Looking Beneath the Surface of Sorting

Andrew T. Kuligowski

FCCI Insurance Group

Presented at the Toronto Area SAS Society

05 March 2010

- INTRODUCTION
- SORTING – THE BASICS
- WHAT *IS* "NUMERICAL ORDER"?
- SORTING ALPHANUMERIC CHARACTERS / VARIATIONS IN OPERATING SYSTEMS?
- SPACE CONCERNS

Introduction

- SELECTIVE RECORD RETENTION and TIEBREAKERS
- ALTERNATIVES TO SORTING
- SYSTEM OPTIONS RELATED TO SORTING
- CONCLUSION

sort [*sOrt*] **To arrange into some order, especially numerically, alphabetically or chronologically. From Old French *sortir* ("allot, sort"), from Latin *sortiri* ("draw lots, divide, choose")**

Basics of sorting:

- Name of input dataset.
- Variable(s) to be sorted.

```
PROC SORT DATA=< DatasetName >;
    BY  Variable1  < Variable2 … >;
RUN;
```

A *slightly* more complicated sort:

- Name of output dataset (if different).
- Direction of sort for each variable.

```
PROC SORT DATA=< DatasetName >
          OUT=< OutputDatasetName >;
   BY  < DESCENDING >  Variable1
       << DESCENDING > Variable2 … >;
RUN;
```

Sorting – the Basics

This is sorting in SAS at its most basic level.

(Everyone who was simply looking for a high level overview can leave now …)

(Then again, I assume everyone in the room covered this in their Intro to SAS class – *early* in their Intro to SAS class – and is expecting some additional details!)

Numerical Order

(A refresher from elementary school …)

I think everyone in the room can count, forwards and backwards … some in several different languages!

Positive numbers are higher than negative numbers, with zero falling between them.

What about missing values ???

What *is* "numerical order"?

Let's find out via experimentation.

Obs	Orig Order	RandomPos NegMiss
1	1	1
2	3	2
3	8	4
4	11	3
5	14	-1
6	17	5
7	19	-3
8	20	.
9	22	-5
10	27	-4
11	78	-2

→ *SORT*

Obs	Orig Order	RandomPos NegMiss
1	20	.
2	22	-5
3	27	-4
4	19	-3
5	78	-2
6	14	-1
7	1	1
8	3	2
9	11	3
10	8	4
11	17	5

What *is* "numerical order"?

So, missing values are the smallest "number" that we have.

What about Special Missing Values?

What *is* "numerical order"?

Let's find out via experimentation.

Obs	Orig Order	RandomPos NegMiss
1	1	1
2	3	2
3	10	M
4	11	3
5	14	-1
6	19	-3
7	20	.
8	27	-4
9	65	A
10	78	-2
11	85	_

➔ *SORT*

Obs	Orig Order	RandomPos NegMiss
1	85	_
2	20	.
3	65	A
4	10	M
5	27	-4
6	19	-3
7	78	-2
8	14	-1
9	1	1
10	3	2
11	11	3

Special Missing Values

What *is* "numerical order"?

Let's fin ... ation.

Obs	Orig Order	RandomPos NegMiss
1	85	_
2	20	.
3	65	A
4	10	M
5	27	-4
6	19	-3

The underscore is lowest.
Then comes "normal" missing value.
Finally, comes alpha characters A – Z.

Sorting Alphanumeric Characters

(Another refresher from elementary school …)

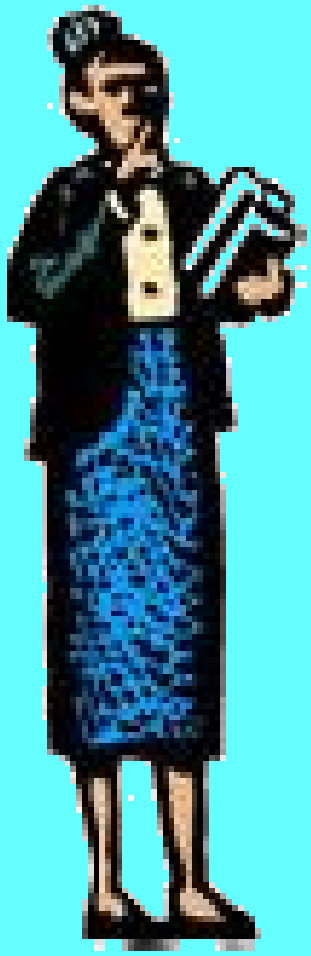
A B C D E F G

(But did Big Bird discuss the difference between upper and lower case letters?)

a b c d e f g

(And what about "special characters"?)

@ % * + < >



My elementary school librarian told me:

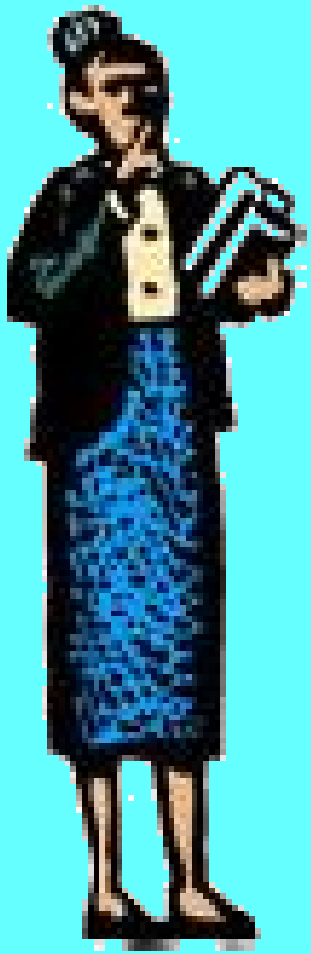
"Use alphabetical order – that's why they call it that!"

"In case of ties, UPPER CASE dominates lower case."

"Ignore special characters altogether."

"Shhhhhhh …"

Sorting Alphanumeric Characters / Variations in Operating Systems



My elementary school

"U why

"I

"I

"S

BUT …

My elementary school librarian never heard of EBCDIC or ASCII !

(Come to think of it, she probably never heard of "computers" either … it *was* a long time ago!)

Sorting Alphanumeric Characters

EBCDIC (Z/OS i.e. "*mainframes*")

- Lower case before upper case.
- Upper case before digits.
- Both lower case and upper case letter sequences interrupted by special characters.

Sorting Alphanumeric Characters

EBCDIC (Z/OS i.e. "*mainframes*")

```
<blank> . < ( + | & ! $ * ) ; ¬
        - / , % _ > ?: # @ ' = "

a b c d e f g h i j k l m
n o p q r ~ s t u v w x y z

{ A B C D E F G H I } J K L
M N O P Q R \ S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9
```

Sorting Alphanumeric Characters

ASCII (Unix & derivatives, Windows, OpenVMS)

- Digits before upper case.
- Upper case before lower case.
- Some special characters before digits, some after digits but before alphas, some after upper case, rest after lower case. No special characters interrupt the alphabetic sequences.

Sorting Alphanumeric Characters

ASCII (Unix & derivatives, Windows, OpenVMS)

```
<blank> ! " # $ % & ' ( ) * + , - . /

0 1 2 3 4 5 6 7 8 9 : ; < = > ? @

A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _

a b c d e f g h i j k l m n
o p q r s t u v w x y z { } ~
```

Sorting Alphanumeric Characters

ASCII (Unix & derivatives, Windows, OpenVMS)

So … sort results for alphanumeric characters will return different output on a mainframe and on a PC.

This is an exception to the philosophy of "transparent results across operating systems"!!

Sorting Alphanumeric Characters

ASCII (Unix & derivatives, Windows, OpenVMS)

Philosophical question :

Which do you prefer?

- **100% fully compatible results across operating systems?**

Or

- **Should SAS be consistent with other applications on a given operating system?**

You Can Have BOTH !!!

Sorting Alphanumeric Characters

ASCII (Unix & derivatives, Windows, OpenVMS)

By Default, SAS will sort using the appropriate character set for the operating system.

Or

You can override the default for your operating system by using the `ASCII` or `EBCDIC` keyword (as appropriate) on PROC SORT!

Let's find out via experimentation.

```
PROC SORT DATA=AscEbc
          ASCII ;
   BY  AByte;
RUN;
```

Obs	AByte	Obs	AByte
1		11	>
2	#	12	@
3	$	13	A
4	,	14	F
5	.	15	[
6	0	16	]
7	1	17	_
8	9	18	a
9	;	19	f
10	<	20	{
		21	}

```
PROC SORT DATA=AscEbc
          EBCDIC;
   BY  OneByte;
RUN;
```

Obs	AByte	Obs	AByte
1		11	a
2	.	12	f
3	<	13	[
4	$	14	]
5	;	15	{
6	,	16	A
7	_	17	F
8	>	18	}
9	#	19	0
10	@	20	1
		21	9

Let's find out via experimentation.

```
PROC SORT DATA=AscEbc
          ASCII ;
   BY  AByte;
RUN;
Obs AByte   Obs AByte
 1
 2
 3
 4
 5
 6     0     16     ]
 7     1     17     _
 8     9     18     a
 9     ;     19     f
10     <     20     {
            21     }
```

```
PROC SORT DATA=AscEbc
          EBCDIC ;
   BY  OneByte;
RUN;
Obs AByte   Obs AByte
 6     ,     16     A
 7     _     17     F
 8     >     18     }
 9     #     19     0
10     @     20     1
            21     9
```

Using the appropriate keywords, the end results will be consistent regardless of the machine / operating system used.

ASCII vs. EBCDIC

Sorting Alphanumeric Characters / Variations in Operating Systems

Oh, and if that wasn't enough, … let's throw one more complicating factor into the mix!

How many people in the room are NOT from the United States or Canada?

(OK, everyone from Great Britain, Australia, and New Zealand, put your hands down, too.)

Oh, and if that wasn't enough, … let's throw one more complicating factor into the mix!

Scandinavians might be interested in this!

```
PROC SORT DATA=< DatasetName >
          SWEDISH ;
   BY  < DESCENDING >  Variable1
       << DESCENDING > Variable2 … >;
RUN;
```

or FINNISH

or DANISH

or NORWEGIAN

Sorry, no ICELANDIC or FAROESE

Oh, and if that wasn't enough, … let's throw one more complicating factor into the mix!

Europeans, Central and South Americans, …

~~Scandinavians~~ might be interested in this:

```
PROC SORT DATA=< DatasetName >
          SORTSEQ=SWEDISH ;
   BY  < DESCENDING >  Variable1
       << DESCENDING > Variable2 … >;
RUN;
```

or SPANISH

or ITALIAN

or POLISH (ver 9.2)

or FINNISH

or DANISH

or NORWEGIAN

Collating Sequences

Sorting Alphanumeric Characters / Variations in Operating Systems

Oh, and if that wasn't enough ... let's throw one r

How did Scandinavia get first dibs?

I suspect … they offered the Powers-That-Be at SAS something that no one else could …

Scandinavians might be interested in th

```
PROC SORT

   BY   <
        <                    >;
RUN;
```

or

SPANISH

or

ITALIAN

or

POLISH

(ver 9.2)

o

FINNISH DANISH NORWEGIAN

Oh, and if that wasn't enough, … let's throw one more complicating factor into the mix!

What if your preferred sort sequence still isn't available?

PROC TRANTAB can create, edit and/or display a customized translation table.

(Looks like our friends from Iceland and the Faroe Islands might be alright, after all!)

Scandinavians might be interested in th… $, …

PROC … ISH

BY < DESCENDING > Variable1 … ITALIAN

RUN;

or

POLISH

(ver 9.2)

or FINNISH

or DANISH

or NORWEGIAN

Collating Sequences

Sorting Alphanumeric Characters / Variations in Operating Systems

Oh, and if that wasn't enough let's throw one more nix!

Wh nce

Sca

P it

PROC

BY

RUN ;

ISH

ITALIAN

or

POLISH

(ver 9.2)

FINNISH DANISH NORWEGIAN

For the record, only one collating sequence is allowed per unique SORT.

(which makes sense … why would you want to sort by both Finnish AND Polish at the same time?)

Intertwining Upper and Lower Case

Both EBCDIC and ASCII group all 26 upper case letters and all 26 lower case letters together (even if they can't agree on which comes first).

Who wants to have them sorted intertwined?

Let's find out via experimentation.

Obs	Orig Order	Three Letter Mixed
4	4	MZx
20	20	MJv
42	42	mVX
44	44	MPY
62	62	mCA
140	140	MfE
143	143	MWz
159	159	Mda
187	187	MEq
191	191	moU

Let's find out via experimentation.

```
PROC SORT DATA=SampData(KEEP=OrigOrder ThreeLetterMixed)
          OUT=TempMixed;
       BY  ThreeLetterMixed ;
RUN;

PROC PRINT DATA=TempMixed(
      WHERE=(UPCASE(SUBSTR(ThreeLetterMixed,1,1))='M'))
      UNIFORM;
RUN;
```

```
159   159    Mda
187   187    MEq
191   191    moU
```

Let's find out via experimentation.

Obs	Orig Order	Three Letter Mixed
4	4	MZx
20	20	MJv
42	42	mVX
44	44	MPY
62	62	mCA
140	140	MfE
143	143	MWz
159	159	Mda
187	187	MEq
191	191	moU

```
mpData(KEEP=OrigOrder Th
pMixed;
etterMixed ;

empMixed(
SE(SUBSTR LetterMix
```

➔ *SORT*

Obs	Orig Order	Three Letter Mixed
46	187	MEq
47	20	MJv
48	44	MPY
49	143	MWz
50	4	MZx
51	159	Mda
52	140	MfE
187	62	mCA
188	42	mVX
189	191	moU

Note that *UPPER CASE* comes before *lower case*. This indicates that the sort must have been done in ASCII.

Intertwining Upper and Lower Case

Let's add a variable that contains the same content as our character string, only with all fields converted to upper case. Then, sort by THAT field instead of the original one.

Let's find out via experimentation.

```
DATA Temp2;
   SET  SampData(KEEP=OrigOrder
                     ThreeLetterMixed ThreeLetterCaps);
   ThreeLetterCaps = UPCASE( ThreeLetterMixed );
RUN;

PROC SORT DATA=Temp2 ;
      BY  ThreeLetterCaps ;
RUN;

PROC PRINT  DATA=Temp2(
      WHERE=(SUBSTR(ThreeLetterCaps,1,1)='M'))
      UNIFORM;
RUN;
```

Intertwining Upper and Lower Case

Sorting Alphanumeric Characters / Variations in Operating Systems

Let's find out via experimentation.

Obs	Orig Order	Three Letter Mixed	Three Letter Caps
4	4	MZx	MZX
20	20	MJv	MJV
42	42	mVX	MVX
44	44	MPY	MPY
62	62	mCA	MCA
140	140	MfE	MFE
143	143	MWz	MWZ
159	159	Mda	MDA
187	187	MEq	MEQ
191	191	moU	MOU

SORT

Obs	Orig Order	Three Letter Mixed	Three Letter Caps
109	62	mCA	MCA
110	159	Mda	MDA
111	187	MEq	MEQ
112	140	MfE	MFE
113	20	MJv	MJV
114	191	moU	MOU
115	44	MPY	MPY
116	42	mVX	MVX
117	143	MWz	MWZ
118	4	MZx	MZX

Intertwining Upper and Lower Case

That was an easy fix! Except …

a) It added an extra DATA step to the routine.
   i.e. extra clock time and extra CPU time

b) It added an extra variable to our dataset.
   i.e. extra disk space

Intertwining Upper and Lower Case

Can we accomplish the same thing without the extra overheads in time and space (i.e. $$$)?

Version 9.1	**no**
Version 9.2	**yes**

Let's find out via experimentation.

```
DATA Temp2;
   SET  SampData(KEEP=OrigOrder
                        ThreeLetterMixed ThreeLetterCaps);
   ThreeLetterCaps = UPCASE( ThreeLetterMixed );
RUN;

PROC SORT DATA=Temp2  SampData  SORTSEQ=linguistic ;
       BY  ThreeLetterCaps;  ThreeLetterMixed ;
RUN;
                  SampData
PROC PRINT  DATA=Temp2( ThreeLetterMixed
      WHERE=(SUBSTR(ThreeLetterCaps,1,1)='M'))
      UNIFORM;
RUN;
```

Let's find out via experimentation.

```
DATA Temp2;
    SET SampData(KEEP=OrigOrder
                                    s);
    Th
RUN;

PROC
RUN;

PROC PRINT  DATA=Temp2( ThreeLetterMixed
      WHERE=(SUBSTR(ThreeLetterCaps,1,1)='M'))
      UNIFORM;
RUN;
```

```
PROC SORT DATA=SampData SORTSEQ=linguistic;
       BY  ThreeLetterMixed ;
RUN;

PROC PRINT  DATA=SampData(
      WHERE=(SUBSTR(ThreeLetterMixed,1,1)='M'))
      UNIFORM;
RUN;
```

Let's find out via experimentation.

Obs	Orig Order	Three Letter Mixed
4	4	MZx
20	20	MJv
42	42	mVX
44	44	MPY
62	62	mCA
140	140	MfE
143	143	MWz
159	159	Mda
187	187	MEq
191	191	moU

```
A=SampData SORTSEQ=l
eeLetterMixed ;

ATA=
SUBS'
;

mp2( ThreeLetterMix
R(ThreeLetterCaps,1,
```

→ *SORT*

Obs	Orig Order	Three Letter Mixed
109	62	mCA
110	159	Mda
111	187	MEq
112	140	MfE
113	20	MJv
114	191	moU
115	44	MPY
116	42	mVX
117	143	MWz
118	4	MZx

```
RUN;
```

Let's find out via experimentation.

Obs	Orig Order	ig er	Three Letter Mixed
4	4	2	mCA
20	20	9	Mda
42	42	7	MEq
44	44	0	MfE
62	62	0	MJv
140	140	1	moU
143	143	4	MPY
159	159	2	mVX
187	187	3	MWz
191	191	4	MZx

```
RUN;
```

The SAS 9.2 documentation states that the LINGUISTIC option on SORTSEQ "are largely compatible with" the Unicode Collation Algorithms (UCA).

The converse is that they are not 100% compatible with UCA – make a note, in case this is of concern.

Character data containing numbers

How does the following data sort?

Numbers inside character data

2 14 43 4 1 27

1 2 4 14 27 43

(if they're stored as numbers)

1 14 2 27 4 43

(if they're stored as characters)

Numbers inside character data

Sorting Alphanumeric Characters / Variations in Operating Systems

Let's find out via experimentation.

```
Obs ADDRESS
 1  1411 12th Street
 2  14 121st Street
 3  141 12th Street Apt 2
 4  14 12th Street
 5  141 121st Street
 6  141 12th Street Apt 11
 7  141 12th Street Apt 1
```

```
Obs ADDRESS
 1  14 12th Street
 2  14 121st Street
 3  141 12th Street Apt 1
 4  141 12th Street Apt 11
 5  141 12th Street Apt 2
 6  141 121st Street
 7  1411 12th Street
```

```
PROC SORT DATA=ADDRESSES
          OUT=ADDRESSES_CHAR ;
   BY  ADDRESS ;
RUN;
```

Let's find out via experimentation.

Obs	ADDRESS
1	1411 12th Street
2	14 121st Street
3	141 12th Street Apt 2
4	14 12th Street
5	141 121st Street
6	141 12th Street Apt 11
7	141 12th Street Apt 1

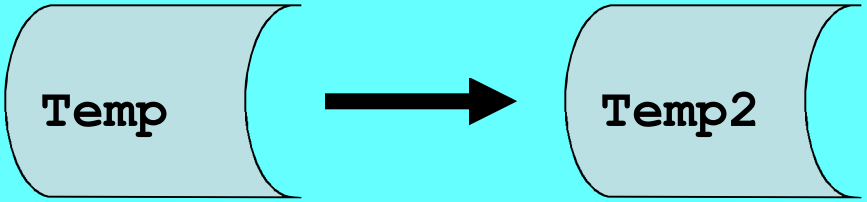
Obs	ADDRESS
1	14 12th Street
2	14 121st Street
3	141 12th Street Apt 1
4	141 12th Street Apt 2
5	141 12th Street Apt 11
6	141 121st Street
7	1411 12th Street

```
PROC SORT DATA=ADDRESSES
          OUT=ADDRESSES_NUM
  SORTSEQ=linguistic(NUMERIC_COLLATION=ON);
  BY  ADDRESS ;
RUN;
```

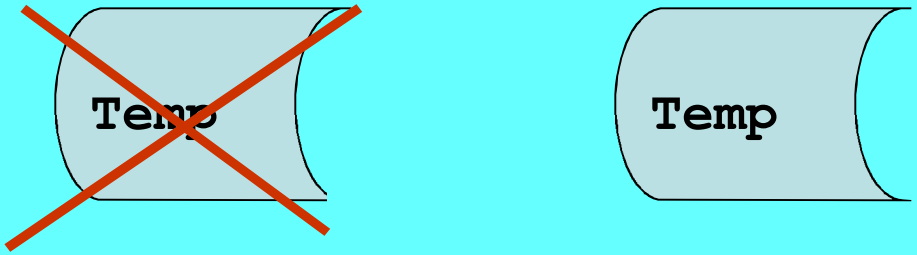
Sorting can take a lot of disk space!

Aside: Picture a simple DATA Step:

```
DATA Temp2;
   SET  Temp;
RUN;
```




```
DATA Temp;
   SET  Temp;
RUN;
```

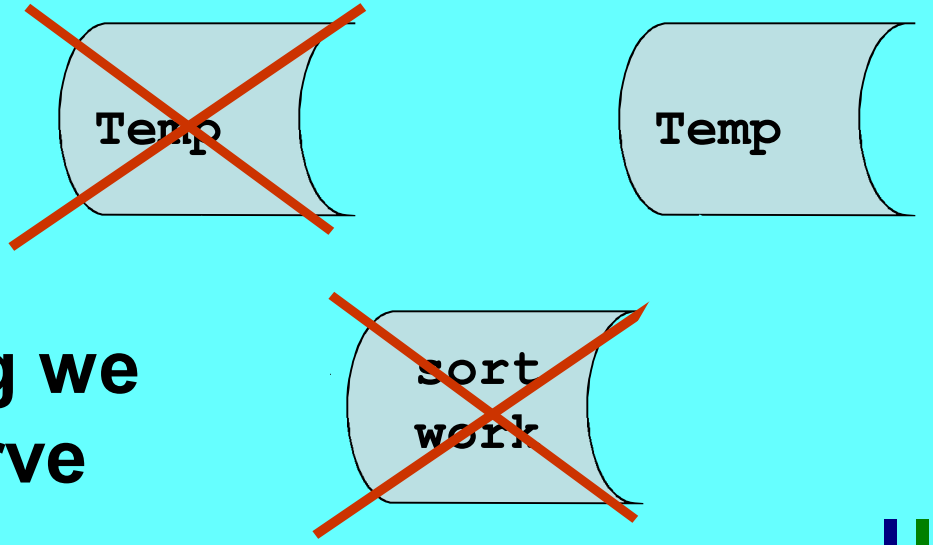


Space Concerns

Sorting can take a lot of disk space!

Well, it's worse with PROC SORT!

```
PROC SORT DATA=Temp;
   BY something;
RUN;
```



Is there something we can do to conserve disk space?

Space Concerns

Sorting can take a lot of disk space!

1) Only bring along the variables you need!

```
DATA Temp2;
   SET Temp;
   DROP  OldVar1 OldVar2;
RUN;

PROC SORT DATA=Temp (KEEP=GoodVar1-GoodVarX);
          OUT=Temp2 (DROP=OldVar1 OldVar2);
   BY something;    (KEEP=GoodVar1-GoodVarX);
RUN;
```

Why?

Range? Or List out?

Sorting can take a lot of disk space!

1) Only bring along the variables you need!

1a) Aside: You probably don't want to do this …

```
1214   PROC SORT DATA=TEMP
1215             OUT=T3(DROP = RandomBool);
1216       BY  RandomBool  RandomPosNegMiss ;
1217   RUN;

NOTE: Input data set is already sorted; it has been
      copied to the output data set.
NOTE: There were 250 observations read from the
      data set WORK.TEMP.
NOTE: The data set WORK.T3 has 250 observations and
      5 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.76 seconds
      cpu time            0.03 seconds
```

Space Concerns

Looking at PROC CONTENTS …

```
The CONTENTS Procedure

Data Set Name WORK.T3                Observations          250
Member Type   DATA                   Variables             5
Engine        V9                     Indexes               0
Created       Tuesday, March 17,     Observation Length    336
Last Modified Tuesday, March 17,     Deleted Observations  0
Protection                           Compressed            NO
Data Set Type                        Sorted                NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Wi
       …      …      …      …      …      …
```

NOT SORTED!!

```
real time            0.76 seconds
cpu time             0.03 seconds
```

Looking at PROC CONTENTS …

```
The CONTENTS

Data Set Nam                                          250
Member Type                                           5
Engine                                                0
Created                                   gth         336
Last Modifie                              tions       0
Protection                                            NO
Data Set Typ                                          NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Wi
      …      …      …      …      …      …

      real time            0.76 seconds
      cpu time             0.03 seconds
```

For the record, if we had removed the secondary sort variable instead of the primary one, SAS would continue to recognize that the dataset was sorted by that variable.

NOT SORTED!!

Sorting can take a lot of disk space!

1) Only bring along the variables you need!

1b) Aside: You *definitely* don't want to do this …

```
1224   PROC SORT DATA=TEMP(DROP = RandomBool)
1225             OUT=T4 ;
1226      BY  RandomBool  RandomPosNegMiss ;
ERROR: Variable RANDOMBOOL not found
1227   RUN;

NOTE: The SAS System stopped processing this step
      because of errors.
WARNING: The data set WORK.T4 may be incomplete.
         When this step was stopped there were
         0 observations and 0 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.04 seconds
      cpu time            0.00 seconds
```

Sorting can take a lot of disk space!

2) Only bring along the observations you need!

```
DATA Temp2;
   SET Temp;
   WHERE ThreeLetterUpOnly<="P" ;
RUN;
```

```
PROC SORT DATA=Temp (WHERE=( ThreeLetterUpOnly<="P")); 
          OUT=Temp2;
   BY something;
RUN;
```

Same deal … put your constraints on the input data, not the output!

Sorting can take a lot of disk space!
3) SAS will watch your back for you! (Somewhat.)

```
1137  PROC SORT DATA=TEMP;
1138      BY  RandomBool  RandomPosNegMiss ;
1139  RUN;

NOTE: There were 250 observations read from
      the data set WORK.TEMP.
NOTE: The data set WORK.TEMP has 250 observations
      and 6 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.09 seconds
      cpu time            0.03 seconds
```

Sorting can take a lot of disk space!
3) SAS will watch your back for you! (Somewhat.)

```
1145  PROC SORT DATA=TEMP(WHERE=(ThreeLetterUpOnly<="P"))
1146            OUT=TEMP2;
1147      BY  RandomBool  RandomPosNegMiss ;
1148     /*** replace with other variables as needed ***/
1149  RUN;

NOTE: Input data set is already sorted; it has been copied
      to the output data set.
NOTE: There were 141 observations read from the data set
      WORK.TEMP.
      WHERE ThreeLetterUpOnly<='P';
NOTE: The data set WORK.TEMP2 has 141 observations and
      6 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.06 seconds
      cpu time            0.01 seconds
```

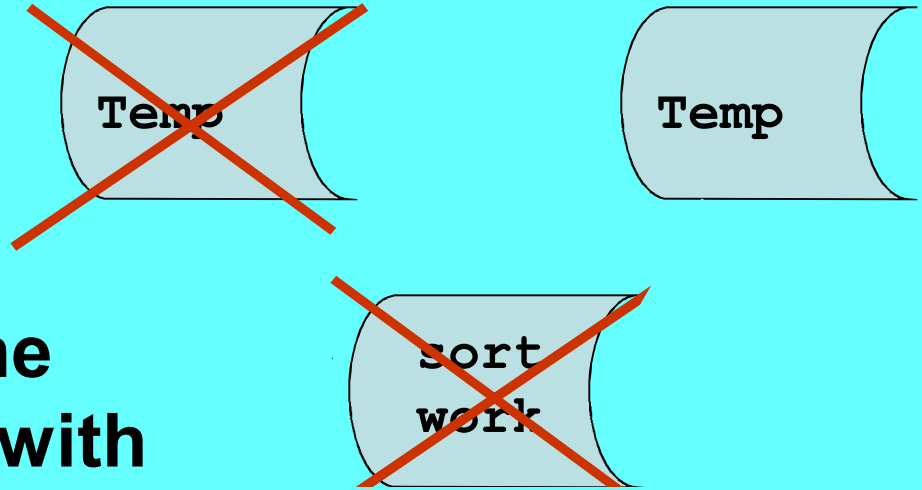
Looking at PROC CONTENTS …

```
The CONTENTS Procedure

Data Set Name WORK.TEMP          Observations          250
Member Type   DATA               Variables             6
Engine        V9                 Indexes               0
Created       Tuesday, March 17, Observation Length    344
Last Modified Tuesday, March 17, Deleted Observations  0
Protection                       Compressed            NO
Data Set Type                    Sorted                YES
        …      …      …      …      …      …
Sort Information

Sortedby       RandomBool RandomPosNegMiss
Validated      YES
Character Set  ANSI
```

Sorting can take a lot of disk space!

5) What if we overwrite the original dataset?

```
PROC SORT DATA=Temp OVERWRITE;
   BY something;
RUN;
```



Question: What is the biggest risk of sorting with this option?

Take a back-up first! (Which, of course, requires more space …)

Selective Record Retention and Tiebreakers

Sorting can take a lot of disk space!

6) What if we don't bring over all the records!

PROC SORT has options that will remove duplicate records from the output dataset.

NODUPRECS (a.k.a. NODUP)
NODUPKEY

Selective Record Retention and Tiebreakers

Sorting can take a lot of disk space!

6) What if we don't bring over all the records!

NODUPRECS

Only keeps one occurrence if every variable on consecutive observations has the same value.

(*Aside: I normally use the* NODUP *alias, but the full word is more explanatory.*)

Selective Record Retention and Tiebreakers

Sorting can ta

6) What if we dor

NODUPRECS

The manual reminds us that the only way to be absolutely certain that every variable matches the previous one is to sort by every variable in the file.

Does that sound like overkill to anyone else???

Only keeps one occurrence if every variable on consecutive observations has the same value.

(*Aside: I normally use the* NODUP *alias, but the full word is more explanatory.*)

Selective Record Retention and Tiebreakers

Sorting can take a lot of disk space!

6) What if we don't bring over all the records!

NODUPKEY~~S~~

Only keeps first observation if every variable in the sort sequence on consecutive observations has the same value.

(***Aside: Which keyword is plural and which one is not? Another good reason to code "NODUP" instead of "NODUPRECS"!***)

Selective Record Retention and Tiebreakers

Sorting can take a lot of disk space!

6) What if we don't bring over all the records!

NO

Onl y
vari
con same
value.

WARNING: Quoth the manual:

If you use the VMS operating environment sort, then the observation that is written to the output data set is not always the first observation of the BY group.

(*Aside: Which keyword is plural and which one is not? Another good reason to code "*NODUP*" instead of "*NODUPRECS*"!*)

Selective Record Retention and Tiebreakers

Sorting can take a lot of disk space!

6) What if we don't bring over all the records!

DUPOUT=

Specifies the dataset to which duplicate observations are written.

(*Again, quoting the manual.)*

Selective Record Retention and Tiebreakers

What if I have two or more observations with the same values for their sort variables?

EQUALS and NOEQUALS

EQUALS (the default) maintains the order in which the observations were brought in.

NOEQUALS does not.
Or to be more precise … *might* not.

Selective Record Retention and Tiebreakers

What if I have two or more observations with the same values for their sort variables?

EQUALS and NOEQUALS

EQUALS is usually the preferred alternative.
(which is probably why it is the default!)

NOEQUALS will save processing time and CPU.

Selective Record Retention and Tiebreakers

What if I have two or more observations with the same values for their sort variables?

EQU

EQU

(which is probably why it is the default!)

NOEQUALS will save processing time and CPU.

Your choice between these two options could have a significant effect on which observations are passed along / dropped when using NODUP and NODUPKEY.

Selective Record Retention and Tiebreakers

What if I have two or more observations with the same values for their sort variables?

Your choice between these two options affects the order in which the observations are processed.

EQUALS is the default.

EQUALS is usually the fastest way.

(which is probably why it is the default!)

NOEQUALS will save processing time and CPU.

Use of this keyword overrides the system option SORTEQUALS / SORTNOEQUALS.

Selective Record Retention and Tiebreakers

What if I ha more observations with the same sort variables?

EQU eff

EQU

(which is probably w !)

NOEQUALS will save processing and CPU.

Another alternative: Randomization! Add a random variable to your dataset, and make it the most granular variable of your sort sequence!

Alternatives to Sorting

Sorting is expensive.

(Well, relatively so.)

Think about why you want to sort, and if there are any alternatives.

(And then, are those alternatives BETTER alternatives for you?)

BY vs. CLASS

MERGE vs. JOIN

Indexes & Hashing

Indexing and Hashing do not really fall into the topic of sorting.

At least not until you expand the topic to start thinking about just WHY you want to perform a sort in the first place!

Indexing and Hashing do not really fall into the topic of sorting.

If you want to be able to reference your data in a particular order, or you want to merge two datasets together, or … perhaps you don't need to sort after all!

Indexing and Hashing do not really fall into the topic of sorting.

I refer you to one of the many fine papers presented over the years to gain more information on these topics.

System Options

Due to time constraints, we will not be talking about System Options related to sorting (other than what has already been mentioned).

Please refer to the manual – *ESPECIALLY* the SAS Companion for the Operating System of your choice!

Conclusion

We talked about a lot of aspects of sorted data.

(Probably not all of them, but enough for one session.)

(Probably more than you thought there were, too!)

If you only take one thing out of this session …

Read the manual.

Then, RE-READ the manual on occasion; don't just assume you know it!

(a) You might have forgotten something.

(b) They might have added something in the current release – or the one before that!

For further information …

The author can be reached at :

KuligowskiConference@gmail.com

Conclusion

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

My lawyer