

Efficiencies of Using Count Data Rather than Raw Data in Regression

Steve Fleming

National Center for Educational Accountability

sfleming@just4kids.org

Abstract

Running SAS regression procedures such as PROC REG and LOESS on large datasets can be quite time-consuming. However, when the data can be summarized as a smaller set of unique values coupled with a count of the number of cases having each unique value, a significant amount of time can be saved for some procedures. A motivating example involving prediction of scale scores on the Texas Assessment of Knowledge and Skills (TAKS) using prior year scores will be presented. Simulated data will be used to assess how the number of records and level of data redundancy affect the time saved.

Introduction

In my work, I often run large datasets with more than 100,000 records through regression procedures. This can be very time-consuming particular for computationally intensive methods. For certain data structures, processing time can be cut dramatically by summarizing the data as counts of each unique combination of the variables and using the FREQ statement. My initial motivation for pursuing this approach was to cut the time needed to open high-resolution scatterplots in PDF format.

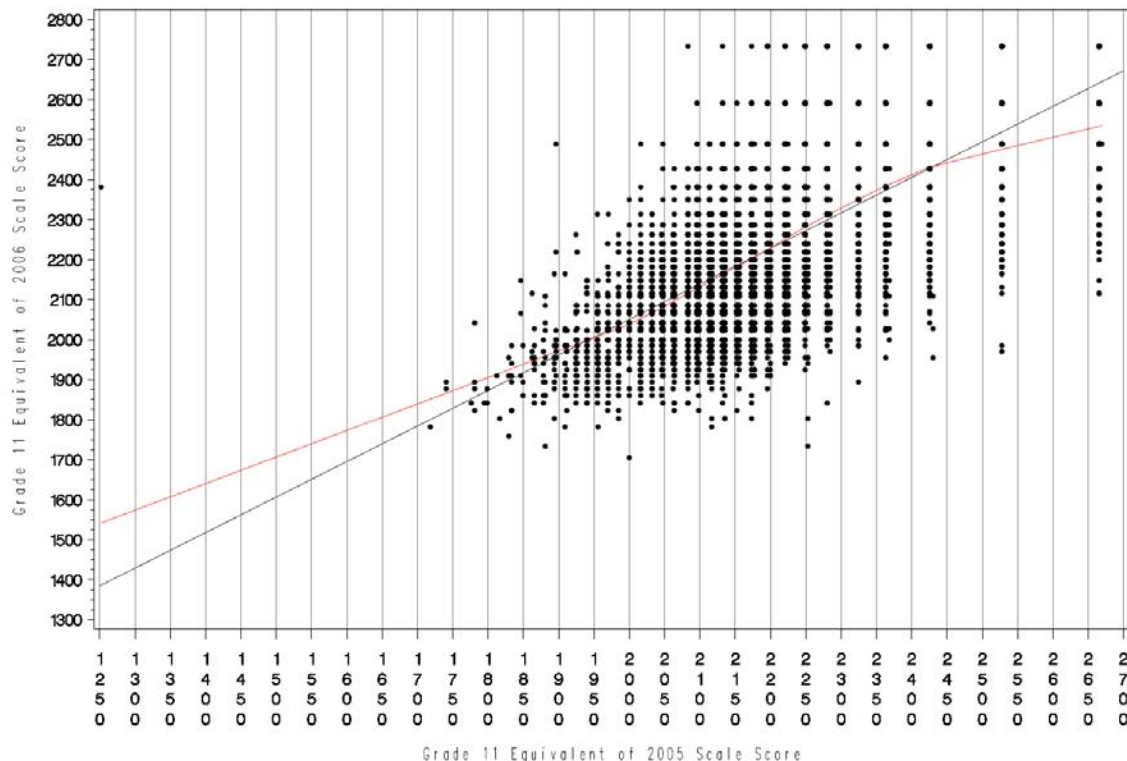
Papers covering efficiency of SAS programs have been written by Lafler, Sadof, and Virgile among many other authors. Please refer to the References section for links to these papers.

TAKS Scores

The state of Texas mandates that most students in grades 3 through 11 take the Texas Assessment of Knowledge and Skills (TAKS) each year in varying subjects. Each test consists of approximately 50 multiple choice questions. The number of questions that the student gets correct is called the raw score. The raw score is converted to a scale score based upon the performance of students state-wide.

A typical scatterplot of two scale scores representing the results for about 250,000 students is shown in Figure 1. As you can see there are a limited number of unique combinations of the two scale scores. Naively plotting all of the data resulted in scatterplots with each point representing thousands of students. Such a scatterplot takes a noticeable length of time to open on a PC as the software places thousands of points in exactly the same spot. One solution to speed up the opening of scatterplots, in this case, would be to summarize the unique combinations of the two scale scores into a dataset and make the scatterplot from the smaller dataset. Another option would be to take a random sample of the observations and plot the sample data.

Figure 1. Typical Scatterplot of Year-to-Year TAKS scores



A question of interest to education researchers is whether a student's scale score for a particular subject one year predicts their scale score in a subsequent year. These predictions can then be used to assess how schools do with their students in the aggregate. Running SAS regression procedures on the large datasets can take a very long time, so a natural question was whether the summarized dataset used for the scatterplot could be used to speed up the regression procedures. Once a new variable indicating the number of records having each unique combination of the original variables is added, more efficient processing can be achieved through the use of the FREQ statement. The numeric variable given in the FREQ statement tells SAS to consider each record as representing that many records as if you were processing the larger dataset.

Experimental Design

What efficiency gains can be achieved by summarizing and when is it a good idea to do so. As an initial investigation into these questions, a simulated dataset was created that varied in the number of records and the amount of "redundancy" in the data. One million records were created according to the following rules. First an "x" value was created as a random normal variable with mean 0.5 and standard deviation 0.15.

$$x \sim N(0.5, 0.15)$$

A “y” variable was created by multiplying the “x” value by 0.5, adding 0.25, and then adding a random normal variable with mean 0.5 and standard deviation 0.15.

$$y = 0.5x + 0.25 + \varepsilon$$

$$\varepsilon \sim N(0.5, 0.15)$$

To simulate the censoring at high scores observed in Figure 1, both x and y were constrained to be between 0 and 1. A program to create the simulated dataset can be found in Appendix A.

The primary factor of interest in this study is comparing the efficiency of regression procedures run on the raw data versus summarized data. In addition, the effect of dataset size was explored through the use of the original 1,000,000 record simulated dataset versus a systematic sample of 100,000 of those records. Finally, the level of redundancy in the data was controlled by either rounding the “x” and “y” values to the nearest tenth or hundredth. Table 1 summarizes these factors while Figure 2 displays a plot of the data with 100,000 records and values rounded to the nearest tenth.

Table 1.

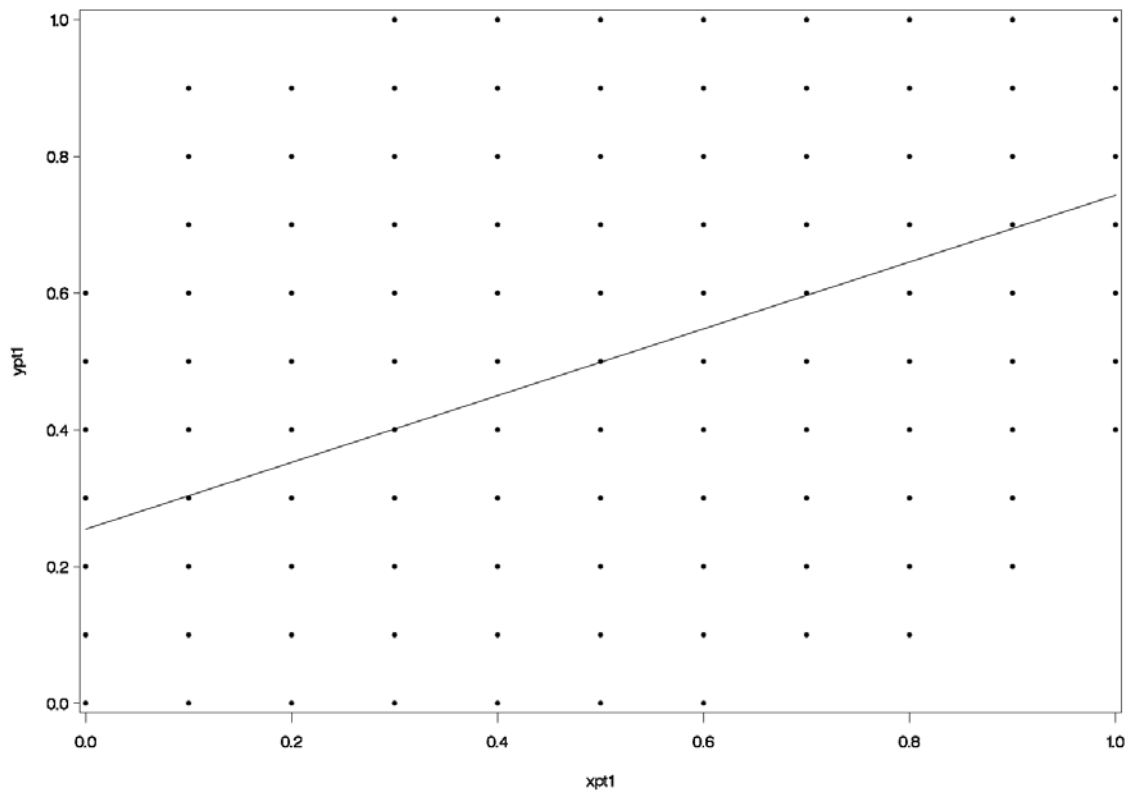
Factor	Level 1	Level 2
Data type	Raw	Summarized
Dataset size	100,000 records	1,000,000 records
Data redundancy	Rounded to the nearest tenth	Rounded to the nearest hundredth

Five replications of all eight unique combinations of the factors were run. For each replication the order of the eight combinations was randomly determined. For each run the following data was recorded.

- CPU Time needed to either bring the raw dataset into the work library or summarize it using PROC SQL.
- CPU Time needed to run PROC REG for the simple model predicting rounded “y” by rounded “x”
- CPU Time needed to create a PDF file of a scatterplot created by PROC GPLOT

Investigation of the effect of summarization on PROC LOESS was initially planned. However, this procedure ran so slowly on the full dataset that the only reasonable course would be to cut down the dataset size. Interestingly, PROC LOESS does not have a FREQ statement. There is a WEIGHT statement, but that does not do the same thing as FREQ. For PROC LOESS, the most reasonable course of action would be to randomly sample from the full dataset before running.

Figure 2. Scatterplot of the Simulated Dataset



Results

The average CPU time needed for each of the measured processes is summarized in Tables 2-4. Table 2 presents that average CPU time taken to bring the data into the SAS work directory from a permanent file. The raw data was read in by a DATA step as follows.

```
data rawData;  
    set lib.dataset ;  
run;
```

The summarized data was brought in by the following SQL procedure yielding counts of each unique combination of the predictor and response variables.

```
proc sql noprint;  
create table countData as  
    select x, y, count(*) as count  
    from lib.dataset  
    group by x, y;  
quit;
```

Table 2. Data Loading

CPU Time in Seconds		Raw	Summarized
100,000 Records			
Rounding	Nearest Tenth	0.32	0.59
	Nearest Hundredth	0.37	0.58
1,000,000 Records			
Rounding	Nearest Tenth	3.23	5.00
	Nearest Hundredth	3.52	5.71

As can be seen, the initial step of summarizing the data took about 50% to 80% longer in these trials than simply reading in the raw data. This and the extra coding time required write the SQL statement should be taken into account when deciding whether to summarize. The number of unique values in the data which is controlled by the level of rounding does not seem to affect the processing speed very much. It seems that the benefits of summarizing may be minimal if not negative when a small number of procedures are to be run on the data.

Table 3 reports the average CPU time to process the following regression request.

```
proc reg data=dataset
    outest=regLine(keep=intercept x
                    rename=(x=slope));
    %if %upcase(&dataType.) = COUNT %then %do;
        freq count;
    %end;
    model y = x ;
run;
quit;
```

Table 3. PROC REG

CPU Time in Seconds		Raw	Summarized
100,000 Records			
Rounding	Nearest Tenth	0.25	0.10
	Nearest Hundredth	0.28	0.13
1,000,000 Records			
Rounding	Nearest Tenth	1.59	0.08
	Nearest Hundredth	1.64	0.08

Efficiency gains of using summarized data for PROC REG were modest for the 100,000 record dataset but noticeable for the 1,000,000 record dataset. In fact, for the summarized data there was no practical difference in the average CPU time between the larger and smaller dataset. This is probably due to the summarized data for the larger dataset being not much larger than the summarized data for the smaller one. For example, there were 107 records in the dataset rounded to the nearest tenth summarized from 100,000 records and 114 in the one summarized from 1,000,000 records.

Table 4 presents the CPU time required to create a PDF version of a scatterplot showing the regression line. The PROC GPLOT statement used was as follows.

```

ods listing close;
ods pdf file="file.pdf";
    proc gplot data=dataset;
        plot y * x / haxis=axis1 vaxis=axis2
                anno=annoRegLine;
    run;
    quit;
ods pdf close;
ods listing;

```

Table 4. PROC GPLOT

CPU Time in Seconds		Raw	Summarized
100,000 Records			
Rounding	Nearest Tenth	10.40	0.06
	Nearest Hundredth	10.41	0.80
1,000,000 Records			
Rounding	Nearest Tenth	78.94	0.07
	Nearest Hundredth	78.57	0.95

Efficiency gains for using summarized data in PROC GPLOT were quite large particularly when coupled with the increased speed with which the plots created from the summarized data can be opened on a PC. If the data structure allows, a significant amount of time can be saved when creating high-resolution graphics by plotting summarized data instead of the raw data.

Conclusion

The effect of summarization on the efficiency of SAS regression procedures was investigated. In general summarization seems to confer a benefit when creating high-quality plots, using computationally intensive procedures, or running a large number of regression procedures.

Some factors for additional research on the effect of data summarization could be:

- More complex models involving additional predictor variables.
- Larger datasets.
- Stepwise procedures.

I hope you will be able to apply these findings in your work.

References

Lafler, Kirk Paul; Efficiency Techniques for Beginning PROC SQL Users; SUGI29 Paper 127-29; retrieved from <http://www2.sas.com/proceedings/sugi29/127-29.pdf>.

Sadof, Michael G.; Keeping Your Data In Step - Utilizing Efficiencies; SUGI24 Paper 37; retrieved from <http://www2.sas.com/proceedings/sugi24/Advtutor/p37-24.pdf>.

Virgile, Bob; The Most Important Efficiency Techniques; SAS Global Forum 2007 Paper 209-2007; retrieved from <http://www2.sas.com/proceedings/forum2007/209-2007.pdf>.

Appendix A: SAS Program to Create Simulated Data

```
/*-----*\
** Description:  Create data to test effect of summarizing data      **
**               prior to regression procedure.                      **
/*-----*/

%include "C:\SAS\SCSUG 2007\SCSUG Utilities.sas";

%macro censor(var);
    &var._cen = &var.;
    if &var. < 0 then &var._cen = 0;
    else if &var. > 1 then &var._cen = 1;
%mend censor;

%macro roundVal(unit, name);
    x&name. = round(x_cen,&unit.);
    y&name. = round(y_cen,&unit.);
%mend roundVal;

data scsug07.small(drop=i)
    scsug07.medium(drop=i);

    /* initialize random number generator so that results are
       repeatable */
    call streaminit(38140405);

    do i = 1 to 1000000;
        /* create data from true regression relation */
        x = rand("NORMAL",0.5,0.15);
        y = 0.25 + 0.5 * x + rand("NORMAL",0,0.15);

        %censor(x);
        %censor(y);

        %roundVal(0.1, pt1);
        %roundVal(0.01, pt01);

        if mod(i,10) = 0 then output scsug07.small ;
        output scsug07.medium ;
    end;
run;
```

Appendix B: SAS Program to Duplicate¹ the Experiment

```
/*-----*\
** Description:  Test effect of summarizing data prior to regression **
**              procedure.                                         **
/*-----*/

%include "C:\SAS\SCSUG 2007\SCSUG Utilities.sas";

data runs;
    /* Read in data with 8 runs covering all combinations of Data
       type, size and rounding replicated 5 times. The order of the
       runs within each replication was determined randomly */
    infile "&dirname.\runs.csv"
        delimiter = ',' MISSOVER DSD lrecl=100 firstobs=3;
    length rep $1 DataType $5 DataSize $6 rounding $5;
    /*          Raw or      Small or      0.1 or
               Count      Medium      0.01 */

    input rep DataType DataSize rounding;
run;

proc sql noprint;
    select count(*)
        into :n
        from runs;
    select rep, dataType, dataSize, rounding
        into :rep1 - :rep%left(&n.),
             :dt1 - :dt%left(&n.),
             :ds1 - :ds%left(&n.),
             :ro1 - :ro%left(&n.)
        from runs;
quit;

%annomac;
symbol1 interpol = none
    value = dot
    height = 6pt
    color=black
;
axis1 order=0 to 1 by 0.2
    minor=none
;
axis2 label=(angle=90)
    order=0 to 1 by 0.2
    minor=none
;

%macro runExp;
    %do i = 1 %to &n.;
        %put ##### ;
        %put &i.) Rep &&rep&i., Data Type &&dt&i., Data Size
            &&ds&i., Rounding &&ro&i. ;
        %put ##### ;
    %end;
%macro runExp;
```

¹ The resulting CPU times will vary depending on the system used.

```

    %if &&ro&i. = 0.1 %then %let curRo = pt1;
    %else %if &&ro&i. = 0.01 %then %let curRo = pt01;

    %if &&dt&i. = Count %then %do;
        %let curDT = count;
        proc sql noprint;
            create table countData as
                select x&curRo, y&curRo, count(*) as
                    count
                from scsug07.&&ds&i.
                group by x&curRo, y&curRo;
        quit;
    %end;
    %else %do;
        %let curDT = raw;
        data rawData;
            set scsug07.&&ds&i. ;
        run;
    %end;

    proc reg data=&curDT.Data
        outest=regLine(keep=intercept x&curRo.
                        rename=(x&curRo.=slope));
        %if %upcase(&curDT.) = COUNT %then %do;
            freq count;
        %end;
        model y&curRo. = x&curRo. ;
    run;
    quit;

    data annoRegLine;
        set regLine(in=reg);

        hsys = "2";
        xsys = "2";
        ysys = "2";

        %line(0, intercept, 1, slope+intercept, black, 1, 0.1);
    run;

    ods listing close;
    ods pdf file="&dirname.\&&rep&i. &&dt&i. &&ds&i.
        &curRo..pdf";
        title "Rep: &&rep&i. Data Type: &&dt&i. Data Size:
            &&ds&i. Rounding: &&ro&i";
    proc gplot data=&curDT.Data;
        plot y&curRo. * x&curRo.
            / haxis=axis1 vaxis=axis2
              anno=annoRegLine;
    run;
    quit;
    ods pdf close;
    ods listing;

    %end;
%mend runExp;
%runExp;

```