Getting Data Into SAS® : INFILE and INPUT

Prepared for the wonderful folks at the Boston Area SAS Users Group

Andrew T. Kuligowski
independent consultant

Getting Data Into SAS® : INFILE and INPUT

Prepared for the wonderful folks at the Boston Area SAS Users Group

(We can respectfully disagree on
Bruins vs. Lightning
Red Sox vs. Rays
Patriots vs. Buccaneers
Celtics vs. … ok, you got us there)

Using INFILE and INPUT …

Introduction

Introduction

Basic INFILE / INPUT

Conditional INPUT

Column and Line Pointers

Informats

Using INFILE and INPUT …

Introduction

Variable Length Files

Comma Separated Value Files

Dynamic Data Exchange

HTML

INPUT Function

Conclusion

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Francisco  CA
PHARMASUG 2006 Bonita Springs FL
<… Some DATALINES removed to fit example on screen …>
MWSUG      2006 Dearborn       MI
PNWSUG     2006 Seaside        OR
SUGI       2007 Orlando        FL
;
```

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
   INFILE DATALINES;
   INPUT ConfName
         ConfYear
         ConfCity
         ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

INFILE

Define the source of the external data to the DATA step. Usually followed by a file name or a *file reference* (a "nickname" for an external source).

May also include optional parameters describing the data source.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

INPUT

Define the format of the data to be processed.

This is the command which actually causes the data to be transferred from the external source into the DATA step.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
   INFILE DATALINES;
   INPUT ConfName
         ConfYear
         ConfCity
         ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

List Input

Input statement contains the name of each variable to be read in.

Fields separated on external file by one or more blanks (or other delimiter).

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
   INFILE DATALINES;
   INPUT ConfName
         ConfYear
         ConfCity
         ConfST   ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

DATALINES

A special *file reference* that tells SAS there will be instream data following the conclusion of the DATA Step.

Terminated with a ; (semicolon)

Also known as CARDS

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES4;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES4;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;;;;
```

(A quick aside …)

DATALINES4 can be used when the instream data could contain a semicolon in the first position.

Terminated with ;;;; (4 consect. semicolons)

CARDS4 also works.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San
PHARMASUG  2006 Boni
<… Some DATALINES om …>
MWS                ear
PNWSUG     200  Seas
SUGI       2007 Orla
;
```

```
SUGI       2006 San Francisco  CA
PHARMASUG  2006 Bonita Springs FL
NESUG      2006 Philadelphia   PA
WUSS       2006 Irvine         CA
SESUG      2006 Atlanta        GA
SCSUG      2006 Irving         TX
MWSUG      2006 Dearborn       MI
PNWSUG     2006 Seaside        OR
SUGI       2007 Orlando        FL
```

<This is the entire instream dataset.>

Using INFILE and INPUT …

Basic INFILE / INPUT

```
NOTE: Invalid data for ConfName in line 9 1-9.
NOTE: Invalid data for ConfCity in line 9 16-18.
NOTE: Invalid data for ConfST in line 9 20-28.
RULE: ----+----1----+----2----+----3---
9     SUGI       2006 San Francisco  CA
ConfName=. ConfYear=2006 ConfCity=. ConfST=.
_ERROR_=1 _N_=1
NOTE: Invalid data for ConfName in line 10 1-9.
NOTE: Invalid data for ConfCity in line 9 16-18.
NOTE: Invalid data for ConfST in line 9 20-28.
10    PHARMASUG 2006 Bonita Springs FL
ConfName=. ConfYear=2006 ConfCity=. ConfST=.
_ERROR_=1 _N_=2
```

<… messages repeated for each line of input data …>

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName $
        ConfYear
        ConfCity $
        ConfST   $ ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

Modified List Input

Input statement contains the name of each variable to be read in PLUS format modifiers that specify some additional piece of information about fields to be processed.

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName $
        ConfYear
        ConfCity $
        ConfST   $ ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

$ Format Modifier

Additional information about field to be read – Field is to be treated as character rather than numeric.

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San	Francisc
2	PHARMASU	2006	Bonita	Springs
3	NESUG	2006	Philadel	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
         ConfCity $ 14.;
  INFILE DATALINES;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

& Format Modifier

Additional information about field to be read – Character field can have single embedded blanks.

LENGTH

In this example, override default length of 8 for character variable.

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

```
NOTE: SAS went to a new line when INPUT
      statement reached past the end of a line.
NOTE: The data set WORK.SASCONF has 8
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	ConfCity	Conf Year	Conf ST
1	SUGI	San Francisco	2006	CA
2	PHARMASUG	Bonita Springs	2006	NESUG
3	WUSS	Irvine	2006	CA
4	SESUG	Atlanta	2006	GA
5	SCSUG	Irving	2006	TX
6	MWSUG	Dearborn	2006	MI
7	PNWSUG	Seaside	2006	OR
8	SUGI	Orlando	2007	FL

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

What happened?

Behind the scenes, reading the 2nd line …

```
INPUT ConfName    $ PHARMASUG    OK
      ConfYear   2006   OK
      ConfCity & $ Bonita Springs FL
```

Read “Bonita” + blank + “Springs” + blank + “FL”.
End of line, skip to next line.

```
      ConfST      $ ;   NESUG
```

(Ignore rest of this input line.)

So what happened to the “FL” in our output?

```
LENGTH ConfName $ 9.   ConfCity $ 14.;
```

Bonita Springs FL (“ FL” in position 15-17)

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
       ConfCity $ 17.;
  INFILE DATALINES
         TRUNCOVER ;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
  IF  ConfST = " "  TH
    ConfST = SUBSTR(Co
    ConfCity = SUBSTR(
  END;
DATALINES;
<… All DATALINES removed …>
;
```

TRUNCOVER

INFILE option. Prevents SAS from moving to the next input line if End of Line encountered in middle of an INPUT.

Default is FLOWOVER.

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
       ConfCity $ 17.;
  INFILE DATALINES
         TRUNCOVER ;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
  IF  ConfST = " "  TH
    ConfST = SUBSTR(Co
    ConfCity = SUBSTR(
  END;
DATALINES;
<… All DATALINES removed …>
;
```

(A quick aside …)

MISSOVER would also work. The difference – when INPUT reaches end of line in the middle of a variable, MISSOVER sets that value to missing, while TRUNCOVER keeps the partial value.

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
         ConfCity $ 17.;
  INFILE DATALINES
          TRUNCOVER ;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
  IF  ConfST = " "  THEN  DO;
    ConfST = SUBSTR(ConfCity,16);
    ConfCity = SUBSTR(ConfCity,1,14);
  END;
DATALINES;
<… All DATALINES removed to fit example on screen …>
;
```

(Additional coding logic added to deal with special case of field reaching maximum length.)

Using INFILE and INPUT …

4th Iteration & final

Basic INFILE / INPUT

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMPLE */
FILENAME SASCONF 'USERID.SASCONF.DATA';
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType $CHAR1. @ ;
  IF  RecordType = 'R'  THEN DO;
     INPUT @  2 ConfName $CHAR9.
           @ 11 ConfYear      4.
           @ 15 ConfCity $CHAR14.
           @ 29 ConfST   $CHAR2. ;
     OUTPUT;
  END;
  ELSE  INPUT;  /* optional */
RUN;
```

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EX
FILENAME SASCONF 'U
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordT
  IF  RecordType =
     INPUT @  2 Con
           @ 11 Con
           @ 15 Con
           @ 29 Con
     OUTPUT;
  END;
  ELSE  INPUT;  /*
RUN;
```

```
        'USERID.SASCONF.DATA'
ISUGI      2006San Francisco CA
SPHARMASUG2006Bonita SpringsFL
RNESUG     2006Philadelphia  PA
RWUSS      2006Irvine        CA
RSESUG     2006Atlanta       GA
RSCSUG     2006Irving        TX
RMWSUG     2006Dearborn      MI
RPNWSUG    2006Seaside       OR
ISUGI      2007Orlando       FL
```

This is the entire dataset. It is basically the same data that we used in our previous example, except:

- A record type has been added.
- Blank Padding has been removed.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF 'USER
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  2 ConfNa
           @ 11 ConfYe
           @ 15 ConfCi
           @ 29 ConfSt
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

FILENAME

Assign a *file reference* to an external data source (which is an MVS dataset in this example).

This file ref. will be used by the INFILE statement.

This statement occurs outside of the DATA step.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF 'USER
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  2 ConfNa
           @ 11 ConfYe
           @ 15 ConfCi
           @ 29 ConfSt
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

Formatted Input

Formats and column pointers work together to perform the task of reading data based on a specified list of variables.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF 'USER
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  2 ConfNa
           @ 11 ConfYe
           @ 15 ConfCi
           @ 29 ConfSt
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

@ "At sign" Column Pointer Control

Move the column pointer to an absolute position on the input dataset.

Continue reading data from that point.

Using INFILE and INPUT …

Conditional INPUT

Formats

$CHAR*nn*. Read character value, *nn* positions long, keeping leading blanks.

(**$*nn*.** would read character value, *nn* positions long, dropping leading blanks.)

***nn*.** Read numeric value, *nn* positions long, store in numeric SAS variable.

```
YPE" EXAMPLE */
SASCONF 'USERID.SASCONF.D
Conf;
SASCONF ;
@ 1 RecordType $CHAR1. @;
cordType = 'R'  THEN DO;
UT @  2 ConfName $CHAR9.
@ 11 ConfYear      4.
@ 15 ConfCity $CHAR14.
@ 29 ConfST   $CHAR2. ;
PUT;

INPUT;  /* optional */
```

Using INFILE and INPUT …

Conditional INPUT

@ “Trailing At sign”

Hold the line pointer on the current input line. Do not advance to the next line unless triggered to do so by another INPUT statement or by the RUN statement.

```
YPE” EXAMPLE */
SASCONF ‘USERID.SASCONF.D
Conf;
SASCONF ;
@ 1 RecordType $CHAR1. @;
cordType = ‘R’  THEN DO;
UT @  2 ConfName $CHAR9.
@ 11 ConfYear      4.
@ 15 ConfCity $CHAR14.
@ 29 ConfST   $CHAR2. ;
PUT;

INPUT;  /* optional */
```

Using INFILE and INPUT …

Conditional INPUT

(A quick aside)

@@ "Double Trailing At sign"

Hold the line pointer on the current input line. Do not advance to the next line unless triggered to do so by another INPUT statement.~~or by the RUN statement.~~

```
YPE" EXAMPLE */
SASCONF 'USERID.SASCONF.D
Conf;
SASCONF ;
@ 1 RecordType $CHAR1. @;
cordType = 'R'  THEN DO;
UT @  2 ConfName $CHAR9.
@ 11 ConfYear      4.
@ 15 ConfCity $CHAR14.
@ 29 ConfST   $CHAR2. ;
PUT;

INPUT;  /* optional */
```

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF 'USER
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  2 ConfNa
           @ 11 ConfYe
           @ 15 ConfCi
           @ 29 ConfSt
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

Null Input

Processes no data, but advances the internal line pointer to the next line (releasing any hold on that line, if applicable – as in this example).

Using INFILE and INPUT …

Conditional INPUT

```
NOTE: The infile SASCONF is:
      File Name=USERID.SASCONF.DATA,
      Lrecl=80,Recfm=FB,Blksize=960
NOTE: 9 records were read from the infile SASCONF.
NOTE: The data set WORK.SASCONF has
      6 observations and 5 variables.
```

< Results of a PROC PRINT >

Obs	Record Type	Conf Name	Conf Year	ConfCity	Conf ST
1	R	NESUG	2006	Philadelphia	PA
2	R	WUSS	2006	Irvine	CA
3	R	SESUG	2006	Atlanta	GA
4	R	SCSUG	2006	Irving	TX
5	R	MWSUG	2006	Dearborn	MI
6	R	PNWSUG	2006	Seaside	OR

Using INFILE and INPUT …

Column and Line Pointers

```
/* "Pointers" EXAMPLE */
FILENAME SASCONF2 'USERID.SASCONF2.DATA';
DATA  UGLYINPUT;
  LENGTH  ConfName $ 9. ;
  Pos1 =  9;
  Pos2 =  3;
  Str1 = "LOC=";
  INFILE SASCONF2 ;
  INPUT #1 ConfCity   $ 22-35 +1
           ConfST     $       +(-39)
           RecordType $       +Pos1
           ConfYear           +(-1*(Pos1+6))
           ConfName   $
```

Using INFILE and INPUT …

Column and Line Pointers

```
    / @ "20"          Year2Digit2
      @    1          RecordType2 $
      @ Pos2          ConfName2   $  9.
      @ Str1          ConfCity2   $ 14.
      @ (Pos2*12+1) ConfST2       $ ;
  ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

Using INFILE and INPUT …

Column and Line Pointers

```
/* "Pointers
FILENAME SAS
DATA  UGLYIN
  LENGTH  Co
  Pos1 =  9;
  Pos2 =  3;
  Str1 = "LO
  INFILE SAS
  INPUT #1 C
           C
           R
           C
           C
```

<continued …>

'USERID.SASCONF2.DATA'

```
I SUGI       2006 LOC=San Francisco  CA
S PHARMASUG  2006 LOC=Bonita Springs FL
R NESUG      2006 LOC=Philadelphia   PA
R WUSS       2006 LOC=Irvine         CA
R SESUG      2006 LOC=Atlanta        GA
R SCSUG      2006 LOC=Irving         TX
R MWSUG      2006 LOC=Dearborn       MI
R PNWSUG     2006 LOC=Seaside        OR
I SUGI       2007 LOC=Orlando        FL
R SESUG      2007 LOC=Hilton Head    SC
```

This is the entire dataset. It is basically the same data that we used in our previous example, except:

- **LOC=** has been added before City Name.
- Blank Padding has been restored.

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INF
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
            RecordType  $       +Pos1
            ConfYear            +(-1*(Pos1+6))
            ConfName    $
```

 # Absolute Line Pointer Control

Move the line pointer to the xth line of the current input buffer. The size of the input buffer can be overridden by the N= option on the INFILE statement.

<continued>

Using INFILE and INPUT …

Column and Line Pointers

Column Specification

Read the current variable between positions x and y of the current input line – in this example, columns 22 & 35.

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INF
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
            RecordType  $       +Pos1
            ConfYear            +(-1*(Pos1+6))
            ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INF
   INP
```

Column Input

Start and end positions are specified for each input variable. Character data can contain embedded blanks and can exceed 8 characters in length, but it must be aligned consistently throughout the file.

```
RecordType $        +Pos1
ConfYear            +(-1*(Pos1+6))
ConfName   $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

+ Relative Column Pointer Control

Move the column pointer X positions – in this example, 1 position to the right.

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INFILE SASCONF2 ;
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
            RecordType  $       +Pos1
            ConfYear            +(-1*(Pos1+6))
            ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INFILE SASCONF2 ;
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
```

+ Relative Column Pointer Control

Move the column pointer X positions – in this example, 1 position to the right.

And in this example, 39 positions to the left. The + denotes movement – positive number is default, but not mandatory.

<continued>

Using INFILE and INPUT …

Column and Line Pointers

+ Relative Column Pointer Control

```
/* "P
FILEN
DATA
   LEN
   Pos1 =  9;
   Pos
   Str
   INF
   INP
```

And in this example, 9 positions to the right. (The variable `Pos1` was assigned the numeric value 9 and is substituted.)

```
RecordType $        +Pos1
ConfYear            +(-1*(Pos1+6))
ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos1 =  9;
   Pos
   Str
   INF
   INP
```

+ Relative Column Pointer Control

And in this example, 15 positions to the left. (The variable `Pos1` was assigned the numeric value `9`, add `6`, and multiply by `–1` to get `–15`, which is substituted.)

```
        ConfYear              +(-1*(Pos1+6))
        ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

/ (Implied) Relative Line Pointer Control

Move the line pointer one line forward. No flexibility – use 3 slashes to move 3 lines forward, cannot move backwards.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

Move the column pointer to the specified position – in this example, 1 position to the right of the next occurrence of the text string "20" – NOT the 20th position.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
       / @ "20"          Year2Digit2
         @    1          RecordType2 $
         @ Pos2          ConfName2   $  9.
         @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

And then back to the 1st position of the current line.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

Then move the column pointer to position 3 on the current line. (The variable `Pos2` was assigned the numeric value 3 and is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
       / @ "20"          Year2Digit2
         @    1          RecordType2 $
         @ Pos2          ConfName2   $  9.
         @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

Next, move the column pointer to the position immediately after the value "LOC=" on the current line. (The variable `Str1` was assigned the string value "LOC=" and is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
< … first part of input line
        / @ "20"            Year2Digit2
          @    1            RecordType2 $
          @ Pos2            ConfName2   $  9.
          @ Str1            ConfCity2   $ 14.
          @ (Pos2*12+1)     ConfST2     $ ;
    ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

@ Absolute Column Pointer Control

Finally, move the column pointer to position 37 of the current line. (The variable `Pos2` was assigned the numeric value 3, multiply by `12`, and add `1` to get `37`, which is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

… And one more thing …

```
    Pos2 =  3;
    Str1 = "LOC=";
< … first part of input line
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.
          @ (Pos2*12+1)   ConfST2     $ ;
    ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

We need to adjust the value for the year to include the century. (Remember, we used the "20" to find the year – we never actually *read* those 2 digits!)

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Tricks with text column pointers

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Lower case, no trailing blank.

```
NEXT8_1 = "me of th"
```

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Lower case, trailing blank.

NEXT8_2 = "Thebes t"

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Upper case, no trailing blank.

```
NEXT8_3 = " theme o"
```

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Upper case, trailing blank.

NEXT8_4 = “theme of”

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <e
;
```

Upper case, trailing blank, continue on same input line.

NEXT8_5 = “bes thea”

Using INFILE and INPUT …

Informats

Informats are used to interpret incoming data into a form that SAS can understand.

They specify whether an input value is character or numeric, its length, the number of decimal places (if applicable), and other special conditions

Using INFILE and INPUT …

Informats

Two methods to assign an informat to a SAS variable.

1) Temporary – Specify the informat on the INPUT statement.

```
INPUT # 1 @"the"  NEXT8_1 $CHAR8.
```

(This method has already been demonstrated in this presentation.)

Using INFILE and INPUT …

Informats

Two methods to assign an informat to a SAS variable.

2) Permanent – Specify the informat with a separate statement.

```
INFORMAT NEXT8_1 $CHAR8. DEFAULT=$CHAR20.;

ATTRIB variables INFORMAT=informat;
```

Using INFILE and INPUT …

Informats

Five types of SAS Informats:

1) Numeric
2) Character
3) Date / Time
4) Column-binary — not covered
5) User-Defined — `PROC FORMAT`

Using INFILE and INPUT …

Informats (Numeric)

A few examples of SAS Informats

`w.d` **Read numeric value of width *w* with *d* decimal places.**

`COMMAw.d` **Read numeric value of width *w* with *d* decimal places. Ignore embedded commas (and other accounting symbols, such as "$", percent signs, etc.). A left parenthesis at the beginning of the value causes the subsequent numeric value to be treated as a negative.**

Using INFILE and INPUT …

Informats (Character)

A few examples of SAS Informats

`$ w.` Read character value of width *w*, ignore leading blanks (if any).

`$CHARw.` Read character value of width *w*, include leading blanks (if any).

`$ASCIIw.` On IBM Mainframe, convert from ASCII to EBCDIC. On other systems, treat as `$CHAR`.

`$EBCDICw.` On IBM Mainframe, treat as `$CHAR`. On other systems, convert from EBCDIC to ASCII.

Using INFILE and INPUT …

Informats (Date / Time)

A few examples of SAS Informats

`DATEw.` **Read date (*ddMMMyy* or *ddMMMyyyy*) into numeric field.**

`JULIANw.` **Read Julian date (*yyddd* or *yyyyddd*) into numeric field.**

`TIMEw.` **Read time (*hh:mm:ss.ss*) into numeric field.**

Using INFILE and INPUT …

Variable Length Files

```
/* VARIABLE LENGTH FILE EXAMPLE #1*/
FILENAME FAMILY 'USERID.FAMILY.DATA';
DATA FAM_ST;
  ARRAY  ST (5) $ ;
  INFILE FAMILY(STATE) LENGTH=linelen ;
  INPUT @ 1 Person $CHAR10. @ ;
  idx = (linelen-10) / 2;
  DO loop = 1 TO idx;
     INPUT ST(loop) $ 2. @ ;
  END;
RUN;
```

Using INFILE and INPUT …

Variable Length Files

```
/* VARIABLE LENGTH
FILENAME FAMILY 'US
DATA FAM_ST;
  ARRAY  ST (5) $ ;
  INFILE FAMILY(STA
  INPUT @ 1 Person
  idx = (linelen-10
  DO loop = 1 TO id
     INPUT ST(loop)
  END;
RUN;
```

'USERID.FAMILY.DATA(STATE)'

```
Angus      NY
Trixie     ARNYGARIOK
Vladmir    NYFL
Swen       NYMIPA
Petula     NY
Caleb      NYPACA
```

This is the entire dataset. It contains one record per family member, and a variable-length string containing the 2-digit State IDs where that person has lived.

Using INFILE and INPUT …

Variable Length Files

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA FAM_ST;
  ARRAY  ST (5) $ ;
  INFILE FAMILY(STATE) LENGTH=linelen ;
  INPUT @ 1 Person $CHAR10. @ ;
  idx = (linelen-10) / 2;
  DO loop = 1 TO idx;
     INPUT ST(loop) $ 2. @ ;
  END;
RUN;
```

LENGTH= Option

Set a temporary SAS variable to the length of the current input line.

Using INFILE and INPUT …

Variable Length Files

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA FAM_ST;
  ARRAY  ST (5) $ ;
  INFILE FAMILY(STATE) LENGTH=linelen ;
  INPUT @ 1 Person $CHAR10. @ ;
  idx = (linelen-10) / 2;
  DO loop = 1 TO idx;
     INPUT ST(loop) $ 2. @ ;
  END;
RUN;
```

The processing logic …

Use a loop to read in each state abbreviation on each line, however many there happen to be!

Using INFILE and INPUT …

Variable Length Files

```
NOTE: The infile library FAMILY is:
      Dsname=CUSATK.FAMILY.DATA,
      Unit=3390,Disp=OLD,Blksize=27998,
      Lrecl=80,Recfm=VB
NOTE: The infile FAMILY(STATE) is:
      Dsname=USERID.FAMILY.DATA(STATE),
      Unit=3390,Disp=SHR,Blksize=27998,
      Lrecl=80,Recfm=VB
NOTE: A total of 6 records were read from the
      infile library FAMILY.
      The minimum record length was 12.
      The maximum record length was 20.
```

Using INFILE and INPUT …

Variable Length Files

```
NOTE: 6 records were read from the infile
      FAMILY(STATE).
      The minimum record length was 12.
      The maximum record length was 20.
NOTE: The data set WORK.FAM_ST has
      6 observations and 8 variables.
```

< Results of a PROC PRINT >

Obs	ST1	ST2	ST3	ST4	ST5	Person	idx	loop
1	NY					Angus	1	2
2	AR	NY	GA	RI	OK	Trixie	5	6
3	NY	FL				Vladmir	2	3
4	NY	MI	PA			Swen	3	4
5	NY					Petula	1	2
6	NY	PA	CA			Caleb	3	4

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
/* VARIABLE LENGTH FILE EXAMPLE #2*/
FILENAME FAMILY 'USERID.FAMILY.DATA';
DATA CITY (KEEP=city state varlen) ;
  INFILE FAMILY(CITY)  LENGTH=linelen ;
  INPUT @ ;
  INPUT name $VARYING40. linelen;
  city  = SUBSTR( name, 1, linelen-2 );
  state = SUBSTR( name, linelen-1 );
  varlen = linelen;
RUN;
```

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
/* VARIABLE LENGTH
FILENAME FAMILY 'US
DATA CITY (KEEP=cit
  INFILE FAMILY(CIT
  INPUT @ ;
  INPUT name $VARYI
  city  = SUBSTR( n
  state = SUBSTR( n
  varlen = linelen;
RUN;
```

'USERID.FAMILY.DATA(CITY)'

```
San Francisco CA
Bonita Springs FL
Philadelphia PA
Irvine CA
Atlanta GA
Irving TX
Dearborn MI
Seaside OR
Orlando FL
```

This is the entire dataset. It contains the list of cities and states from the earlier "conference" examples.

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA CITY (KEEP=city state varlen) ;
  INFILE FAMILY(CITY)  LENGTH=linelen ;
  INPUT @ ;
  INPUT name $VARYING40. linelen;
  city  = SUBSTR( name, 1, linelen-2 );
  state = SUBSTR( name, linelen-1 );
  varlen = linelen;
RUN;
```

Reads a character value of varying length, based on the specified length (obtained from LENGTH= option).

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA CITY (KEEP=city state varlen) ;
  INFILE FAMILY(CITY)  LENGTH=linelen ;
  INPUT @ ;
  INPUT name $VARYING40. linelen;
  city  = SUBSTR( name, 1, linelen-2 );
  state = SUBSTR( name, linelen-1 );
  varlen = linelen;
RUN;
```

Another way to avoid the infamous SASLOG message:

```
NOTE: INPUT statement reached
      past the end of a line.
```

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA CITY (KEEP=city state varlen) ;
  INFILE FAMILY(CITY)  LENGTH=linelen ;
  INPUT @ ;
  INPUT name $VARYING40. linelen;
  city  = SUBSTR( name, 1, linelen-2 );
  state = SUBSTR( name, linelen-1 );
  varlen = linelen;
RUN;
```

Need to invoke INPUT twice. The 1st time will assign the value of the LENGTH= variable, the 2nd time will use it. (Remember to use the trailing @.)

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
/* VARIABLE LENGTH
FILENAME FAMILY 'U
DATA CITY (KEEP=city state varlen) ;
  INFILE FAMILY(CITY)  LENGTH=linelen ;
  INPUT @ ;
  INPUT name $VARYING40. linelen;
  city  = SUBSTR( name, 1, linelen-2 );
  state = SUBSTR( name, linelen-1 );
  varlen = linelen;
RUN;
```

LENGTH= produces a temporary SAS variable. We need to assign that value to a permanent SAS value if we want to look at it after the DATA step is done.

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
NOTE: The infile library FAMILY is:
      Dsname=CUSATK.FAMILY.DATA,
      Unit=3390,Disp=OLD,Blksize=27998,
      Lrecl=80,Recfm=VB
NOTE: The infile FAMILY(CITY) is:
      Dsname=USERID.FAMILY.DATA(CITY),
      Unit=3390,Disp=SHR,Blksize=27998,
      Lrecl=80,Recfm=VB
NOTE: A total of 9 records were read from the
      infile library FAMILY.
      The minimum record length was 6.
      The maximum record length was 14.
```

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
NOTE: 9 records were read from the
      infile FAMILY(CITY).
      The minimum record length was 6.
      The maximum record length was 14.
NOTE: The data set WORK.CITY has
      9 observations and 4 variables.
```

< Results of a PROC PRINT >

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

< Results of a PROC PRINT >

Obs	city	state	varlen
1	San Francisco	CA	16
2	Bonita Springs	FL	17
3	Philadelphia	PA	15
4	Irvine	CA	9
5	Atlanta	GA	10
6	Irving	TX	9
7	Dearborn	MI	11
8	Seaside	OR	10
9	Orlando	FL	10

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName $  9.
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear      4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMP
DATA SasConf
  LENGTH Con
         Con
  INFILE 'C:
  INPUT Conf
        Conf
        Conf
        Conf
RUN;
```

```
C:\Papers\ConfList.CSV
"SUGI",2006,"San Francisco","CA"
"PHARMASUG",2006,"Bonita Springs","FL"
"NESUG",2006,"Philadelphia","PA"
"WUSS",2006,"Irvine","CA"
"SESUG",2006,"Atlanta","GA"
"SCSUG",2006,"Irving","TX"
"MWSUG",2006,"Dearborn","MI"
"PNWSUG",2006,"Seaside","OR"
"SUGI",2007,"Orlando","FL"
```

This is the entire dataset. It is basically the same data that we used in our previous example, except that:

- Blank Padding has been removed.
- Character strings now in quotes.
- Fields separated by commas, not blanks.

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear      4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **The default delimiter is changed from blank to comma. This default, whether blank or comma, can always be overridden with the DELIMITER= option.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear      4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **Multiple consecutive delimiters (commas) are assumed to represent missing values, instead of being treated as redundant and ignored.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear      4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **Character values are assumed to be enclosed in double quotes; these quotation marks are stripped off the input before storing the value.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear      4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **FYI – You can retain the quotation marks in the variable, if desired, by using the tilde format modifier (~).**

Using INFILE and INPUT …

Comma Separated Value Files

```
NOTE: Invalid data for ConfYear in line 11 10-13.
RULE: ----+----1----+----2----+----3---
11    "SUGI",2006,"San Francisco","CA"
ConfName="SUGI",20 ConfCity=San Francisco"
ConfYear=. ConfST=," _ERROR_=1 _N_=1
NOTE: Invalid data for ConfYear in line 12 10-13.
12    "PHARMASUG",2006,"Bonita Springs","FL"
ConfName="PHARMASU ConfCity=006,"Bonita Sp
ConfYear=. ConfST=ri _ERROR_=1 _N_=2
NOTE: Invalid data for ConfYear in line 13 10-13.
13    "NESUG",2006,"Philadelphia","PA"
ConfName="NESUG",2 ConfCity="Philadelphia"
ConfYear=. ConfST=," _ERROR_=1 _N_=3
```

Etcetera

Using INFILE and INPUT …

2nd iteration

Comma Separated Value Files

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName $  9.
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfList.CSV' DSD;
  INPUT ConfName $CHAR9.
        ConfYear    4.
        ConfCity $CHAR14.
        ConfST   $CHAR2. ;
RUN;
```

Using INFILE and INPUT …

2nd iteration

Comma Separated Value Files

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfListTilde.CSV'
         DSD DLM="~";
  INPUT ConfName $
        ConfYear
        ConfCity $

RUN;
```

DELIMITER= parameter
(DLM=) Delimiter override

- **Changes default delimiter from blank or comma (DSD only) to specified value. All other rules of delimiters unchanged.**

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
/* CSV EXAMP
DATA SasConf
  LENGTH Con
         Con
  INFILE 'C:
         DSD
  INPUT Conf
        Conf
        Conf
        Conf
RUN;
```

```
C:\Papers\ConfListTilde.CSV
"SUGI"~2006~"San Francisco"~"CA"
"PHARMASUG"~2006~"Bonita Springs"~"FL"
"NESUG"~2006~"Philadelphia"~"PA"
"WUSS"~2006~"Irvine"~"CA"
"SESUG"~2006~"Atlanta"~"GA"
"SCSUG"~2006~"Irving"~"TX"
"MWSUG"~2006~"Dearborn"~"MI"
"PNWSUG"~2006~"Seaside"~"OR"
"SUGI",2007,"Orlando","FL"
```

This is the entire dataset. It is the same data that we used in our previous example, except that the comma delimiters have been replaced with tildes.

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

DDE

Work in Progress

See separate PowerPoint presentation for DDE information – not covering in this presentation.

Using INFILE and INPUT …

HTML

Work in Progress

```
x1=htmldecode ('not a &lt;tag&gt;');   not a
<tag>
x2=htmldecode ('&amp;');   &
x3=htmldecode ('&#65;&#66;&#67;');  ABC
```

Using INFILE and INPUT …

INPUT Function

Converts SAS variable from one form to another.

Does not interface with external files.

As such, will not be discussed in this presentation. (A quick explanation is in the paper.)

Conclusion

Many ways to use INFILE and INPUT to process external data.

This is just a starting point and cannot hope to be all-inclusive.

You can learn a lot from my errors, but you will learn a lot more from making and correcting your own.

Conclusion

The author can be contacted at:

KuligowskiConference@gmail.com

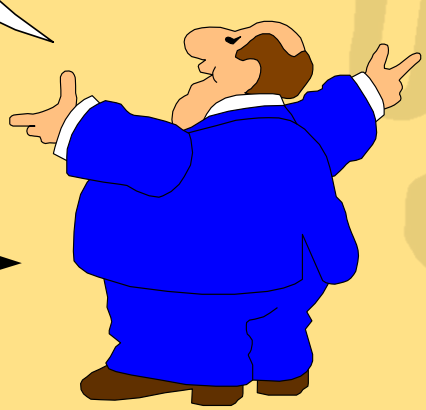
**(Monitored regularly,
but not religiously.
Ain’t retirement wonderful?)**

Using INFILE and INPUT …

Conclusion

SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries. (R) indicates USA registration.

My lawyer →



Andrew Kuligowski

list of 68 titles published in 128 papers

1991–2022

Contents

---

1 alphabetical list of titles

1. Advanced Methods To Introduce External Data Into The SAS(R) System[23] [24] [16]
2. An Introduction To SAS(R) Arrays[100] [114] [117]
3. An Overview Of Techniques To Introduce External Data Into The SAS(R) System[12] [13] [14]
4. Basic Methods To Introduce External Data Into The SAS(R) System[119]
5. Case Study: Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets[101] [123] [125] [97]
6. Character Data: Acquisition, Manipulation, And Analysis[111]
7. Converting A Binary String To A Decimal Numeric — Â“ An Exercise In Problem Solving[75]
8. Datalines And Sequential Files And Csv And Html And More — Using Infile And Input To Introduce External Data Into The SAS(R) System[40]
9. Datalines, Sequential Files, Csv, Html And More: Using Infile And Input To Introduce External Data Into The SAS(R) System[46]
10. Datalines, Sequential Files, Csv, Html, And More — Using Infile And Input Statements To Introduce External Data Into The SAS(R) System[47]

11. Dealing With Missing Data In Epidemiological And Clinical Research[106]
12. Dealing With Non-Traditional Data Formats — Â“ An Update On Writing A Data Parser Using SAS(R)(R)[80]
13. Easy Come, Easy Go — Interactions Between The Data Step And External Files[81] [82] [53]
14. Easy Come, Easy Go- Interactions Between The Data Step And External Files[83]
15. From There To Here: Getting Your Data Into The SAS(R) System.[118]
16. Getting Data Into SAS(R)(R): Infile And Input[48] [41]
17. Getting The Most Out Of Your Conference[54]
18. Getting The Most Out Of Your Pnwsug 2007 Conference[55]
19. Getting The Most Out Of Your SAS(R) Conference[91]
20. Help! — My Merge Statement Has More Than One Data Set With Repeats Of By Values![62]
21. How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U"[29]
22. How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U'?[19]
23. How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?[56]
24. How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?'[57] [36] [37] [38] [18]
25. How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U[49]
26. How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U?[31] [30] [20]
27. Idcams To SAS(R) — The Parseer Two-Step[6]
28. In Search Of The Lost Card[84] [73] [42] [43]
29. Interactions Between The Data Step And External Files — Infile/Input And More[58]
30. Introducing External Data To The SAS(R) System — The Interactive Session[25]
31. Know Thy Data: Techniques For Data Exploration[120] [121] [122]

32. Looking Beneath The Surface Of Sorting[85] [74] [67]
33. Notes In A SAS(R) Log : Of Lost Cards And Merge Statements With Repeats Of By Values[63]
34. Parsing — Using SAS(R) When The Data Are Hiding In A Non-Standard Format[107]
35. Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)[94] [92] [86] [76] [68]
36. Parsing: Using SAS(R) When The Data Are Hiding In A Non-Standard Format[108] [102] [103]
37. Passing Along SAS(R) Data — Set, Merge, And Update[32] [26]
38. Passing Along SAS(R) Data: Set, Merge, And Update[27]
39. Planning For A Smooth Transition To Release 6.06 Of The SAS(R) System In Production Support Or What Do You Mean, 'not Transparent!'[4]
40. Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors[50] [44] [39] [33] [34] [21] [22] [17]
41. Pruning The SAS(R) Log — Â“ Digging Into The Roots Of Notes, Warnings, And Errors[95] [87]
42. Pruning The SAS(R) Log Digging Into The Roots Of Notes, Warnings, And Errors[51]
43. Quote The SAS(R) Log[104] [98]
44. Quote The SAS(R) Log . . . [99]
45. Quote The SAS(R) Log(R)[109] [105]
46. S-M-U (Set, Merge, And Update) Revisited[88] [89]
47. Secrets Of Efficient SAS(R) Coding Techniques[112]
48. So You Want To Be A Manager? — A Panel Discussion On Issues To Consider Before Aiming For A Management Career Path[127] [128]
49. So You Want To Be A Manager? A Discussion On Issues To Consider Before Making A Career Move[3]
50. Software Validation And Testing[10] [11]
51. Sparse Matrices[126]
52. Stalking The Sequential File — A Guide To The Infile And Input Statements[28]

53. Table Look-Up Techniques: Is The Format Procedure The Best Tool For The Job?[113]
54. The Best. Message In The SAS(R) Log[79] [52] [45] [35]
55. The Best. Message In The SAS(R) Log(R)[77]
56. The Building Blocks Of SAS(R) Datasets — S-M-U (Set, Merge, And Update)[96] [69]
57. The Building Blocks Of SAS(R) Datasets — Â“ S-M-U (Set, Merge, And Update)[71]
58. The Ins And Outs And Ups And Downs Of Sorted Data[72] [70]
59. The SAS(R) User Group Newsletter — Start The Presses![7] [5]
60. Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets[124]
61. Using Infile And Input Statements To Introduce External Data Into The SAS(R) System[2] [93] [60] [61]
62. Using Infile And Input Statements To Introduce External Data Into SAS(R)[59]
63. Using Infile And Input Statements To Introduce External Data Into SAS(R)(R)[90] [78]
64. Using SAS(R) To Parse External Data[64] [65] [66]
65. Working With Character Data[1] [110]
66. Working With Sparse Matrices In SAS(R)(R)[116] [115]
67. Writing A Data Parser Using The SAS(R) System[8] [9]
68. You Can Run; But Your Data Cannot Hide: Advanced Methods To Introduce External Data Into The SAS(R) System[15]

2 citations, by year, conference

2022
- SESUG 2022 10 [108] [109]

2020
- SGF 2020 06 [106] [107]

2019
- SGF 2019 04 [105] [124]
- PharmaSUG China 2019 08 [100] [101] [102]
- WUSS 2019 09 [103] [104] [116]
- SCSUG 2019 10 [2]

2018
- SGF 2018 04 [113]
- WUSS 2018 09 [123]
- SESUG 2018 10 [125] [98]
- SCSUG 2018 11 [97] [99]

2017
- SGF 2017 04 [115]
- MWSUG 2017 10 [94] [96]
- SCSUG 2017 10 [95] [126]

2016
- SGF 2016 04 [114] [112]
- SCSUG 2016 11 [91] [92] [93]

2015
- SGF 2015 04 [111] [88]
- PharmaSUG 2015 05 [85] [90]
- WUSS 2015 09 [83] [89]
- MWSUG 2015 10 [81] [84] [87]
- SCSUG 2015 10 [117] [82] [86]

2014
- SGF 2014 03 [1]
- MWSUG 2014 10 [80] [110]

2013
- SGF 2013 04 [120]
- MWSUG 2013 09 [121]
- SESUG 2013 10 [122] [79]

2012
- MWSUG 2012 09 [75] [76] [77] [78]

2011
- MWSUG 2011 09 [73]
- SCSUG 2011 11 [74]

2010
- PharmaSUG 2010 05 [72]
- WUSS 2010 11 [71]

2009
- SGF 2009 03 [70]
- NESUG 2009 09 [68] [69]
- SESUG 2009 10 [67]

2008
- SGF 2008 03 [64]
- NESUG 2008 09 [62] [65]
- SESUG 2008 10 [63] [66]

2007
- SGF 2007 04 [53]
- PNWSUG 2007 09 [55] [56] [60]

- SCSUG 2007 09 [54] [57] [61]
- SESUG 2007 11 [58] [59]

2006
- SUGI 2006 03 [47]
- MWSUG 2006 10 [49] [51]
- SCSUG 2006 10 [48] [50]
- SESUG 2006 10 [46] [52]

2005
- SUGI 2005 04 [42] [127]
- NESUG 2005 09 [41] [43]
- SESUG 2005 10 [40] [44] [3] [45]
- PNWSUG 2005 11 [128]

2004
- SUGI 2004 05 [36]
- PNWSUG 2004 09 [37] [39]
- SESUG 2004 10 [38]

2003
- SUGI 2003 03 [35]
- SESUG 2003 09 [32] [33]
- SCSUG 2003 10 [31] [34]

2002
- SUGI 2002 04 [30]
- NESUG 2002 09 [29]

2001
- SUGI 2001 04 [25] [28]
- SCSUG 2001 08 [23] [26]
- SESUG 2001 08 [24] [27]

2000
- SUGI 2000 04 [20] [21]
- WUSS 2000 09 [19] [22]
- SESUG 2000 10 [18]

1999
- SUGI 1999 04 [16]
- SESUG 1999 10 [17]

1998
- SUGI 1998 03 [119] [15]
- SESUG 1998 09 [12]
- SCSUG 1998 10 [13]
- WUSS 1998 10 [14]

1997
- SUGI 1997 03 [118]

1996
- SUGI 1996 03 [10]

- SESUG 1996 10 [11]

1995
- SUGI 1995 04 [8]
- SESUG 1995 09 [9]

1994
- SUGI 1994 04 [7]
- SESUG 1994 09 [6]

1993
- SESUG 1993 02 [5]

1991
- SUGI 1991 02 [4]

References

Agarwal, Swati and Andrew T. Kuligowski (Mar. 2014). "Working With Character Data". In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings14/2023-2014.pdf.

DeFoor, Jimmy and Andrew Kuligowski (Oct. 2019). "Using Infile And Input Statements To Introduce External Data Into The SAS(R) System". In: *South Central SAS Users Group Conference Proceedings*.

Haworth, Lauren, Stephen M. Noga, and Andrew T. Kuligowski (Oct. 2005). "So You Want To Be A Manager? A Discussion On Issues To Consider Before Making A Career Move". In: *SouthEast SAS Users Group Conference Proceedings*.

Kuligowski, Andrew T. (Feb. 1991). "Planning For A Smooth Transition To Release 6.06 Of The SAS(R) System In Production Support Or What Do You Mean, 'not Transparent!'" In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings-archive/SUGI91/Sugi-91-14%20Kuligowski.pdf.

— (Feb. 1993). "The SAS(R) User Group Newsletter — Start The Presses!" In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/sesug/1993/SESUG93047.pdf.

— (Sept. 1994a). "Idcams To SAS(R) — The Parseer Two-Step". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/sesug/1994/SESUG94021.pdf.

— (Apr. 1994b). "The SAS(R) User Group Newsletter — Start The Presses!" In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings-archive/SUGI94/Sugi-94-221%20Kuligowski.pdf.

— (Apr. 1995a). "Writing A Data Parser Using The SAS(R) System". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings-archive/SUGI95/Sugi-95-179%20Kuligowski.pdf.

— (Sept. 1995b). "Writing A Data Parser Using The SAS(R) System". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/sesug/1995/SESUG95071.pdf.

— (Mar. 1996a). "Software Validation And Testing". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://www.lexjansen.com/sugi/sugi21/bt/061-21.pdf.

— (Oct. 1996b). "Software Validation And Testing". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/sesug/1996/SESUG96030.pdf.

— (Sept. 1998a). "An Overview Of Techniques To Introduce External Data Into The SAS(R) System". In: *SouthEast SAS Users Group Confer-*

ence Proceedings. URL: https://www.lexjansen.com/sesug/1998/SESUG97007.pdf.

Kuligowski, Andrew T. (Oct. 1998b). "An Overview Of Techniques To Introduce External Data Into The SAS(R) System". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/1998/SCSUG98019.pdf.

— (Oct. 1998c). "An Overview Of Techniques To Introduce External Data Into The SAS(R) System". In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/1998/WUSS98109.pdf.

— (Mar. 1998d). "You Can Run; But Your Data Cannot Hide: Advanced Methods To Introduce External Data Into The SAS(R) System". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi23/Advtutor/p45.pdf.

— (Apr. 1999a). "Advanced Methods To Introduce External Data Into The SAS(R) System". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi24/Advtutor/p53-24.pdf.

— (Oct. 1999b). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/1999/042.pdf.

— (Oct. 2000a). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?'" In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2000/p-307.pdf.

— (Sept. 2000b). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U'?" In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2000/WUSS00103.pdf.

— (Apr. 2000c). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U?" In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi25/25/btu/25p054.pdf.

— (Apr. 2000d). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi25/25/po/25p194.pdf.

— (Sept. 2000e). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2000/WUSS00088.pdf.

— (Aug. 2001a). "Advanced Methods To Introduce External Data Into The SAS(R) System". In: *South Central SAS Users Group Conference Pro-*

ceedings. URL: https://www.lexjansen.com/scsug/2001/SCSUG01139.pdf.

Kuligowski, Andrew T. (Aug. 2001b). "Advanced Methods To Introduce External Data Into The SAS(R) System". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2001/P-822.pdf.

— (Apr. 2001c). "Introducing External Data To The SAS(R) System — The Interactive Session". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi26/p159-26.pdf.

— (Aug. 2001d). "Passing Along SAS(R) Data — Set, Merge, And Update". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2001/SCSUG01052.pdf.

— (Aug. 2001e). "Passing Along SAS(R) Data: Set, Merge, And Update". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2001/P-354.pdf.

— (Apr. 2001f). "Stalking The Sequential File — A Guide To The Infile And Input Statements". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi26/p051-26.pdf.

— (Sept. 2002a). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U"". In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug02/bt/bt018.pdf.

— (Apr. 2002b). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U?" In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi27/p052-27.pdf.

— (Oct. 2003a). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U?" In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2003/Kuligowski%20-%20SMU%20REVISED%202002.pdf.

— (Sept. 2003b). "Passing Along SAS(R) Data — Set, Merge, And Update". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2003/IN09-Kuligowski.pdf.

— (Sept. 2003c). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2003/SE09-Kuligowski.pdf.

— (Oct. 2003d). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2003/Kuligowski%20-%20Pruning%20the%20SASLOG%20-%202003.pdf.

— (Mar. 2003e). "The Best. Message In The SAS(R) Log". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://

support.sas.com/resources/papers/proceedings/proceedings/sugi28/119-28.pdf.

Kuligowski, Andrew T. (May 2004a). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?'" In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi29/254-29.pdf.

— (Sept. 2004b). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?'" In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2004/SMUREVISED2003-OldSASDataintonewSASDataset.pdf.

— (Oct. 2004c). "How To Incorporate Old SAS(R) Data Into A New Data Step, Or 'what Is S-M-U?'" In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2004/IN05-Kuligowski.pdf.

— (Sept. 2004d). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2004/PruningtheSASLOG-2003.pdf.

— (Oct. 2005a). "Datalines And Sequential Files And Csv And Html And More — Using Infile And Input To Introduce External Data Into The SAS(R) System". In: *SouthEast SAS Users Group Conference Proceedings*.

— (Sept. 2005b). "Getting Data Into SAS(R)(R): Infile And Input". In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug05/io/io11.pdf.

— (Apr. 2005c). "In Search Of The Lost Card". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/058-30.pdf.

— (Sept. 2005d). "In Search Of The Lost Card". In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug05/cc/cc19.pdf.

— (Oct. 2005e). "Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors". In: *SouthEast SAS Users Group Conference Proceedings*.

— (Oct. 2005f). "The Best. Message In The SAS(R) Log". In: *SouthEast SAS Users Group Conference Proceedings*.

— (Oct. 2006a). "Datalines, Sequential Files, Csv, Html And More: Using Infile And Input To Introduce External Data Into The SAS(R) System". In: *SouthEast SAS Users Group Conference Proceedings*.

— (Mar. 2006b). "Datalines, Sequential Files, Csv, Html, And More — Using Infile And Input Statements To Introduce External Data Into The SAS(R) System". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi31/228-31.pdf.

Kuligowski, Andrew T. (Oct. 2006c). “Getting Data Into SAS(R)(R): Infile And Input”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2006/KuligowskiAndy_Input.pdf.

— (Oct. 2006d). “How To Incorporate Old SAS(R) Data Into A New Data Step, Or What Is S-M-U”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2006/tutorials/MWSUG-2006-TU09.pdf.

— (Oct. 2006e). “Pruning The SAS(R) Log — Digging Into The Roots Of Notes, Warnings, And Errors”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2006/KuligowskiAndy_SASLOG.pdf.

— (Oct. 2006f). “Pruning The SAS(R) Log Digging Into The Roots Of Notes, Warnings, And Errors”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2006/appdev/MWSUG-2006-AD02.pdf.

— (Oct. 2006g). “The Best. Message In The SAS(R) Log”. In: *SouthEast SAS Users Group Conference Proceedings*.

— (Apr. 2007a). “Easy Come, Easy Go — Interactions Between The Data Step And External Files”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/forum2007/220-2007.pdf.

— (Sept. 2007b). “Getting The Most Out Of Your Conference”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2007/report/Report-Kuligowski.pdf.

— (Sept. 2007c). “Getting The Most Out Of Your Pnwsug 2007 Conference”. In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2007/Andy%20Kuligowski%20---%20Getting%20the%20Most%20Out%20of%20Your%20Conference.pdf.

— (Sept. 2007d). “How To Incorporate Old SAS(R) Data Into A New Data Step, Or ’what Is S-M-U?” In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2007/Andy%20Kuligowski%20---%20SMU%202007%20---%202%20column.pdf.

— (Sept. 2007e). “How To Incorporate Old SAS(R) Data Into A New Data Step, Or ’what Is S-M-U?’” In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2007/data/Data-Kuligowski.pdf.

— (Nov. 2007f). “Interactions Between The Data Step And External Files — Infile/Input And More”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2007/IS04.pdf.

— (Nov. 2007g). “Using Infile And Input Statements To Introduce External Data Into SAS(R)”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2007/HW07.pdf.

Kuligowski, Andrew T. (Sept. 2007h). “Using Infile And Input Statements To Introduce External Data Into The SAS(R) System”. In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2007/how/AndyKuligowski-UsingINFILEandINPUTStatements.pdf.

— (Sept. 2007i). “Using Infile And Input Statements To Introduce External Data Into The SAS(R) System”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2007/how/HOW-Kuligowski.pdf.

— (Sept. 2008a). “Help! — My Merge Statement Has More Than One Data Set With Repeats Of By Values!” In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug08/cc/cc21.pdf.

— (Oct. 2008b). “Notes In A SAS(R) Log : Of Lost Cards And Merge Statements With Repeats Of By Values”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2008/SBC-117.pdf.

— (Mar. 2008c). “Using SAS(R) To Parse External Data”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/190-2008.pdf.

— (Sept. 2008d). “Using SAS(R) To Parse External Data”. In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug08/ff/ff08.pdf.

— (Oct. 2008e). “Using SAS(R) To Parse External Data”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2008/HOW-065.pdf.

— (Oct. 2009a). “Looking Beneath The Surface Of Sorting”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2009/FF005.Kuligowski.pdf.

— (Sept. 2009b). “Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)”. In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug09/hw/HW02.pdf.

— (Sept. 2009c). “The Building Blocks Of SAS(R) Datasets — S-M-U (Set, Merge, And Update)”. In: *NorthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/nesug/nesug09/ff/FF15.pdf.

— (Mar. 2009d). “The Ins And Outs And Ups And Downs Of Sorted Data”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings09/142-2009.pdf.

— (Nov. 2010a). “The Building Blocks Of SAS(R) Datasets — Â“ S-M-U (Set, Merge, And Update)”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2010/TUT/2982_2_TUT-Kuligowski.pdf.

— (May 2010b). “The Ins And Outs And Ups And Downs Of Sorted Data”. In: *Pharmaceutical SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/pharmasug/2010/TU/TU07.pdf.

Kuligowski, Andrew T. (Sept. 2011a). “In Search Of The Lost Card”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2011/coders/MWSUG-2011-CC15.pdf.

— (Nov. 2011b). “Looking Beneath The Surface Of Sorting”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2011/kuligowski/Looking%20Beneath%20the%20Surface%20of%20Sorting.pdf.

— (Sept. 2012a). “Converting A Binary String To A Decimal Numeric — Â“ An Exercise In Problem Solving”. In: *MidWest SAS Users Group Annual Conference Proceedings*.

— (Sept. 2012b). “Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2012/HW/MWSUG-2012-HW04.pdf.

— (Sept. 2012c). “The Best. Message In The SAS(R) Log(R)”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2012/S1/MWSUG-2012-S116.pdf.

— (Sept. 2012d). “Using Infile And Input Statements To Introduce External Data Into SAS(R)(R)”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2012/HW/MWSUG-2012-HW05.pdf.

— (Oct. 2013). “The Best. Message In The SAS(R) Log”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2013/CC-14.pdf.

— (Oct. 2014). “Dealing With Non-Traditional Data Formats — Â“ An Update On Writing A Data Parser Using SAS(R)(R)”. In: *MidWest SAS Users Group Annual Conference Proceedings*.

— (Oct. 2015a). “Easy Come, Easy Go — Interactions Between The Data Step And External Files”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2015/BI/MWSUG-2015-BI-08.pdf.

— (Oct. 2015b). “Easy Come, Easy Go — Interactions Between The Data Step And External Files”. In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2015/Easy-Come-Easy-Go.pdf.

— (Sept. 2015c). “Easy Come, Easy Go- Interactions Between The Data Step And External Files”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2015/146_Final_Paper_PDF.pdf.

— (Oct. 2015d). “In Search Of The Lost Card”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2015/BI/MWSUG-2015-BI-03.pdf.

— (May 2015e). “Looking Beneath The Surface Of Sorting”. In: *Pharmaceutical SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/pharmasug/2015/TT/PharmaSUG-2015-TT13.pdf.

Kuligowski, Andrew T. (Oct. 2015f). "Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)". In: *South Central SAS Users Group Conference Proceedings*.

— (Oct. 2015g). "Pruning The SAS(R) Log — Â" Digging Into The Roots Of Notes, Warnings, And Errors". In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2015/SA/MWSUG-2015-SA-05.pdf.

— (Apr. 2015h). "S-M-U (Set, Merge, And Update) Revisited". In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings15/3444-2015.pdf.

— (Sept. 2015i). "S-M-U (Set, Merge, And Update) Revisited". In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2015/75_Final_Paper_PDF.pdf.

— (May 2015j). "Using Infile And Input Statements To Introduce External Data Into SAS(R)(R)". In: *Pharmaceutical SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/pharmasug/2015/HT/PharmaSUG-2015-HT07.pdf.

— (Nov. 2016a). "Getting The Most Out Of Your SAS(R) Conference". In: *South Central SAS Users Group Conference Proceedings*.

— (Nov. 2016b). "Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2016/Parsing-Useful-Data-out-of-Unusual-Formats-Using-SAS-SCSUG-2016.pdf.

— (Nov. 2016c). "Using Infile And Input Statements To Introduce External Data Into The SAS(R) System". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2016/Using-INFILE-INPUT-2016_SCSUG.pdf.

— (Oct. 2017a). "Parsing Useful Data Out Of Unusual Formats Using SAS(R)(R)". In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2017/SA/MWSUG-2017-SA09.pdf.

— (Oct. 2017b). "Pruning The SAS(R) Log — Â" Digging Into The Roots Of Notes, Warnings, And Errors". In: *South Central SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/scsug/2017/Pruning-the-SASLOG-SCSUG-2017.pdf.

— (Oct. 2017c). "The Building Blocks Of SAS(R) Datasets — S-M-U (Set, Merge, And Update)". In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2017/SA/MWSUG-2017-SA10.pdf.

— (Nov. 2018a). "Case Study: Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets". In: *South Central SAS Users Group Conference Proceedings*.

— (Oct. 2018b). "Quote The SAS(R) Log". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2018/SESUG2018_Paper-256_Final_PDF.pdf.

— (Nov. 2018c). "Quote The SAS(R) Log . . . " In: *South Central SAS Users Group Conference Proceedings*.

Kuligowski, Andrew T. (Aug. 2019a). “An Introduction To SAS(R) Arrays”. In: *Pharmaceutical Software Users Group China*. URL: https://www.lexjansen.com/pharmasug-cn/2019/AD/Pharmasug-China-2019-AD41.pdf.

— (Aug. 2019b). “Case Study: Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets”. In: *Pharmaceutical Software Users Group China*. URL: https://www.lexjansen.com/pharmasug-cn/2019/DM/Pharmasug-China-2019-DM40.pdf.

— (Aug. 2019c). “Parsing: Using SAS(R) When The Data Are Hiding In A Non-Standard Format”. In: *Pharmaceutical Software Users Group China*. URL: https://www.lexjansen.com/pharmasug-cn/2019/HW/Pharmasug-China-2019-HW32.pdf.

— (Sept. 2019d). “Parsing: Using SAS(R) When The Data Are Hiding In A Non-Standard Format”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2019/198_Final_Paper_PDF.pdf.

— (Sept. 2019e). “Quote The SAS(R) Log”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2019/154_Final_Paper_PDF.pdf.

— (Apr. 2019f). “Quote The SAS(R) Log(R)”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3642-2019.pdf.

— (June 2020a). “Dealing With Missing Data In Epidemiological And Clinical Research”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/5072-2020.pdf.

— (June 2020b). “Parsing — Using SAS(R) When The Data Are Hiding In A Non-Standard Format”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4230-2020.pdf.

— (Oct. 2022a). “Parsing: Using SAS(R) When The Data Are Hiding In A Non-Standard Format”. In: *SouthEast SAS Users Group Conference Proceedings*.

— (Oct. 2022b). “Quote The SAS(R) Log(R)”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://www.lexjansen.com/sesug/2022/SESUG2022_Paper_168_Final_PDF.pdf.

Kuligowski, Andrew T. and Swati Agarwal (Oct. 2014). “Working With Character Data”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2014/SA/MWSUG-2014-SA06.pdf.

— (Apr. 2015). “Character Data: Acquisition, Manipulation, And Analysis”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings15/3224-2015.pdf.

— (Apr. 2016). “Secrets Of Efficient SAS(R) Coding Techniques”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings16/11741-2016.pdf.

Kuligowski, Andrew T. and Swati Agarwal (Apr. 2018). “Table Look-Up Techniques: Is The Format Procedure The Best Tool For The Job?” In: *SAS Global Forum Annual Conference Proceedings*. URL: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2774-2018.pdf.

Kuligowski, Andrew T. and Lisa Mendez (Apr. 2016). “An Introduction To SAS(R) Arrays”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings16/6406-2016.pdf.

— (Apr. 2017). “Working With Sparse Matrices In SAS(R)(R)”. In: *SAS Global Forum Annual Conference Proceedings*.

— (Sept. 2019). “Working With Sparse Matrices In SAS(R)(R)”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2019/155_Final_Paper_PDF.pdf.

Kuligowski, Andrew T. and Lisa Morrow Mendez (Oct. 2015). “An Introduction To SAS(R) Arrays”. In: *South Central SAS Users Group Conference Proceedings*.

Kuligowski, Andrew T. and Nancy Roberts (Mar. 1997). “From There To Here: Getting Your Data Into The SAS(R) System.” In: *SAS Users Group International Annual Conference Proceedings*.

— (Mar. 1998). “Basic Methods To Introduce External Data Into The SAS(R) System”. In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi23/Begtutor/p51.pdf.

Kuligowski, Andrew T. and Charu Shankar (Apr. 2013a). “Know Thy Data: Techniques For Data Exploration”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings13/145-2013.pdf.

— (Sept. 2013b). “Know Thy Data: Techniques For Data Exploration”. In: *MidWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/mwsug/2013/HW/MWSUG-2013-HW01.pdf.

— (Oct. 2013c). “Know Thy Data: Techniques For Data Exploration”. In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2013/HOW-02.pdf.

Mendez, Lisa and Andrew Kuligowski (Sept. 2018a). “Case Study: Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets”. In: *Western Users of SAS Software Annual Conference Proceedings*. URL: https://www.lexjansen.com/wuss/2018/10_Final_Paper_PDF.pdf.

— (Apr. 2019). “Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets”. In: *SAS Global Forum Annual Conference Proceedings*. URL: https://www.sas.com/content/

dam/SAS/support/en/sas-global-forum-proceedings/2019/3051-2019.pdf.

Mendez, Lisa and Andrew T. Kuligowski (Oct. 2018b). "Case Study: Using Base SAS(R) To Automate Quality Checks Of Excel Workbooks That Have Multiple Worksheets". In: *SouthEast SAS Users Group Conference Proceedings*. URL: https://analytics.ncsu.edu/sesug/2018/SESUG2018_Paper-253_Final_PDF.pdf.

Mendez, Lisa and Andy Kuligowski (Oct. 2017). "Sparse Matrices". In: *South Central SAS Users Group Conference Proceedings*.

Noga, Stephen M., Lauren Haworth, and Andrew T. Kuligowski (Apr. 2005a). "So You Want To Be A Manager? — A Panel Discussion On Issues To Consider Before Aiming For A Management Career Path". In: *SAS Users Group International Annual Conference Proceedings*. URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/143-30.pdf.

— (Nov. 2005b). "So You Want To Be A Manager? — A Panel Discussion On Issues To Consider Before Aiming For A Management Career Path". In: *Pacific NorthWest SAS Users Group Annual Conference Proceedings*. URL: https://www.lexjansen.com/pnwsug/2005/Lauren%20Haworth%20---%20So%20You%20Want%20to%20Be%20a%20Manager.pdf.