Arrays – Data Step Efficiency

Harry Droogendyk – Stratia Consulting Inc.

Introduction

- row-wise processing handled efficiently by PROCs
  - e.g. BY processing
- column-wise.... not as easy
  - e.g. row of data with 12 monthly columns

- SAS arrays are different than most other languages
  - _temporary_ – akin to other languages
  - PDV – Program Data Vector

  - Physical
  - Logical

Definitions

- temporary:

```
array factors(12,3)    _temporary_ ;
```

- variable lists:

```
array names    $    _character_;
array qtr_data      qtr1-qtr4; * create?;
array months        month: ;
array all_codes     co_cd -- dept_cd;
```

Initialization

```
array mths1 (12) _temporary_ (12*0);

array mths2 (12) _temporary_
                 (0 0 0 0 0 0 0 0 0 0 0 0);

array mth_qtr (12)_temporary_
                 (3*1 3*2 3*3 3*4);

call missing(of mths(*));
```

Array Indexing

```
array ages (5:18) _temporary_;
d = dim(ages);
l = lbound(ages);
h = hbound(ages);

put 'Ages array has ' d ' elements,
     bounded by ' l ' and ' h;

Log: Ages array has 14 elements,
    bounded by 5 and 18
```

Array Indexing

```
set sashelp.class;
ages(age) + 1; * temp array so auto retained;

do i = lbound(ages)
  to hbound(ages);
    put i 2. @7 ages(i);
end;
```

Log results:

```
<snip>
10   .
11   2
12   5
13   3
14   4
15   4
16   1
17   .
```

Functions

```
a = ‘SUGs';          b = ‘are';
c = 'the';           d = 'best';

array chars _character_;

e = catx('.',of chars(*));
put e;
```

```
Log: SUGs.are.the.best
```

Functions

```
set lib.ds;
array sales dvds cds books cards;

do _i = 1 to dim(sales);
     total_sales + sales(_i);
end;

total_sales = sum(dvds,  cds,
                  books, cards);

total_sales = sum(of sales(*));
```

Sorting

```
array name(8) $10
      ('George' 'nancy' 'Susan'
       'george' 'James' 'Robert'
       'Rob' 'Fred');
call sortc(of name(*));
put +3 name(*);
```

Log:

```
 Fred George James Rob Robert Susan
 george nancy
```

Efficiencies

- wall paper code

```
data monthly_sales;
  set sales;
  if month1=. then month1=0; else month1=round(month1,.1);
  if month2=. then month2=0; else month2=round(month2,.1);
  ....
  if month11=. then month11=0; else month11=round(month11,.1);
  if month12=. then month12=0; else month12=round(month12,.1);
run;
```

Efficiencies

- 12 lines of code reduced to 3

```
data monthly_sales;
   set sales;
   array m month1 - month12;
   do _i = 1 to dim(m);
      m(_i)= ifn(missing(m(_i)),
                 0,
                 round(m(_i),.1));
   end;
   drop _: ;
run;
```

Transposing

- annual table of revenue & expenses
  - region_cd, year
    - rev_mth1 – rev_mth12, exp_mth1 – exp_mth12
- difficult to use for analysis
  - need separate rows for each month
  - one revenue / one expense column per row
- PROC TRANSPOSE
  - but...
    - we need to "build" the date
    - calculated metrics

Transposing

```
data sales;
  array rev_mth(12);
  array exp_mth(12);
  yr = 2015;

  do region_cd = 1,3,6,11,12;
      do _i = 1 to 12;
            rev_mth(_i) = ranuni(1) * 1000;
            exp_mth(_i) = ranuni(1) * 900;
      end;
      output;
  end;
  drop _: ;
run;
```

Transposing – normalized

```
data sales_vert;
   set sales;
   array rev_mth(12);
   array exp_mth(12);

   do _i = 1 to 12;
        dt     = intnx('month',input(
                     cats(put(yr, 4.),put(_i,z2.),'01'),
                                    yymmdd8.),0,'end');
        rev    = rev_mth(_i);
        exp    = exp_mth(_i);
        profit= rev - exp;
        output;
   end;

   drop _: rev_mth: exp_mth: yr;
run;
```

Transposing – for reporting

```
data sales_yr /
       view=sales_yr;
  set sales_vert
       ( drop = profit );
  yr = year(dt);
run;
```

```
data sales_xls;

  array rev_mth(12);
  array exp_mth(12);

  do until(last.yr); * DOW;
       set sales_yr;
        by region_cd yr ;

       _i           = month(dt);
       rev_mth(_i) = rev;
       exp_mth(_i) = exp;
  end;

  drop _: rev exp dt;
run;
```

Lookups

```
set sashelp.class;

array dosages (5:18) _temporary_
  ( .5 .5 .6 .7 1 1.2 1.3 1.5
     1.6 1.8 2.1 2.3 2.7 3.0);

dosage = dosages(age);
```

Lookups

```
array sale_products(3) $
    _temporary_
    ('TV','Fridge','Stove');

set catalog;

if product in sale_products;
```

Special Array Functions

- VNAME variable name
- VLABEL label of the variable
- VFORMAT format applied to variable
- VLENGTH defined variable length
- VINARRAY boolean result, variable in array

Special Array Functions

```
length    age 3;
label     age = 'Yrs';
format    age 2.
          salary dollar8.
          grade percent8.2;
array nums _numeric_;
do _i = 1 to dim(nums);
     n = vname(nums(_i));
     l = vlabel(nums(_i));
     f = vformat(nums(_i));
     t = vlength(nums(_i));
     put n @10 l @18 f @30 t;
end;
```

Special Array Functions

Name	Label	Format	Len
age	Yrs	F2.	3
salary	salary	DOLLAR8.	8
grade	grade	PERCENT8.2	8

Real World Example

- customer care at a Telco
  - customer calls support
  - feedback survey via text message
    - Fizzback – customer engagement
    - flags unfavorable responses
  - dissatisfied customers to be contacted
    - generate fixed width text dialer file
    - outbound loyalty calls

Real World Example

- define a skeleton dataset

```
data dialer_skeleton;
    format    num        $12. name$20.
              expiry     $10. msg  $15.
              rep        $3.  ;

    array txtfile _character_;
    call symputx('dim',dim(txtfile));
         * capture array length;
    stop;
run;
```

Real World Example

- define test data

```
data dialer_input;
  if 0 then set dialer_skeleton;* set variable
                                   lengths/formats;
  num        = '9052941234'; name = 'Tim Brokking';
    expiry   = '31Oct2016'd; msg  = '1st call';
    rep      = '01';                  output;
  num        = '5196471212'; name = 'Susan Jones';
    expiry   = '01Apr2016'd; msg  = 'Second call';
    rep      = '01';                  output;
  num        = '4162235678'; name = 'Sam Brown';
    expiry   = '02Apr2016'd; msg  = 'New customer';
    rep      = '2';                output;
  num        = '6137219988'; name = 'David Wharton';
    expiry   = '29Mar2016'd; msg  = 'Attriter';
    rep      = '2';                output;
run;
```

Real World Example

- create output file

```
%macro txt;
  data dialer ( drop = _: );
      if 0 then set dialer_skeleton;* set variable
                                      lengths/formats;

      array txtfile _character_;    * make available in array;

      set dialer_input ( rename = ( expiry = _exp ));

      expiry = put(_exp,yymmddd10.);
      length record $100;
      record = putc(txtfile(1),vformat(txtfile(1)))
      %do i = 2 %to &dim;    * loop limit from skeleton step;
             || putc(txtfile(&i),vformat(txtfile(&i)))
      %end;
                              ;
      put record;
  run;
%mend txt;

%txt
```

Real World Example

- output file

```
9052941234  Tim Brokking        2016-03-311st call        01
5196471212  Susan Jones         2016-04-01Second call     01
4162235678  Sam Brown           2016-04-02New customer    2
6137219988  David Wharton       2016-03-29Attriter        2
```

- modified requirements?
  - change the skeleton file
  - done

- Arrays rock

Wrap

- arrays
  - coding efficiency
  - eases maintenance
  - readability
  - productivity
  - makes you look smart ☺

Harry Droogendyk harry@stratia.ca

www.stratia.ca