Using Cardinality Ratio for Fast Data Review
Ronald J. Fehd, Stakana Analytics, Atlanta, GA, USA

Abstract

Description: This paper reviews the database concept Cardinality Ratio. The SAS® frequency procedure can produce an output data set with a list of the values of a variable. The number of observations of that data set is called N-Levels. The quotient of N-Levels divided by the number-of-observations of the data is the variable's Cardinality Ratio (CR). Its range is in (0–1] and four categories of variables' CR are obvious: useless, discrete, continuous and unique.

Purpose: Cardinality Ratio provides an important value during data review. The four groups of values are used to create control data sets which can automate data review. Macros are provided that facilitate fast data review.

Audience: data managers and programmers.

Programs: in this paper are available in Fehd [5].

Keywords: continuous, database, dimensionless, discrete, fact, frequency, keys: foreign or primary, nlevels, number of observations (nobs), unique

Quote: Information is the difference that makes a difference. Gregory Bateson 1904–1980 Steps to an Ecology of Mind, 1972 italics added by R.J. Fehd

Contents

Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cr-calc.sas</td>
<td>4</td>
</tr>
<tr>
<td>listvars.sas</td>
<td>7</td>
</tr>
<tr>
<td>cr-data.sas</td>
<td>8</td>
</tr>
<tr>
<td>cr-freq.sas</td>
<td>8</td>
</tr>
<tr>
<td>cr-mode.sas</td>
<td>9</td>
</tr>
<tr>
<td>cr-smry.sas</td>
<td>10</td>
</tr>
</tbody>
</table>

Unit Tests

<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cr-calc-test-class.sas</td>
<td>12</td>
</tr>
<tr>
<td>cr-calc-test-class-with-format.sas</td>
<td>13</td>
</tr>
<tr>
<td>cr-calc-test-shoes.sas</td>
<td>13</td>
</tr>
</tbody>
</table>

Summary

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

Bibliography

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
</tr>
</tbody>
</table>
Introduction

Overview

When starting a new project a programmer or data detective may use several procedures during data discovery to confirm the relationships between variables in a data set.

The cardinality ratio (CR) of a variable is the quotient of the number of levels of that variable divided by the number of rows of the data set. The dimension: n-rows of numerator and denominator, cancels out leaving a pure number in the range of > zero – one.

CR is similar in concept to the log function: it reduces large numbers to a finite range which makes comparisons easier to grasp.

The Sets of Values

CR can be grouped into four categories.

- continuous: CR ≥ 0.5
- discrete: CR ≤ 0.5
- unique: CR = 1
- single-valued: n-levels = 1

Note: One-half (0.5) is an arbitrary separation value.

continuous: information: is a fact variable; if numeric can be summarized

discrete: character variables have standardized case: either upper or lower; numerics may be integers, or in a small finite range

information: is a classification variable;

todo: locate one-to-one formats, or dimension (lookup) tables

unique: variable is a row-identifier; if numeric and the range is exactly 1:n-observations then it is the row-number

information: is a primary key;

single-valued: values may be:

- character: blank
- numeric: missing
- a single value, indicating data set is a subset of larger data

information: worthless, discard

Task

The task of this article is to show that the mean of the cardinality ratios separates the discrete from the continuous; the former is summarized with the frequency procedure and the latter with the summary procedure.
These are the steps in the cardinality-ratio calculator program.

1. \textit{listvars}: make list of variable names from contents
2. \textit{freq}: make list of n-levels
   \hspace{1cm} out= print n-levels
3. sort and merge contents and frequency output data sets
   \hspace{1cm} calculate cardinality ratio
   \hspace{1cm} out=CR-list
4. calculate mean of CR
5. add mean to CR-list
   \hspace{1cm} assign categories: unique, discrete, continuous
   \hspace{1cm} make lists for frequency of discrete,
   \hspace{1cm} mode of either, and summary of continuous
6. print CR-list with categories
7. for each list, call macros of frequency, mode, summary,
   \hspace{1cm} out = CR-summaries
8. print CR-summaries

\textbf{Note:} This algorithm is from Fehd [7], SmryEachVar and Fehd [9], Cardinality Ratio.

These are the steps in the cardinality-ratio summarization routines, cr-frequency, cr-mode and cr-summary.

1. procedure, out=out-from-proc
2. data for-append, standardize data structure: cr-data
3. append
   \hspace{1cm} base = CR-summaries
   \hspace{1cm} data = out-from-proc

This is the common data structure provided by the subroutine cr-data.

```
create table WORK.OUT_FREQUENCIES( bufsize=16384 )
libname char(8),
memname char(32),
name char(32),
valu_c char(32),
valu_txt char(32),
valu_n num,
count num format=COMMA. label='N',
percent num format=6.2 label='-%-',
```

\textbf{Notes:} Each procedure renames the variable according to its type. Character variables are renamed to \texttt{valu_c}; numeric variables are renamed to \texttt{valu_n}; the mode procedure and formated values are written into \texttt{valu_txt}. 


Using Cardinality Ratio for Fast Data Review, continued

Programs

Overview

This section lists the suite of programs that produce a report of the cardinality ratios of variables in a data set.

- cardinality ratio calculator
- listvars, make list of variables from contents
- cr-data, standardized data structure for cr-*
- cr-frequency
- cr-mode, proc mode is proc freq order = freq
- cr-summary

Note: The listing of list-processing macro callmacr.sas is in Fehd [11].

**cr-calc.sas**

```sas
/* name: <UNC>\SAS-site\macros\cr_calc.sas
author: Ronald J. Fehd 2012, 2013
-------------------------------------------
Summary : description : cardinality ratio calculator
purpose : data review
-------------------------------------------
Contexts : program group: list processing program
program type: module
SAS type: macro routine
uses routines : callmacr to call:
uses subroutines: cr_freq, cr_mode, cr_smry
-------------------------------------------
Specifications: input : required: libname, memname
process: contents out=
freq nlevels out=
data: merge contents + freq,
calculate CR
summary: mean of CR
data: add category is_a,
output various
output: save and print: out-lib.out-data
-------------------------------------------
Parameters : libname : actually the libref
memname : data set name
out_lib : work | library
out_data : check the default: cr_&memname
out_smry : kest_lib..smry_ou-data
-------------------------------------------
Usage Example:
*options mprint source2;
%cr_calc(libname = sashelp
       ,memname = shoes)
Notes:
+ certain data sets will generate these warnings:
WARNING: The variable nmisslevels in the DROP, KEEP,
or RENAME list has never been referenced.
WARNING: The variable nonmisslevels in the DROP, KEEP,
or RENAME list has never been referenced.
when present, these variables are in:
create table FREQ_N_LEVELS
(name char(14) label='Number of Variable Levels',
TableVarLabel char(28) label='Table Variable',
n_levels num format=BEST8. label='Number of Levels',
NMissLevels num format=BEST8.
label='Number of Missing Levels',
NNonMissLevels num format=BEST8.
```

4
Using Cardinality Ratio for Fast Data Review, continued

* this is a Derivative Work of these suites:
  ** SmryEachVar
  http://www.sascommunity.org/wiki/
  SmryEachVar_A_Data_Review_Suite
  ** Cardinality Ratio
  http://www.sascommunity.org/wiki/Cardinality_Ratio

%MACRO cr_calc
  %let libname = sashelp;
  %let memname = class;
  %let out_lib = work;
  %let out_data = cr_&memname;
  %let out_smry = cr_&memname._smry;
  %let mode_show = 3;
  %let mode_nlevels = 33; /* fib: 1 1 2 3 4 8 13 21 33 */
  %let testing = 0;
  %let des = 'site: cardinality ratio calculator';
  %let out_contents = list_names;
  %let out_freq = freq_n_levels;
  %let testing = %eval( not(0 eq &testing) or %sysfunc(getoption(mprint)) eq %upcase(mprint));

  %listvars(data = &libname..&memname, drop_keep = keep = libname memname name type format nobs, out = &out_contents)

  *** make data set with nlevels;
  PROC freq data = &libname..&memname nlevels;
  ods exclude onewayfreqs;
  ods output nlevels = &out_freq(rename = (tablevar = name nlevels = n_levels));
  title3 "n-levels of &libname..&memname;"
  %if &testing %then %do;
    PROC sql; describe table &syslast.;
    quit;
  %end;

  *** prepare for merge;
  PROC sort data = &out_contents out = &out_contents;
  by libname memname name;
  PROC sort data = &out_freq out = &out_freq;
  by name;
  DATA &out_data
    (keep = card_ratio libname memname name type format label n_levels nobs nmisslevels nnonmisslevels);
    if 0 then do;
      attrib card_ratio length = 8
      label = 'cardinality ratio';
      set &out_contents (obs = 0);
      set &out_freq (obs = 0);
    end;
    do until(endofile);
      merge &out_contents &out_freq end = endofile;
      by name;
      card_ratio = n_levels/nobs;
      libname = lowcase(libname);
      memname = lowcase(memname);
      output;
    end;
    call symputx('nobs',nobs);
  stop;
run;
%if &testing %then %do;
  PROC sql; describe table &syslast.;
  quit;
%end;

**** calculate middle of CR range:
separates foreign keys::discrete and facts::continuous;
PROC summary data = &out_data
  mean;
  var card_ratio;
  output out = mean
  (keep = mean)
  mean (card_ratio) = mean;
%if &testing %then %do;
  PROC sql; describe table &syslast;
  quit;
%end;

**** save mean for use in calculation, label and title;
PROC sql noprint;
  select mean into :mean
  from &syslast;
  quit;
%put note: &sysmacroname mean: &mean;
run;
%if &mean eq 1 %then %do;
  %put note: &sysmacroname exit: mean eq 1;
  %return;
%end;

%let drop_list = drop = is_a card_ratio;
DATA &out_data
  cr_list_freq(&drop_list)
  cr_list_mode(&drop_list)
  cr_list_smry(&drop_list);
  keep is_a card_ratio libname memname name type format;
  if 0 then do;
    attrib is_a length = $ %length(_primary key?!);
    set &out_data
      (keep = card_ratio n_levels
       libname memname name type format);
    end;
    set mean;
  do until(endofile);
    set &out_data end = endofile;
    if card_ratio eq 1 then
      is_a = ‘primary key?!’;
    else if n_levels eq 1 then
      is_a = ‘nlevels=1’;
    else if card_ratio le mean then do;
      is_a = ‘foreign key’;
      if n_levels le &mode_nlevels then output cr_list_freq;
      else output cr_list_mode;
    end;
    else if card_ratio gt mean then do;
      is_a = ‘fact’;
      if type = ‘c’ then output cr_list_mode;
      else if type = ‘n’ then output cr_list_smry;
      end;
    output &out_data;
  end;
  stop;
  run;
%if &testing %then %do;
  PROC sql; describe table &syslast.;
  quit;
%end;
%proc print data =cr_list_freq;title1 cr_list_freq;
%proc print data =cr_list_mode;title1 cr_list_mode;
%proc print data =cr_list_smry;title1 cr_list_smry;
%end;
Using Cardinality Ratio for Fast Data Review, continued

PROC sort data = &out_data
   out = &out_lib..&out_data
   (label="&out_lib..&out_data nobs=&nobs mean=&mean")
   by is_a name card_ratio;
PROC print data = &out_lib..&out_data;
   title3 "cardinality ratios of &libname..&memname"
   nobs=&nobs mean=&mean"
   by is_a;
id is_a;
run;

**** use routine to call subroutines;
%callmacr(data = cr_list_freq
   ,macro_name = cr_freq
   ,hex16 = 0)
*** note: hex16 = 0 :: pass varnum as integer;
%callmacr(data = cr_list_mode
   ,macro_name = cr_mode
   ,macro_parms = %nrstr(mode_hi_lo=&mode_show)
   ,hex16 = 0)
%callmacr(data = cr_list_smry
   ,macro_name = cr_smry
   ,hex16 = 0)
%callmacr(data = cr_list_smry
   ,macro_name = cr_univ
   ,hex16 = 0)
PROC datasets library = work
   nodetails nolist;
change
   cr_summaries = &out_smry;
%if %lowcase(&out_lib) ne work %then %do;
PROC copy in = work
   out = &out_lib;
select &out_smry;
%end;
proc print data = &out_lib..&out_smry;
   title4 &out_lib..&out_smry;
run;
***
%mend cr_calc;

listvars.sas
/*
 * name: <UNC>SAS-site>macros>listvars.sas
 * Make List of Variable Names
 * description: make list of variable names
 * input : data : _all_ | data set name
 *         drop_keep: keep=libname memname name type
 *         where : 1 | logical expression
 * purpose: prepare list processing data set
 * output: out =
 * usage:
 * %listvars(data = sashelp.class
 *   ,drop_keep =
 *   ,where = 1
 *   ,out = list_vars
 *   ,testing = 0)
 * see also: http://www.sascommunity.org/wiki/Making_Lists **/
 * %MACRO listvars
 * (data = sashelp.class
 *   ,drop_keep =
 *   ,where = 1
 *   ,out = list_vars
 *   ,testing = 0)
 * / des =site from contents make list of variable names';
 * %let testing = %eval( not(0 eq &testing)
 *      or %sysfunc(getoption(mprint)) eq %upcase(mprint));
 * PROC contents data = &data
 *   noprintr
 *   out = &out
 *   (keep = libname memname memtype varnum
 *     name type length label format nobs
 *     keep = libname memname name type format)
 */
rename = (type = type_n)
where = (where
    and memtype eq 'DATA');
DATA &out (&drop_keep
    label = "list vars of &data";
    attr libname length =$ 8 label = 'libname'
    memname length =$ 32 label = 'mem name'
    varnum length =$ 8 label = 'var num'
    name_lc length =$ 32 label = 'low (var name)'
    name length =$ 32 label = 'var name'
    type length =$ 8 label = 'type'
    length length =$ 32 label = 'length'
    format length =$ 32 label = 'format'
    label length =$256 label = 'label'
    nobs length = 8 label = 'n obs';
do until(endofile);
set &out(drop = memtype)
end = endofile;
libname = lowcase(libname);
memname = lowcase(memname);
name_lc = lowcase(name);
* align format, type to sql.dictionary.columns:
1. convert type from numeric to $char1
2. add char.prefix $ and suffix dot to formats;
if format ne ' ' then
    format = catt(format,'.);
if type_n eq 1 then type = 'n';
else if type_n eq 2 then type = 'c';
output &out;
end;
stop;
run;
%mend listvars;

%if &testing %then %do;
proc sql; describe table &out;
quit;
%mend;
%mend cr_data;

/* name: <UNC>SAS-site.macros\cr_data.sas
author: Ronald J. Fehd 2013
Summary : description : common data structure of cr_ *
purpose : standardization
Contexts: program group: list processing program
program type: subroutine
SAS type: macro subroutine
called by routines cr_* ***** /
%Macro cr_data(_)
/des = 'site: card. ratio suite: common data structure';
keep libname memname name
valu_c valu_txt valu_n count percent;
attrib libname length =$8
memname length =$32
name length =$32
valu_c length =$32
valu_txt length =$32 %* from format or mode;
valu_n length = 8
count length = 8 format = comma. label = 'N'
percent length = 8 format = 6.2 label = '-%-';
retain libname "lowcase(&libname)"
memname "lowcase(&memname)"
name "&name"
valu_c "." valu_txt "."
valu_n . count . percent .;
%mend cr_data;

/* name: cr_freq.sas
author: Ronald J. Fehd 2013
Using Cardinality Ratio for Fast Data Review, continued

---

**Summary**
- Description: proc freq
- Purpose: standardize output

**Contexts**
- Program group: list processing program
- Program type: routine
- SAS type: macro routine

**Specifications**
- Input: libname memname name type format
- Process: freq out= ***

```sas
%macro cr_freq
  (libname = ,memname = ,name = ,type = ,format = ,out_append = cr_summaries ,out_data = out_frequencies ,testing = 0)
  / des = 'site: list processing proc freq';
  %let testing = %eval( not(0 eq &testing) or %sysfunc(getoption(mprint)) eq %upcase(mprint));
  PROC freq data = &libname..&memname;
  format &name; *remove formats;
  tables &name / list missing noprint
  out = &out_data (rename = (&name = valu_&type));
  DATA &out_data;
  %cr_data(attributes);
  do until(endofile);
    set &out_data end = endofile;
    %if &format ne %then %do;
      valu_txt = put(valu_&type,&format);
    %end;
    output;
    end;
  stop;
  run;
  %if &testing %then %do;
    PROC sql; describe table &syslast; quit;
  %end;
  PROC append base = &out_append force data = &out_data;
  PROC delete data = &out_data;
  run;
%mend cr_freq;
```

---

**cr-mode.sas**

```sas
/* name: <UNC>\SAS-site\macros\cr_mode.sas
author: Ronald J. Fehd 2013
Summary : description : calculate modes, hi and low
purpose : standardize output
Contexts : program group: list processing program
program type: routine
SAS type: macro routine
Specifications: input : libname, memname, name, type
process: freq order=freq out= ***
%macro cr_mode
  (libname = ,memname = ,name = ,type = ,format = ,mode_hi_lo = 3 ,out_append = cr_summaries ,out_data = out_modes ,testing = 0)
  / des = 'site: card. ratio suite: proc mode';
```

---
%let testing = %eval( not(0 eq &testing) or %sysfunc(getoption(mprint)) eq %upcase(mprint));

PROC freq data = &libname..&memname
order = freq;
format &name; remove formats;
tables &name
  / list missing noprint
  out = &out_data
  (rename = (&name = valu_&type));
DATA &out_data;
%cr_data(attributes);

do rownmbr = 1 to &mode_hi_lo;
  set &out_data nobs = n_rows point = rownmbr;
  valu_txt = "mode: high &mode_hi_lo";
  output;
end;

do rownmbr = n_rows +1 - &mode_hi_lo to n_rows;
  set &out_data nobs = n_rows point = rownmbr;
  valu_txt = "mode: low &mode_hi_lo";
  output;
end;
stop;
run;

%if &testing %then %do;
  PROC sql; describe table &syslast;
  quit;
%end;

PROC append base = &out_append force data = &out_data;
PROC delete data = &out_data;
run;
%mend cr_mode;

cr-smry.sas

/* name: cr_smry.sas
author: Ronald J. Pehd 2013
Summary : description : summarize numeric variable
purpose : standardize output
Contexts : program group: list processing program
          program type: routine
SAS type: macro routine
Specifications: input : libname memname name type format
process: summary out= ***/
%macro cr_smry
  (libname =
   ,memname =
   ,name =
   ,type =
   ,format =
   ,out_append = summary
   ,out_data = out_summaries
   ,testing = 0)
  / des = 'site: card. ratio suite: proc summary';
%let testing = %eval( not(0 eq &testing) or %sysfunc(getoption(mprint)) eq %upcase(mprint));

PROC summary data = &libname..&memname;
format &name; remove formats;
var &name;
output
  out = summary
  ( drop = _type_ _freq_ )
n  ( &name ) = n
nmiss ( &name ) = nmiss
min ( &name ) = min
p25 ( &name ) = p25 %*q1;
mean ( &name ) = mean %*average;
median ( &name ) = median %*p50:%middle;

10
Using Cardinality Ratio for Fast Data Review, continued

```
p75 (\&name) = p75 \%*q3;
max (\&name) = max ;

PROC transpose data = summary
  out = summary_t
  (keep = col1 valu_c
   rename = (col1 = valu_n ))
  name = valu_c ;
DATA \&out_data;
%cr_data(attributes);
if 0 then set \&libname..\&memname(obs=0)
nobs = nobs_data;
do until(endofile);
  set summary_t end = endofile;
  if valu_c in ('n','nmiss') then
    percent = 100*(valu_n/nobs_data);
  else percent = .;
  %if \&format ne %then %do;
    valu_txt = put(\&name,\&format);
  %end;
  output;
end;
stop;
run;

%if \&testing %then %do;
  PROC sql;
describe table \&syslast;
quit;
%end;

PROC append base = \&out_append
  force
  data = \&out_data;
PROC delete data = \&out_data;
PROC delete data = summary;
PROC delete data = summary_t;
run;
%mend cr_smry;
```

Unit Tests

Overview

This section lists the set of programs used to do unit tests of the cardinality-ratio-calculator suite.

- cr-calc-test-class
- cr-calc-test-class-with-format, shows valu_txt with format
- cr-calc-test-shoes, shows mode in valu_txt

Continued on next page.
Using Cardinality Ratio for Fast Data Review, continued

**cr-calc-test-class.sas**

```
program ____________________________ cr-calc-test-class.sas ____________________________

options mprint source2;
%cr_calc(libname = sashelp ,memname = class)
```

<table>
<thead>
<tr>
<th>Number of Variable Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Levels</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Sex</td>
</tr>
<tr>
<td>Age</td>
</tr>
<tr>
<td>Height</td>
</tr>
<tr>
<td>Weight</td>
</tr>
</tbody>
</table>

cardinality ratios of sashelp.class nobs=19 mean=0.621053

card_
is_a ratio libname memname name type format
primary key?! 1.00000 sashelp class Name c
fact 0.89474 sashelp class Height n
0.78947 sashelp class Weight n
foreign key 0.31579 sashelp class Age n
0.10526 sashelp class Sex c

work.cr_class_smry
libname memname name valu_c valu_txt valu_n count percent
sashelp class Age . . 11.000 2 10.53
sashelp class Age . . 12.000 5 26.32
sashelp class Sex . . F 9 47.37
sashelp class Sex . . M . 10 52.63
sashelp class Height . . 19.000 . 100.00
sashelp class Height nmiss . . 0.000 . 0.00
sashelp class Height min . . 51.300 . .
sashelp class Height mean . . 62.337 . .
sashelp class Height median . . 62.800 . .
sashelp class Weight . . 19.000 . 100.00
sashelp class Weight nmiss . . 0.000 . 0.00
sashelp class Weight max . . 150.000 . .

Continued on next page.
cr-calc-test-class-with-format.sas

program

This test program shows a report of a variable with a format.

```sas
options mprint source2;
proc format;
value $sex 'F' = 'female'
'M' = 'male';
data class_with_format;
if 0 then set sashelp.class;
attrib sex length = $1 format = $sex.;
do until(EndOfFile);
set sashelp.class end = endofile;
output;
end;
Name = 'Chris'; Sex = 'I';
output;
end;
run;
proc sql; describe table &syslast;
quit;
%cr_calc(libname = work
,memname = class_with_format)
```

report

Formats of values are in variable valu_txt.

```sas
<table>
<thead>
<tr>
<th>cardinality ratios of work.class_with_format nobs=20 mean= 0.61</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_a ratio libname memname name type format</td>
</tr>
<tr>
<td>foreign key 0.15 work class_with_format sex c $SEX.</td>
</tr>
<tr>
<td>libname memname name valu_c valu_txt valu_n count percent</td>
</tr>
<tr>
<td>work class_with_format sex F female . 9 45.00</td>
</tr>
<tr>
<td>work class_with_format sex M male . 10 50.00</td>
</tr>
</tbody>
</table>
```

---

cr-calc-test-shoes.sas

program

This test program shows output from the mode procedure.

```sas
options mprint source2;
%cr_calc(libname = sashelp
,memname = shoes)
```

report

Variable Subsidiary is a foreign key with n-levels greater than 33, the default value of cr_calc parameter mode_nlevels; instead of proc freq, the mode macro is used. Note the high values in variable valu_txt.

```sas
<table>
<thead>
<tr>
<th>Variable Label Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsidiary 53</td>
</tr>
<tr>
<td>card_ is_a ratio libname memname name type format</td>
</tr>
<tr>
<td>foreign key 0.13418 sashelp shoes Subsidiary c</td>
</tr>
<tr>
<td>libname memname name valu_c valu_txt valu_n count percent</td>
</tr>
<tr>
<td>sashelp shoes Subsidiary Addis Ababa mode: high 3 . 8 2.03</td>
</tr>
<tr>
<td>sashelp shoes Subsidiary Al-Khobar mode: high 3 . 8 2.03</td>
</tr>
<tr>
<td>sashelp shoes Subsidiary Budapest mode: high 3 . 8 2.03</td>
</tr>
</tbody>
</table>
```
Using Cardinality Ratio for Fast Data Review, continued

Summary

Cardinality Ratio is valuable information to have in data review. Its small finite range is easier to parse for meaning than the constantly changing and larger number-of-observations (nobs) of the data set. The mean of CR offers an easy way to differentiate between discrete and continuous variables, which are processed with frequency for the former and summary for the latter.

Further Reading

Programs: for this paper are in Fehd [5].
Predecessors: This paper was first published in Fehd [9].

Fehd [7] (SmryEachVar) developed a suite of programs to return a list of the frequencies of each variable in a data set or libref.

Cardinality Ratio is identified in Fehd [8] for which SmryEachVar is the predecessor.

Programs for SmryEachVar which includes calculations for Cardinality Ratio are here: Fehd [6].

Call-Macro: The list processing routine Call-Macro is described in Fehd [11].

Theory: The macros in this article illustrate the macro design ideas presented in Fehd [10].

Contributors [4] provides the description of a ratio as a dimensionless quantity, a pure number.

Contributors [1], [2] and [3] discuss the differences between continuous and discrete variables.

References


Closure

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
SAS-L reader since 1994

experience: programmer: 25+ years
data manager using SAS: 17+ years
statistical software help desk: 7+ years
author: 30+ SUG papers
sasCommunity.org: 300+ pages

SAS-L: author: 6,000+ messages to SAS-L since 1997

Trademarks

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. In the USA and other countries ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.