Paper 274-2017

Text Generational Data Sets (Text GDS)

Dr. Kannan Deivasigamani

HSBC

ABSTRACT

This paper offers a way to fill the void that *SAS*® currently has with respect to the missing feature in the language, to create generational datasets of *TEXT* files that is identical to the syntax available to create *SAS-Generational-Data-Sets* (SAS *GDS*) especially in *UNIX* & *AIX* environments.

INTRODUCTION

SAS offers generational dataset structure as part of the language feature that many users are familiar and use in their organizations and manage using keywords such as ***genmax***, ***gennum***, etc. While *SAS* operates in a mainframe environment, users also have the ability to tap into the *GDG* (Generational Data Group) feature available on *Z/OS, OS/390, OS/370, IBM 3070* or *IBM 3090* machines. With cost saving initiatives across businesses and due to some scaling factors, many organizations are in process of migrating to mid-tier platforms to cheaper operating platforms such as *UNIX & AIX*. With Linux being open source and cheaper alternative, several organizations have opted for *UNIX* distribution of *SAS* that could work in *UNIX/AIX* environments.

While this may be a viable alternative, there are certain nuances that the migration effort brings to the technical conversion teams. Unlike mainframes, on *UNIX*, the concept of *GDGs* does not exist. While SAS offers generational datasets feature as part of the language, they are only good for *SAS* datasets. If an organization needs to house and operate with a *GDG* like structure for *TEXT* datasets, there isn't one available in a *UNIX/AIX* environment. While my organization had a similar initiative to migrate programs used to run the subprime mortgage analytic, incentive and regulatory reporting to AIX, we identified the paucity of literature and research on this topic. Hence I ended up developing a *utility* that addresses this need which is henceforth referred as *TextGDS*..

TEXT GDS MACRO

TextGDS is a tool in the form of a user defined *SAS macro* that could be called by programs that need a *GDG* like structure for *TEXT* files in *AIX/UNIX* environment. The macro will require some parameters to be passed that would determine the kind of operation a user intends to perform. The operation can be better explained with the example provided below that uses the macro *TextGDS* that explained throughout this document whose code is included at the end. The macro call for this example is as shown below:

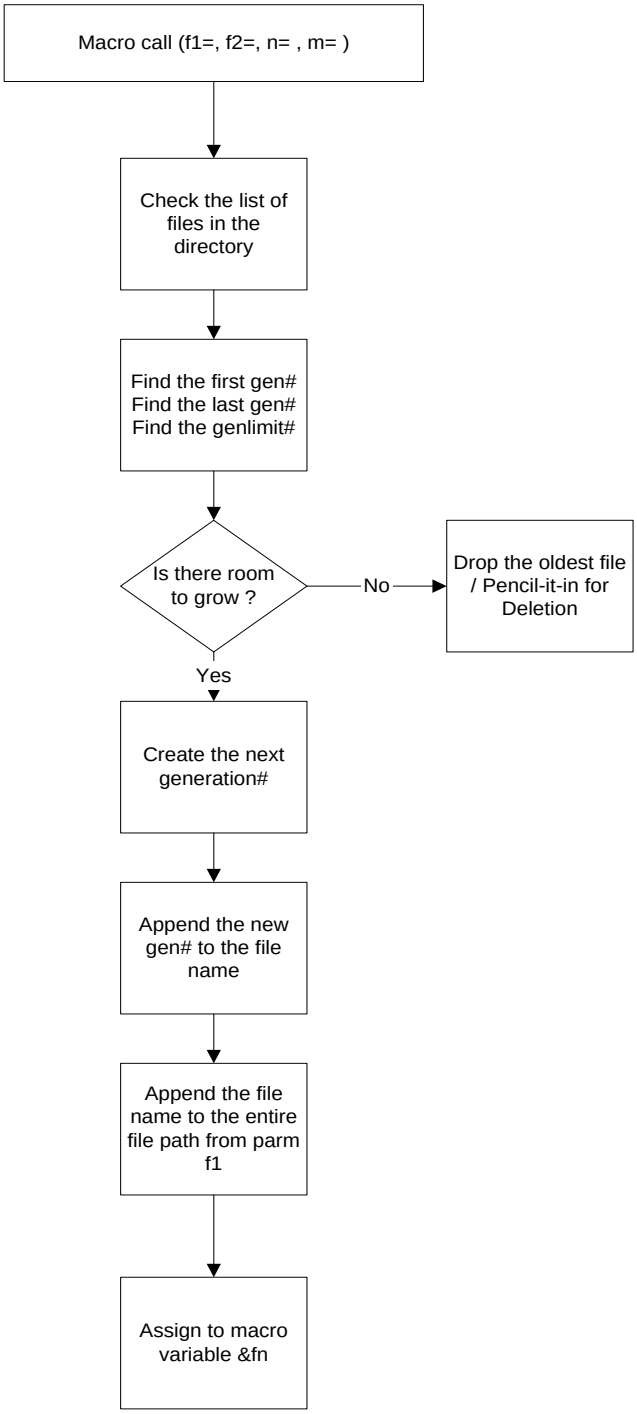
```
%TextGDS (f1=/data, f2=Test_, n=+1, m=5);
```

Such a call would result in populating a macro variable *&fn* with a value that could be assigned to the *file-ref* name in the *filename* statement. This value assigned to the macro variable *&fn* will be named in such a way that this resembles a *GDG* structure. The file can be read in a *SAS* program using the same *SAS* macro by altering the parms to fit a user's need.

An example of a read is demonstrated below. In this code below, the parameter *"n"* has a value of *"+0"* which indicates that the user is requesting the macro to point to the current generation of the text file.

```
%TextGDS(f1=/data,f2=Test_,n=+0,m=5);
Filename f1 &fn;
```

TEXT - GENERATIONAL DATA SETS (TextGDS)



Error! Not a valid bookmark self-reference.**. TextGDS Macro Operation**

MACRO CODE (SHELL)

```
%macro TextGDS(f1=,f2=,n=,m=);
      %sysexec %str(cd &f1; ls &f2* > &WORKFLDR/meta.txt);
      Filename f "&WORKFLDR/meta.txt";
      Data in;
       Infile f length=recl;
       Input @1 inrec $varying100. recl;
       newin = trim(inrec);
       dat pos = index(inrec,".dat");
       len = length(newin);
       nc = substr(newin,dat_pos-10,6);
       n  = input(nc,6.);
       mc = substr(newin,dat pos-3,3);
       qlimit = input (mc,3.);
      run;
      proc sql;
        select  count(*),min(n),max(n),qlimit
          into :calc_tot,:calc_min,:calc_max,:calc_glimit
          from in;
      quit;
      %let abs n = %sysfunc(abs(&n));
      %if &calc_tot ne 0
      %then
        %do;
          proc sql;
            select max(n) &n format=z6.
              into :newmaxn
              from in;
          quit;
          proc sql;
            select  min(n) format=z6.,glimit format=z3.
              into :dropgen,:gen
              from in;
          quit;
          %let igen  = &n+0;
          %if &igen ge 1
          %then
            %do;
              %let drop file name =
              %qsysfunc(dequote(&f1))/&f2&dropgen.m&gen..dat;
              filename dropfile "&drop_file_name";
              data  null ;
                %let calc newhi = &calc_max + &n;
                rc=fdelete("f");
                if (%sysevalf(&calc_glimit+1) le &calc_tot) and (&n gt 0)
                then
                rc=fdelete("dropfile");
              run;
            %end;
            data  null ;
             %global fn; /* ver1.3 chgs below */
             %let fntemp = %qsysfunc(dequote(&f1))/&f2&newmaxn.m&gen..dat;
             %let fn = "&fntemp";
            run;
         %end;
        %else
```

```
        %do;
          data in2;
            m=&m;
            m2=put(m,z3.);
            infile f length=recl;
            input @1 inrec $varying100. recl;
            newin = trim(inrec);
            dat_pos = index(inrec,".dat");
            len = length(newin);
            nc = substr(newin,dat_pos-10,6);
            n  = input(nc,6.);
          run;
          data  null ;
            mchar  = &m;
            glimitc=put(mchar,z3.);
            call symput('glimit',glimitc);
          run;
          data  null ;
            %global fn;
            %if &n gt 0 %then
            %do;
              %let f1=%qsysfunc(dequote(&f1));
              %let fn="&f1/&f2.000001m&glimit..dat";
            %end;
          run;
        %end;
      %if (&calc tot lt &abs n) and &n lt 1
      %then %let fn='FILE NOT FOUND ERROR!';
%mend TextGDS;
%let WORKFLDR=%sysget(BASE_WORK_PATH);
```

MACRO PARAMETERS

Parms *f1* and *f2* point to file location and file name respectively. The next parameter n can either have a positive or a negative value. It cannot have an unsigned numeric value. The last parameter m in this case holds a value of 5 which indicates that this macro is set to hold a max of 5 generations. This is equivalent to the *GENLIMIT* parm on the mainframes while defining a *GDG*. If the file in the directory is empty before running this macro and is run for the first time with parameters shown below, the first generation will be created as per the macro parameters provided. See page 2 the operational flow of the TextGDS macro.

```
%TextGDS (f1=/data, f2=Test_, n=+1, m=5);
Filename f1 &fn;
Data _null_;
    File f1;
    Put 'the TextGDS# is:' &fn;
Run;
```

The data folder will have the following file:

```
/data/Text_000001m005.dat
```

As shown above, the first part of the file name is from parm f2=Test_ and the second part is from parm n=+1 which resulted in 000001. The last part is from the parm m=5 that resulted in 005 which is appended with the file extension ".dat" that completes the file name and results in:

```
Text_000001m005.dat
```

If the code segment shown above with the macro is invoked again, then the next generation is created and the ls –altr command in the /data folder will show the following 2 files:

```
Text_000001m005.dat
Text_000002m005.dat
```

If the code segment is executed again, the 3rd generation will be created and will have 3 files as shown below:

```
Text_000001m005.dat
Text_000002m005.dat
Text_000003m005.dat
```

A repeat of the same will result in the 4th file and finally another repetition will result in 5 files as shown below:

```
Text_000001m005.dat
Text_000002m005.dat
Text_000003m005.dat
Text_000004m005.dat
Text_000005m005.dat
```

Here is the interesting part, if the code is executed again, the 6th generation will be created and most recent 5 generations will be available for use as shown below:

```
Text_000002m005.dat
Text_000003m005.dat
Text_000004m005.dat
Text_000005m005.dat
Text_000006m005.dat
```

The oldest generation #000001 will be deleted and the most recent 5 generations will be available. The oldest generations will keep dropping off as new generations are created. The macro internally manages these generation retention process using the work space mentioned earlier on in the paper. The macro reads the Meta data in the current directory which is nothing but the list of generations matching the file name provided in the macro parameter and stores it for file management operations in the macro. The maximum and minimum generations are identified and their generations numbers are handled depending on the max gen limit that the macro initially defined it to hold. If the user is requesting a +1, then the generation number is incremented to the next higher number and the oldest generation is deleted within the macro code. So far, we have seen examples about creating new TextGDS files but not reading in existing current and historical generations which is discussed in the next section.

MACRO READ OPERATION

A macro call with a negative sign to parameter “n” will result in a read operation of a historic generation as specified by the numeric value assigned to the parameter “n”.

```
%TextGDS (f1=/data, f2=Test_, n=-1, m=5);
Filename f1 &fn;
Data _null_;
    File f1;
    Put 'the TDS# is:' &fn;
Run;
```

A macro call with a value of -1 will result in reading the previous generation of the file. As an example if we have the following 5 files in the /data folder if a read is attempted with a -1 as described,

Text_000005m005.dat will be referenced by the &fn macro variable and assigned in the filename statement.

```
Text_000002m005.dat
Text_000003m005.dat
Text_000004m005.dat
Text_000005m005.dat
Text_000006m005.dat
```

Similar to this, if a -2 is the value to the parameter m, then Text_000004m005.dat will be the file name that the &fn macro variable will be pointing as a result. If the macro is referencing a generation that is out of scope, then as one would expect, the code would not be successful in pointing to the generation.

The macro is intelligent to recognize what the user is requesting and perform the operation in the UNIX region using a temporary work space for internal calculation and populates the final output variable which is nothing but a path followed by the file name that includes the fixed name along with the variable generation number followed by an extension indicating the generation limit that the file structure was defined for.

LIMITATIONS AND DELIMITATIONS

There are certain limitations to the current version of the macro. As one can notice the maximum value that can be held by the 3 digit gen-limit value is 999 which is identical to a SAS GDS. In addition, the highest generation number of the TextGDS indicated by the 6-digit value that follows the TGDS name is 999,999 which is about a million generations. As one might question, while the generations may recycle but the current version of the macro is not capable of recognizing the smaller number after recycling beyond 1,000,000. However, this is a potential enhancement that could be implemented in future or if SAS takes up this idea to incorporate as part of the software enhancement, it would be worthwhile to users. Another alternative is that, upon reaching the limit, the folder can be archived and the new set of million generations can begin, as an alternative starting from 1.

Similar to this, the number of generations can also be adjusted by shrinking or expanding the 6 digits to fit one's needs. The maximum number of generations may also be shrunk or expanded according to one's needs. The extension of the file which is ".dat" may also be modified to ".txt" or another type depending on the need of a project. On the other hand, all these 3 limitations discussed may also be converted to parms that one may input to the macro to accept the values dynamically as the macro is invoked. However, all these enhancements call for a code change in the macro and the way the macro will be invoked.

Another limitation or requirement is, this macro requires a temporary work storage for the computation where the "meta" file is written and cleared out at the end of the macro execution and will be left with no trace. The workspace or the temporary folder is a requirement in the current design. One might choose to have this provided in the parm if desired; however, in the current setup, the invoking job will have a subfolder within the work folder and will be used for the computation. The user, depending on the setup might choose to add a command "mkdir tmp" in place of the work folder as one deems appropriate.

The macro does not prevent anyone from invoking to point to a "+5" or "> +1 " without warning the user of getting ahead of the gen numbers. Therefore, the macro needs to be used with caution and it will be the responsibility of the user to invoke with appropriate parameters.

CONCLUSION

This tool can be enhanced to make it more robust but is a good start and can be utilized by developers. The TextGDS feature, someday hopefully becomes available as part of native SAS language for developers to use it for maintaining generational text files when organizations need them.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *SAS® For Dummies®*
- *Linux/Unix literature*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dr. Kannan Deivasigamani
textdrk@gmail.com