Paper RF-04-2015

Reading a Column into a Row to Count N-levels, Calculate Cardinality Ratio and Create Frequency and Summary Output In One Step

Ronald J. Fehd, Stakana Analytics

Abstract

Description : This paper shows how read a column of numeric values into an array and use the sortn call routine in preparation for counting the number of levels of the variable. The primary goal of this algorithm is to calculate cardinality ratio which is n-levels divided by n-obs. This ratio can be used in Exploratory Data Analysis (EDA) to determine whether a variable is unique and therefore a row identifier, or discrete — a classification variable — or continuous — an analysis variable. A useful benefit of traversing the array and counting n-levels is that the frequency counts and percents can be accumulated. Another benefit of having the values in an array is the ability to calculate summary statistics.

Purpose : The purpose of this paper is to show an optimized algorithm for calculating cardinality ratio in one data step. Previous algorithms required three steps, contents for n-obs, frequency for n-levels and a data step for the calculation.

Audience : programmers, all levels

Keywords : cardinality ratio, n-levels, call routine `sortn`
data structure statements: `attrib, array, retain`
functions: `min, max, mean, median, std (std-dev)`
output from frequency and summary procedures

In this paper

Introduction

Overview

Cardinality Ratio (CR) was recognized as an important metric in classifiying variables in the continuing evolution of the Summarize-Each-Variable exploratory data analysis suite. These are the topics in this introduction.

- cardinality ratio definition
- previous algorithm
- elements of algorithm
- previous report

cardinality ratio definition

The cardinality of a set is the number of elements in the set. The cardinality of a data set is the number of rows of the data set. Within a data set the cardinality of a variable is the number of levels, i.e. distinct values, of the variable. The cardinality ratio (CR) of a variable is n-levels / n-obs. Its range is from >zero to one. Once the large numbers of n-levels and n-obs are reduced to a consistent range four categories are obvious: worthless (n-levels=1), few, many, and unique (CR=1). Intuitively it seems obvious to choose CR=0.5 as the separator of few and many. Extensive testing shows that mean(CR) is more accurate. This scale shows the range of CR.

	n-levels=1	few discrete		many continuous	unique
0			mean(CR)		1

- row-identifier — unique
- classification variables — discrete
- facts for summarization — continuous
- single value — worthless

previous algorithm

The Summarize-Each-Variable Suite uses this algorithm.
The goal of this paper is to reduce the number of steps.

1. proc contents out = out-contents(name n-obs-data) — sort by name
2. proc freq out = out-n-levels (name n-levels) — sort by name
3. data n-levels: merge out-contents out-n-levels, by name
   cardinality-ratio = n-levels /n-obs-data
4. proc summary mean-cr = mean(card-ratio) out = mean-cr
5. data card-ratio: merge n-levels mean-cr
   cr-type in (few, many, unique)
6. for variables cr-type eq few — proc freq
7. for variables type eq n and cr-type eq many — proc summary

elements of algorithm

Here are the essential elements of the algorithm which are implemented in this paper for numeric variables.

- data set name
- n-obs(data)
- n-levels = n-obs(proc freq output data set)
- cardinality-ratio = n-levels / n-obs
- for type eq numeric proc summary out = out-summary

---

previous report

This table is from previous publications about cardinality ratio. It clearly shows the three main categories of cardinality-ratio-type (cr-type): unique, few and many. It highlights the fact that while age is numeric and might therefore be considered an analysis variable, its relatively low n-levels indicates it is discrete and probably a classification variable.

cardinality ratios sashelp.class nobs=19 mean(CR)=0.62

cr-type	card-ratio	n-levels	varnum	name	type	length
unique	1.00	19	1	Name	c	8
few	0.10	2	2	Sex	c	1
few	0.31	6	3	Age	n	8
many	0.89	17	4	Height	n	8
many	0.78	15	5	Weight	n	8

Copy a Column into a Row

Overview

This suite has four programs. Copy the demonstration program and add the list of numeric variables for the data set being analyzed.

- demo sashelp.class
- get data information
- copy column into row
  - data structure
  - read column into row
  - counting n-levels
  - summarizing
- print the output
- example listing

demo sashelp.class

This is a demonstration program.

```
%let     data = sashelp.class;
%include 'get-data-info.sas';
%let     name = age;
%include 'read-column-into-row.sas';
*...;
%include 'proc-print-summaries.sas';
```

get data information

This program shows how to create a macro variable with the data attribute, number-of-observations. This macro variable is in the global symbol table and is available to all subsequent programs.

```
%let dsid  = %sysfunc(open (&data     ));
%let n_obs = %sysfunc(attrn(&dsid,nobs));
%let rc    = %sysfunc(close(&dsid     ));
```

data structure

The data structure of the data set contains compile-time statements.

```
DATA out_freq   (keep = name value count percent)
     out_summary(keep = name n_levels card_ratio ...);
     attrib name       length =$32
            n_levels   length =  8
            card_ratio length =  8;
     array  _n(&n_obs) _temporary_;
     retain name "&name";
```

Caveat: This program is fragile; for very large data sets the program may run out of memory when allocating this array.

---

read column into row

This program paragraph shows how to read all the values in a vertical column into a horizontal row, i.e., the array.
Note the use of the macro variable `n_obs` as the dimension of the array and the upper bound of the `do` loop.
Renaming the variable being read avoids naming collisions, a variable in the data set with the same name as a variable in this data step.

```
   array  _n(&n_obs);
do _i = 1 to &n_obs ;   * read column into row;
   set &data (keep   =  &name
              rename = (&name = _value))
       point = _i;
   _n(_i) = _value;
   end;

call sortn(of _n(*));
```

Once the data are in an array, use the `sortn` function, which groups like values together in preparation for counting levels. This function eliminates the need to write a multi-line sorting algorithm such as bubble-sort, heap-sort, insertion-sort or quick-sort.

---

counting n-levels

This paragraph shows the loop which replaces the frequency procedure and counting of number of levels (n-levels). Note the use of the macro variable `n_obs` as the upper bound of the `do` loop.

```
** initialize with row.1 values;
value = _n(1);* also: previous-value;

do _i = 2 to &n_obs;
** if this-value ne previous-value;
   if _n(_i)     ne value then do;
      output out_freq;
      n_levels + 1;
      value    = _n(_i);*current-value;
      count    = 0;
      end;
   count       + 1;
   end;
card_ratio = n_levels / &n_obs;
```

After n-levels is counted then cardinality-ratio can be calculated.

---

summarizing

This snippet shows only two functions for calculating statistics from an array; this paragraph replaces the mean procedure and calculation of a five-number summary.

```
** calculate summary statistics;
min = min(of _n(*));
max = max(of _n(*));
*...;
output out_summary;
stop;
run;
PROC append data = out_freq    base = list_frequencies;
PROC append data = out_summary base = list_summaries;
```

After each call for a variable the out-* data sets are appended to the list-* data sets.

Note: Refer to the page *SAS Functions and CALL Routines by Category* for other descriptive statistics that can be used with arrays.

---

print the output

This subroutine prints the two output data sets, list-frequencies and -summaries.

```
PROC print data   = list_summaries   ;
           title4       'summaries'  ;
PROC print data   = list_frequencies ;
           title4       'frequencies';
```

---

example listing

This report is the output for this suite.
Compare to the predecessor on page 3.

```
data sashelp.class n-obs=19
summaries
         n     card.ratio:
name   levels  n-levels/19     n    mean  std_dev   min  max

age         6      0.31579    19   13.31    1.492  11.0   16
height     17      0.89474    19   62.33    5.127  51.3   72
weight     15      0.78947    19  100.02   22.773  50.5  150

frequencies
name      value    count    percent
age        11.0      2      10.5263
           12.0      5      26.3158
           13.0      3      15.7895
           14.0      4      21.0526
           15.0      4      21.0526
           16.0      1       5.2632
```

Next task: This suite calculates cardinality ratio for numeric variables. The next task is to copy and modify the program to handle character variables. After processing the cardinality ratio for all variables of a data set then the last program can calculate the mean of CR and cr-type.

Program Listings

Overview

This section lists the programs developed for this paper.

- demo sashelp.class
- get data information
- print summaries
- read column into row

demo-sashelp-class.sas

Copy this program and modify it with the numeric variables from another data set.

```
%let data = sashelp.class;

%include 'get-data-info.sas';

%let name    = age;
%include 'read-column-into-row.sas';

%let name    = height;
%include 'read-column-into-row.sas';

%let name    = weight;
%include 'read-column-into-row.sas';

%include 'proc-print-summaries.sas';
```

get-data-info.sas

```
%let dsid  = %sysfunc(open (&data     ));
%let n_obs = %sysfunc(attrn(&dsid,nobs));
%let rc    = %sysfunc(close(&dsid     ));
%put echo &=data &=n_obs;
%symdel dsid rc;
PROC contents data = &data;
title3 "data &data n-obs=&n_obs";
run;
```

proc-print-summaries.sas

```
PROC print data = list_summaries   noobs label;
           title4     'summaries'  ;
PROC print data = list_frequencies noobs;
           title4     'frequencies';
           by     name;
           id     name;
run;
```

read-column-into-row.sas

```
%put read-column-into-row beginning &=name &=n_obs;

DATA out_freq   (keep   = name value count percent)
     out_summary(keep   = name n_levels card_ratio n n_miss
                          mean std_dev median min max);
     attrib name       length =$32  value   length = 8
            count      length =  8  percent length = 8
            n_levels   length =  8
```

```
             card_ratio length =  8 label  =
            "cardinality ratio: n-levels/&n_obs"
             n          length =  8  n_miss  length = 8
             min        length =  8  max     length = 8
             mean       length =  8  median  length = 8
             std_dev    length =  8;
      array  _n(&n_obs) _temporary_;
      retain name "&name";

** read column into this row :: array;
do _i = 1 to &n_obs;
   *** avoid naming collisions by renaming;
   set &data (keep   =  &name
              rename = (&name = _value))
       point = _i;
   _n(_i) = _value;
   end;

**** sort values for counting n-levels;
call sortn(of _n(*));

** initialize with row.1 values;
value   = _n(1);* also: previous-value;
count    =    1;
percent  =    100*(count/&n_obs);
n_levels =    1;

** start counting n-levels from row.2;
do _i = 2 to &n_obs;
** if this-value ne previous-value;
   if _n(_i)     ne value then do;
      output out_freq;
      n_levels + 1;
      value = _n(_i);
      count  = 0;
      end;
   count   + 1;
   percent = 100*(count/&n_obs);
   end;
output out_freq;

card_ratio = n_levels / &n_obs;

** calculate summary.7 statistics;
n       = n     (of _n(*));* 7;
n_miss  = nmiss (of _n(*));* 7;
min     = min   (of _n(*));*5;
max     = max   (of _n(*));*5;
mean    = mean  (of _n(*));*5;
median  = median(of _n(*));*5;
std_dev = std   (of _n(*));*5;
output out_summary;
stop;
run;
PROC append data =  out_freq
            base = list_frequencies;
PROC append data =  out_summary
            base = list_summaries;
run;
%put read-column-into-row ending &=name;
```

Summary

Conclusion

These programs show that examining the input, process and output of a suite of programs can lead to a lesser description of the algorithm. Thoughtful consideration of the process and a knowledge of functions that can produce the same results can lead to a less complicated and easier to modify algorithm.

Suggested Reading

predecessor :	Fehd [5], Summarize-Each-Variable Suite; Fehd [6], SmryEachVar: A Data-Review Routine;
cardinality ratio :	Fehd [4], Cardinality Ratio; Fehd [8], Data Review Information: N-Levels or Cardinality Ratio; Fehd [7], Database Vocabulary, cardinality ratio
arrays :	Droogendyk [3], using arrays for efficiency
sorting :	Cody [1], survey of functions; Dorfman [2], quick sort; Jia and Lin [9], horizontal sorting; Staff [10], the varieties of sorting experience
programs :	in this paper are available here: http://www.sascommunity.org/wiki/ Read Column Into Row

Contact Information: **Ronald J. Fehd** mailto:Ron.Fehd.macro.maven@gmail.com
http://www.sascommunity.org/wiki/Ronald_J._Fehd

About the author:

education:	B.S. Computer Science, U/Hawaii,	1986
	SAS User Group conference attendee since	1989
experience:	programmer: 30+ years	
	author: 40+ SUG papers	
	sas.community.org wiki: 400+ pages	
SAS-L:	author: 7,000+ messages to SAS-L since	1997

Bibliography

[1] Ron Cody. A survey of some of the most useful SAS(R) functions. In *SAS Global Forum*, 2012. URL http://support.sas.com/resources/papers/proceedings12/241-2012.pdf. 16 pp.; character and numeric functions and call routines for advanced data step programming.

[2] Paul M. Dorfman. QuickSorting an array. In *SUGI-26*, 2001. URL http://www2.sas.com/proceedings/sugi26/p096-26.pdf. 6 pp.; a walk-through of Hoare's QuickSort algorithm.

[3] Harry Droogendyk. Arrays — data step efficiency. In *SESUG*, 2011. URL http://analytics.ncsu.edu/sesug/2011/CC17.Droogendyk.pdf. 9 pp.; using arrays and getting variable information about an array element using the vformat function.

[4] Editor R.J. Fehd. Cardinality-ratio. In *sas community.org Wikipedia*, 2008. URL http://www.sascommunity.org/wiki/Cardinality_Ratio. topics: definition and programs.

[5] Editor R.J. Fehd. SmryEachVar: A data-review suite for each variable in all data sets in a libref. In *sas community.org Wikipedia*, 2008. URL http://www.sascommunity.org/wiki/SmryEachVar_A_Data_Review_Suite. list processing using parameterized includes.

[6] Ronald J. Fehd. SmryEachVar: A data-review routine for all data sets in a libref. In *SAS Global Forum Annual Conference Proceedings*, 2008. URL http://www2.sas.com/proceedings/forum2008/003-2008.pdf. Applications Development, 24 pp.; call execute, data review, data structure, dynamic programming, list processing, parameterized includes, utilities (writattr, writvalu) to repair missing elements in data structure; best contributed paper in ApDev.

[7] Ronald J. Fehd. Database vocabulary: Is your data set a dimension (lookup) table, a fact table or a report? In *Proceedings of the Western Users of SAS Software Annual Conference*, 2008. URL http://wuss.org/proceedings08/08WUSS%2520Proceedings/papers/dmw/dmw04.pdf. Databases and Warehouses, 8 pp.; cardinality ratio, composite key, database design, foreign key, grain, nlevels, normal forms, primary key, relational database, snapshots: accumulating or periodic.

[8] Ronald J. Fehd. Data review information: N-levels or cardinality ratio. In *SAS Global Forum Annual Conference Proceedings*, 2013. URL http://support.sas.com/resources/papers/proceedings13/299-2013.pdf. Statistics and Data Analysis, 6 pp.; using proc freq nlevels and nobs to calculate cardinality ratio — range in (0:1) — of a variable to determine its type in (continuous, discrete, unique, worthless).

[9] Justin Jia and Amanda Lin. Horizontal data sorting and insightful reporting: A useful SAS(R) technique. In *SAS Global Forum*, 2013. URL http://support.sas.com/resources/papers/proceedings13/376-2013.pdf. 14 pp.; practical application of data manipulation.

[10] Wiki Staff. Sorting algorithm. In *Wikipedia, The Free Encyclopedia*, 2001. URL https://en.wikipedia.org/wiki/Sorting_algorithm. definition; classification; comparison of algorithms, classes of sorting: simple, efficient, bubble and distribution.

The road to wisdom?
Well, it's plain and simple to express.
Err and err and err again
but less and less and less.

– Danish mathematician,
poet Piet Hein (1905–1996)