proc ds2 scond=error;
170  data one (overwrite=yes);
171  declare int i ;
172  method init();
173    i = .; put i=;
174    if i = .        then put 'i is missing ' i=;
175      OUTPUT;
176  end;
177  enddata;
178  run;
i=
```



Missing and NULL Values

```
234  proc ds2 scond=error;
235  data one (overwrite=yes);
236  declare int i ;
237  method init();
238    i = .; put i=;
239    if i = .        then put 'i is missing ' i=;
240    if MISSING(i)   then put 'i is missing ' i=;
241      OUTPUT;
242  end;
243  enddata;
244  run;
i=
i is missing  i=
```

Missing and NULL Values

```
247  data one (overwrite=yes);
248  declare int i ;
249  method init();
250    i = .; put i=;
251    if i = .         then put 'i is missing ' i=;
252    if MISSING(i)    then put 'i is missing ' i=;
253    if NULL(i)       then put 'i is NULL    ' i=;
254    OUTPUT;
255  end;
256  enddata;
257  run;
```

```
i=
i is missing  i=
i is NULL     i=
```

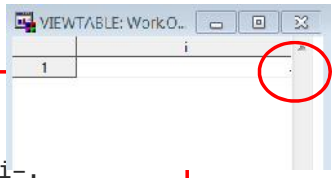
Missing and NULL Values

```
286  data one (overwrite=yes);
287  declare int i ;
288  method init();
289    i = NULL; put i=;
290    if i = .         then put 'i is missing ' i=;
291    if MISSING(i)    then put 'i is missing ' i=;
292    if NULL(i)       then put 'i is NULL    ' i=;
293    OUTPUT;
294  end;
295  enddata;
296  run;
```

```
i=
i is missing  i=
i is NULL     i=
```

Missing and NULL Values

```
328  data one (overwrite=yes);
329  declare int i ;
330  method init();
331    i = NULL; put i=;
332    if i = .A       then put 'i is missing ' i=;
333    if MISSING(i)   then put 'i is missing ' i=;
334    if NULL(i)      then put 'i is NULL    ' i=;
335    OUTPUT;
336  end;
337  enddata;
338  run;
```




```
i=
i is missing  i=
i is NULL     i=
```

Missing and NULL Values

- SAS Missing Values
  - Normal: ., ‘ ‘ (space)
  - Special: ._ .Z-.A
- ANSI SQL NULL Values
- SAS Mode
- ANSI Mode

Agenda

- ✓ Overview of DS2
- ✓ Data types
- ✓ Scope
- ✓ Methods
- ✓ Packages
- ✓ Missing values and NULL values

When to use DS2

- Precision
- Reusable methods
  - PROC FCMP
  - User-defined packages and methods
- SQL
- Offload processing
- Threading
  - Symmetric Multiprocessing & Massively Parallel Processing environment

Questions??

Peter Eberhardt

peter@fernwood.ca

Twitter: @rkinRobin

WeChat: peterOnDroid

Xue Yao

xueyao.statistic@gmail.com