VIRTUAL

SAS® GLOBAL FORUM 2021

#SASGF

Using the %SCAN Function

Jonas V. Bilenas

I have been using SAS ®, primarily in major banks for 30+ years, developing and implementing various statistical models including; regulatory models, credit approval models, segmentation, neural networks, customer behavioral, time series models, and Optimized Response Surface experimental designs to name a few.

OUTLINE

- How to use the %SCAN Macro Function.
- Applications:
  1. How to Uppercase all SAS® DATA VARIABLE NAMES Automatically?
  2. A look at SGSCATTER with a large number of plots to view. Can we split the number of plots into a workable number of SGSCATTER plots?

The %SCAN function:

- %SCAN pulls out parts of a macro variable that are delimited by special characters.
- From SAS Support documentation, the default delimiter can be:
  - For ASCII computer characters:
    - blank ! $ % & ( ) * + , - . / ; < ^¦
  - For EBCDIC:
    - blank ! $ % & ( ) * + , - . / ; < ¬ | ¢¦

The %SCAN function:

- %SCAN function pulls out parts of a macro variable that are delimited by special characters.

- The %SCAN function has 2-4 arguments:
  - First argument is the macro variable that you will parse.
  - Second argument is the part number. 1=first part, 2= second, etc..
  - The 3rd argument defines the delimiter that separates the parts of the macro variable. The examples used in this presentation have the delimiter as a space.
  - The 4th argument is optional. It is a "modifier" that specifies special delimiter codes to handle more specific needs. **May not be available in early versions of SAS 9.4**

Application Example 1

Application Example 1

- A few years ago I received an email from a colleague asking if there was an automatic way or a MACRO that can capitalize all SAS Variable names without having to specify each variable in the program code with a RENAME statement.

- It seemed a bit strange but made sense if you have many variables in the data set with mixed cases.
  - PROC CONTENS uppercase variable names are sorted before lowercase variable names.

Application Example 1

Let’s start with a test data set to see if we can get this request to work.

```
DATA test;
  AAA2 = 5; bbb4 = 6; CC4 = 7; ddd_2 = 8; xYz44 = 'BB';
  OUTPUT;
RUN;
```

Application Example 1

The MACRO starts here:

```
%MACRO UPCASEV(lib, ds);
PROC CONTENTS DATA=&lib..&ds.  OUT=cnts NOPRINT;
RUN;
```

- **Why are we creating an output data set, cnts, from PROC CONTENTS?**
- **How many observations does data test have?**
- **How many observation does data cnts have?**
- **How many variables in data set cnts?**
- **What does NOPRINT do?**

Application Example 1

- A section of PROC CONTENTS output for test data:

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	AAA2	Num	8
3	CC4	Num	8
2	bbb4	Num	8
4	ddd_2	Num	8
5	xYz44	Char	2

Application Example 1

- TEST data has 1 observation 5 variable
- CNTS data has 5 observations and 41 variables, depending on SAS version.

- Variables in CNTS data:
  See Appendix 1 for a list of the 41 variables.

- Some of the variables and values:

Obs	NAME	TYPE	NOBS
1	AAA2	1	1
2	CCC4	1	1
3	bbb4	1	1
4	ddd_2	1	1
5	xYz	2	1

- TYPE: 1=numeric 2=character

Application Example 1

- Let's create the macro variable that will be parsed by the %SCAN function.

```
PROC SQL NOPRINT;
  SELECT name INTO: vars SEPARATED BY ' '
         FROM cnts;
QUIT;
%PUT VARS=&vars.;
```

```
LOG:
SYMBOLGEN:  Macro variable VARS resolves to AAA2 CC4 bbb4 ddd_2 xYz44
127         %put VARS=&vars.;
VARS=AAA2 CC4 bbb4 ddd_2 xYz44
```

NOTE: Maximum length of a MACRO variable is 64Kb =64*2E10 = 65,536

Application Example 1

```
%LET I = 1; /* Initialize &I. */

/* use PROC DATASETS to modify the metadata of the data set * /
PROC DATASETS LIB=&lib. nolist; MODIFY &ds.;
  %do %until(%scan(&vars,&I.,%str( ) ) = %str( ) );
    %let var=%scan(&vars,&I.,%str( ) );
    %let ren=%upcase(&var.); /* &REN=Upcase of &var */
    %if &var. ne &ren %then %do; /* if &VAR not equal to &REN*/
      rename &var. = &ren.; /* Use RENAME STATEMENT in PROC DATASETS */
    %end; /* END OF THE 2nd %do LOOP */
    %let I = %eval(&I. + 1); /* Increment &I by 1 */
  %end; /* End First Loop */
run;quit;

PROC CONTENTS DATA=&ds.;
RUN;
%MEND; /* END MACRO */
% UPCASEV(lib=WORK, ds=TEST);
```

Application Example 1 Result:

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	AAA2	Num	8
2	BBB4	Num	8
3	CC4	Num	8
4	DDD_2	Num	8
5	XYZ44	Char	2

- **This code did not work since some variables are already capitalized. For example, test data, AAA2 is already capitalized and DATASETS returned an error.**

```
%let I = 1; /* Initialize &I. */
proc datasets lib=work nolist; /* DATASETS to modify the data*/
  modify test;
  %do %until(%scan(&vars,&I.,%str( ) ) = %str( ) );
    %let var=%scan(&vars,&I.,%str( ) );
    rename &var. = %upcase(&var.);
    %let I = %eval(&I. + 1); /* Increment &I by 1 */
  %end; /* End First Loop */
run;quit;
proc contents data=test;
run;
%mend; /* END MACRO */
%rename;
```

Application Example 2

Application Example 2

- You are building a Regression Model. You have a good number of independent variables that you want to see how they predict the dependent variable using SGSCATTER, one independent variable at a time.

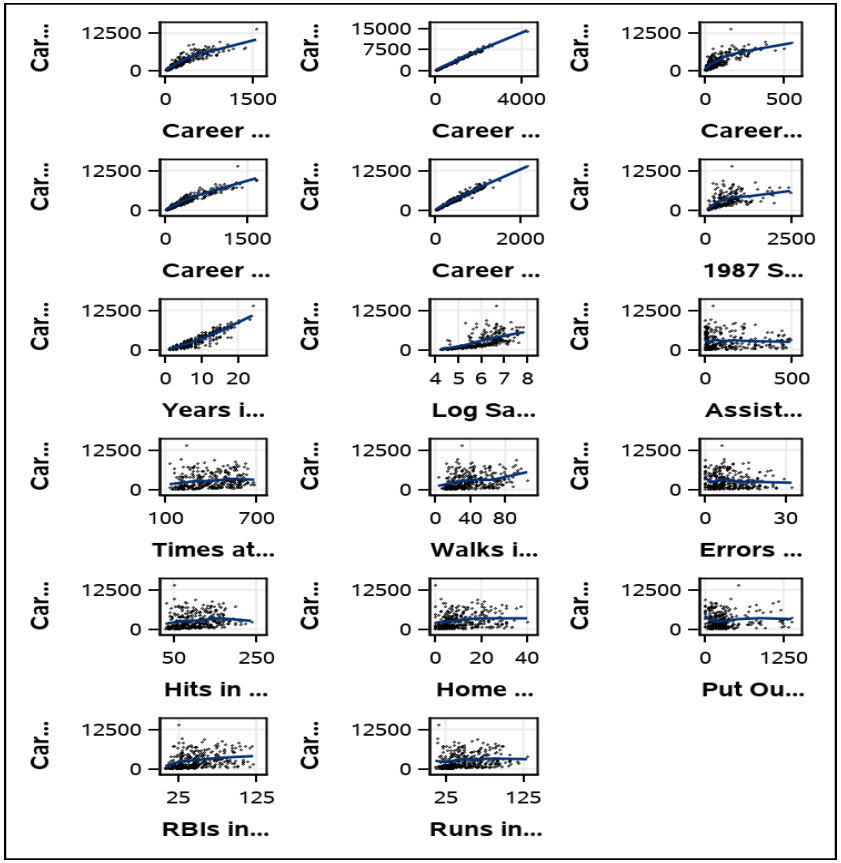
- You want to check for linearity. If not linear, you can create linear splines or natural splines and still use Linear Regression.

- You don’t want to see all the plots in a single run of SGSCATTER. You want to break it out into, say 2 rows and 3 columns of plots.

Application Example 2. First Try.

```
%LET lib=sashelp; %let ds=baseball;
%let dv=CrAtBat;  /* Career Times at Bat */

PROC CONTENTS DATA=&lib..&ds. NOPRINT OUT=cnts; RUN;

PROC SQL NOPRINT;
  SELECT name
    INTO: vars SEPARATED BY ' '
      FROM cnts
        WHERE type=1 AND NAME NOT EQ "&dv."
  ;
QUIT;

PROC SGSCATTER DATA=&lib..&ds.;
     PLOT  &DV.*(&vars.)
     / MARKERATTRS=(SIZE=2 COLOR=black) GRID
      LOESS=(SMOOTH=0.5)
      ROWS=2 COLUMNS =3;
run;
```

Application Example 2. OOPS?



Application Example 2. Second Try.

```
%LET nplots=4; %LET rows=2;
%LET lib=sashelp; %LET ds=baseball;
%LET dv=CrAtBat;

PROC CONTENTS DATA=&lib..&ds. NOPRINT OUT=cnts; RUN;

PROC SQL NOPRINT;
  SELECT name
    INTO: vars SEPARATED BY ' '
      FROM cnts
        WHERE type=1 AND NAME NOT EQ "&dv."
  ;
QUIT;
```

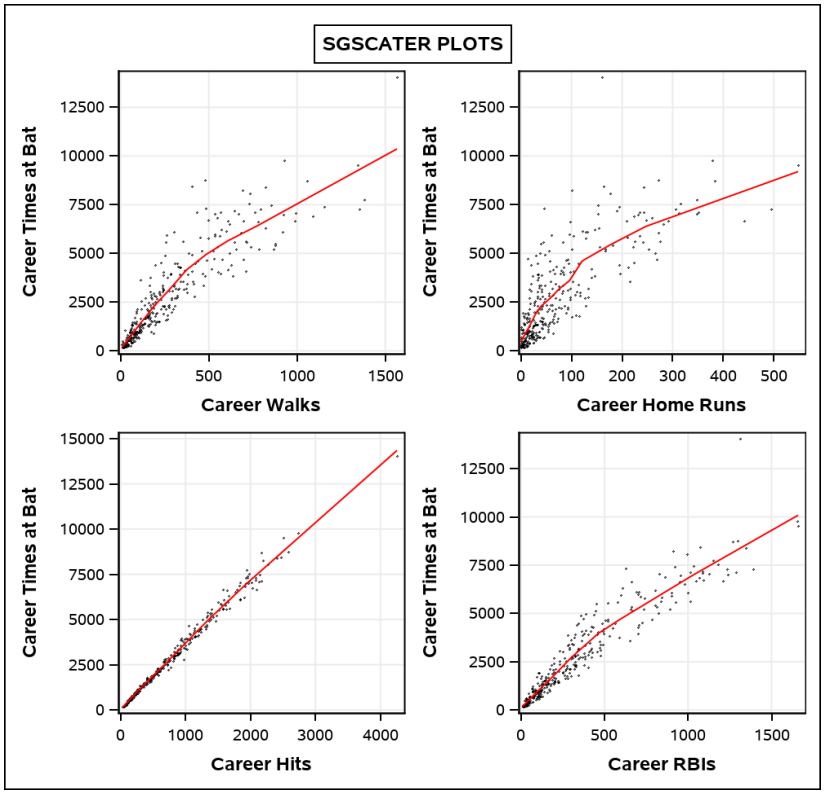
Application Example 2. Second Try Continued.

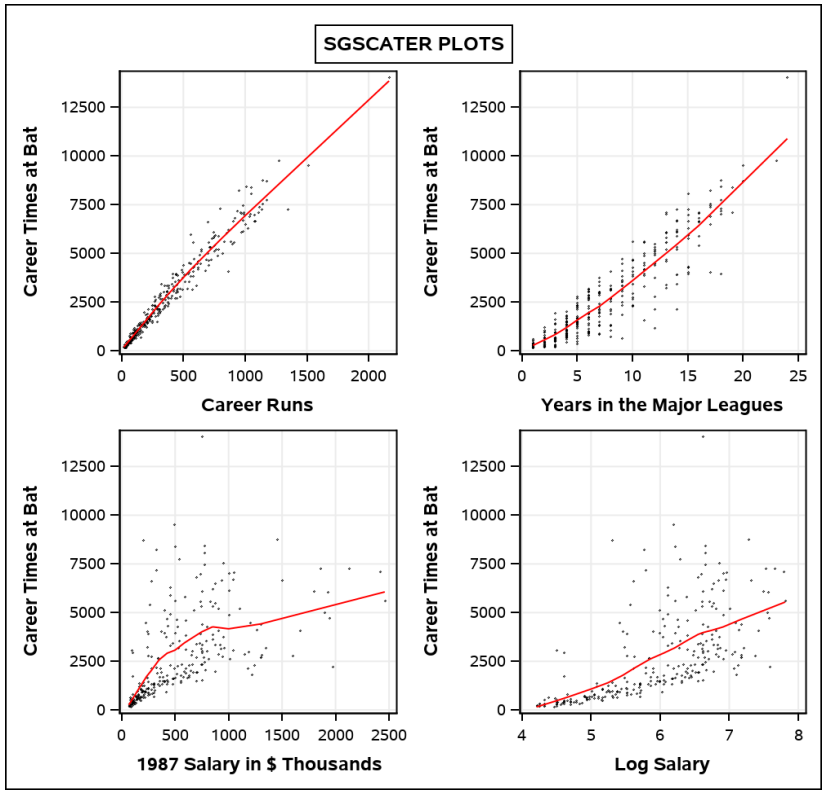
```
%MACRO plotit;
%LET I = 1; /* Initialize &I. */

%DO %UNTIL(%SCAN(&vars,&I.,%STR( ) ) = %STR( ) );
  %LET VAR=%SCAN(&vars,&I.,%STR( ) );
  %do plt_stream= 1 %to &nplots;
    %let plt_&plt_stream. = %scan(&vars,&I.,%str( ) );
    %let I = %eval(&I. + 1);
  %end;
```

Application Example 2. Second Try Continued.

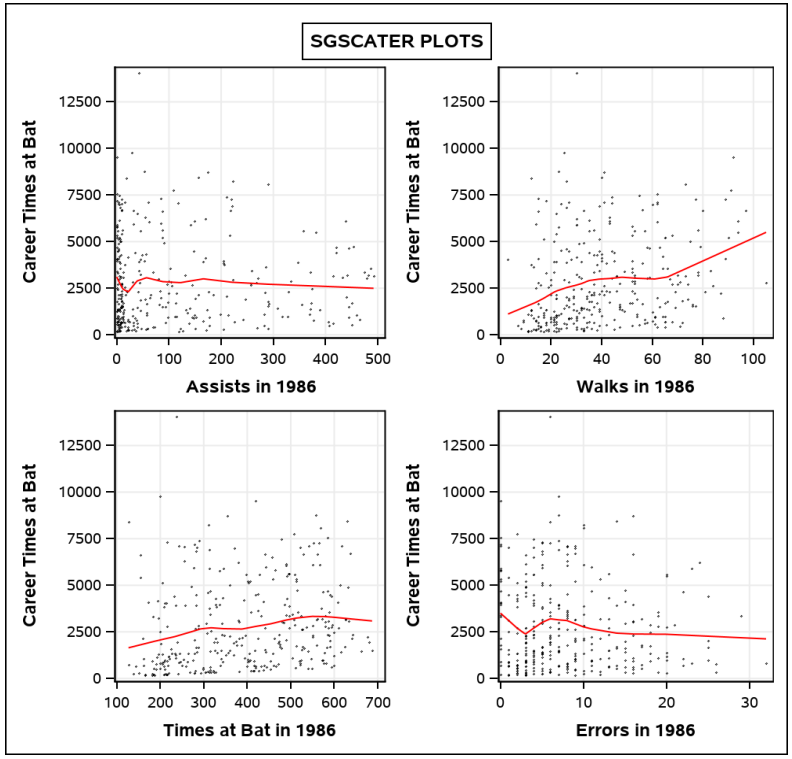
```
  proc sgscatter data=&lib..&ds.;
  PLOT  &dv. * (
  %do plt_stream= 1 %to &nplots;
        &&plt_&plt_stream.
  %end; )
     /  markerattrs=(size=2 color=black) GRID
        LOESS=(smooth=0.5
        lineattrs=(color=red thickness=.5))
        rows=&rows.
        ;
        title bold box=1 "SGSCATER PLOTS";
 run;
     %*****let I = %eval(&I. + 1); /* NOT REQUITRD here in this code */
%end;

%mend; /* END MACRO */
%plotit;
```

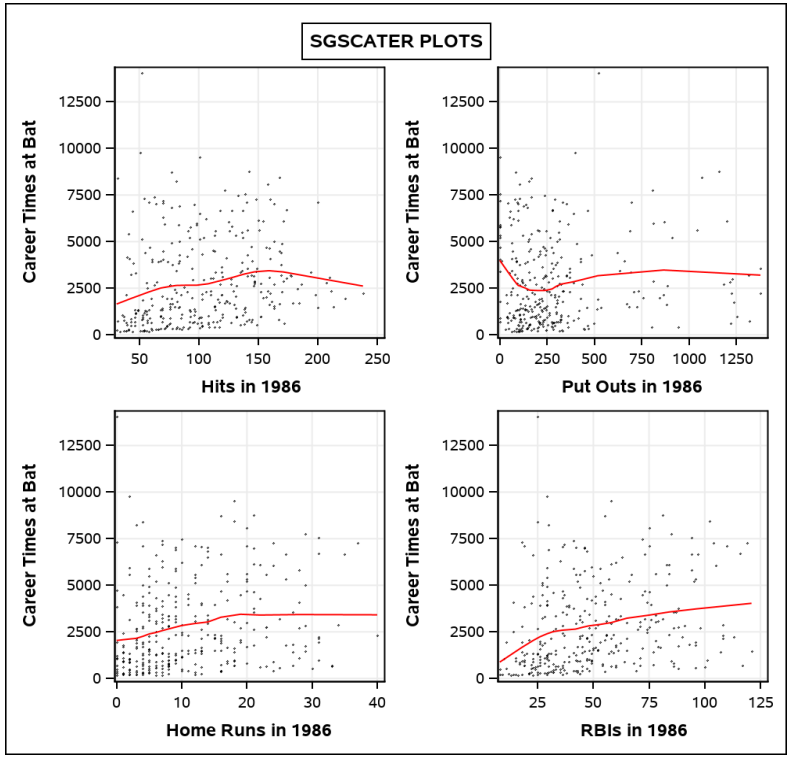
Second Try Results



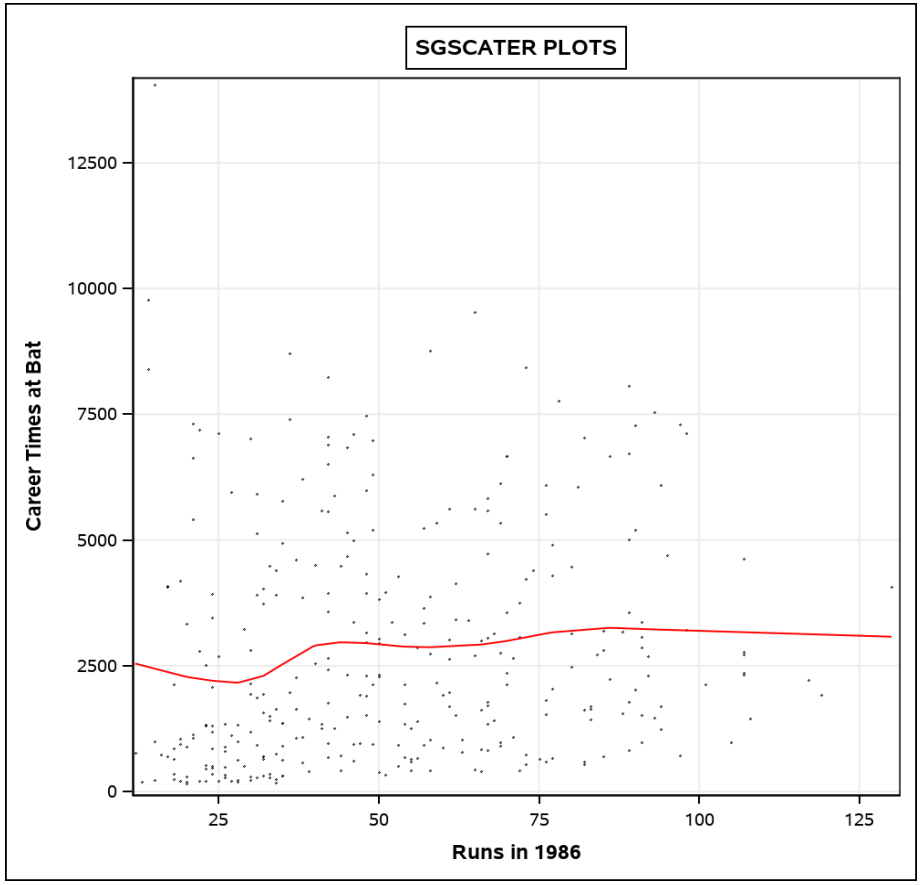


Second Try Results





Second Try Results



Appendix 1: variable list of CONTENTS OUT= DATA

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GENNEXT	Num	8		Next Generation Number
39	GENNUM	Num	8		Generation Number
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
13	INFORMAT	Char	32		Variable Informat
15	INFORMD	Num	8		Number of Informat Decimals
14	INFORML	Num	8		Informat Length
16	JUST	Num	8		Justification
9	LABEL	Char	256		Variable Label
7	LENGTH	Num	8		Variable Length
1	LIBNAME	Char	8		Library Name
3	MEMLABEL	Char	256		Data Set Label
2	MEMNAME	Char	32		Library Member Name
24	MEMTYPE	Char	8		Library Member Type
21	MODATE	Num	8	DATETIME16.	Last Modified Date
5	NAME	Char	32		Variable Name
18	NOBS	Num	8		Observations in Data Set
34	NODUPKEY	Char	3		Sort Option: No Duplicate Keys
35	NODUPREC	Char	3		Sort Option: No Duplicate Records
17	NPOS	Num	8		Position in Buffer
37	POINTOBS	Char	3		Point to Observations
26	PROTECT	Char	3		Password Protection (Read Write Alter)
29	REUSE	Char	3		Reuse Space
30	SORTED	Num	8		Sorted and/or Validated
31	SORTEDBY	Num	8		Position of Variable in Sortedby Clause
41	TRANSCOD	Char	3		Character Variables Transcoded
6	TYPE	Num	8		Variable Type
4	TYPEMEM	Char	8		Special Data Set Type (From TYPE=)
8	VARNUM	Num	8		Variable Number

Disclaimers

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 used in this presentation are trademarks of their respective companies.

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of any company that we have worked for or are currently working for.

There is no warranty on the code illustrated in this presentation.

A Few References

- Arthur L. Carpenter (2005). Storing and Using a List of Values in a Macro Variable. SAS USERS GROUP INTERNATIONAL (SUGI) 30. https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/028-30.pdf
- Carpenter, Arthur L. (2016). Carpenter's Complete Guide to the SAS® Macro Language, Third Edition, SAS Institute.
- Arthur L. Carpenter (2017). Building Intelligent Macros: Using Metadata Functions with the SAS® Macro Language. https://support.sas.com/resources/papers/proceedings17/0835-2017.pdf
- Cleveland, W. S., Devlin, S. J., and Grosse, E. (1988), "Regression by Local Fitting," Journal of Econometrics, 37, 87–114.
- Cleveland, W. S. and Grosse, E. (1991), "Computational Methods for Local Regression," Statistics and Computing, 1, 47–62.
- Lafler, Kirk Paul (2019). PROC SQL: Beyond the Basic Using SAS, Third Edition, SAS Institute
- Lora D. Delwiche, Susan J. Slaughter (2012). Graphing Made Easy with SG Procedures. SAS Global Forum 2012. https://support.sas.com/resources/papers/proceedings12/259-2012.pdf
- Xiangxiang Meng (2010). Creating High-Quality Scatter Plots: An Old Story Told by the New SGSCATTER PROCEDURE, SAS Global Forum 2010. https://support.sas.com/resources/papers/proceedings10/057-2010.pdf
- Michael A. Raithel (2010). PROC DATASETS; The Swiss Army Knife of SAS® Procedures. SAS Global Forum 2010. https://support.sas.com/resources/papers/proceedings10/138-2010.pdf

Thank you!

Contact Information
jonas@jonasbilenas.com

VIRTUAL

SAS® GLOBAL FORUM 2021

#SASGF