Looking Beneath the Surface of Sorting

Andrew T. Kuligowski
FCCI Insurance Group
Presented at the Toronto Area SAS Society
05 March 2010
Introduction

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

Looking Beneath the Surface of Sorting
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 \(\text{Variable1} < \text{Variable2} \ldots >\);
RUN;
A *slightly* more complicated sort:

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

```plaintext
PROC SORT DATA=<DatasetName>
   OUT=<OutputDatasetName>;
   BY <DESCENDING> Variable1
      <<DESCENDING> Variable2 ...>;
RUN;
```
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!)
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”?  

Numerical Order  
(A refresher from elementary school …)
Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>Orig</th>
<th>RandomPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Order</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orig</th>
<th>RandomPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Order</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>78</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
</tr>
</tbody>
</table>

Looking Beneath the Surface of Sorting
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.

<table>
<thead>
<tr>
<th>Orig</th>
<th>RandomPos</th>
<th>Obs</th>
<th>Order</th>
<th>NegMiss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>19</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>27</td>
<td>-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>85</td>
<td>_</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Looking Beneath the Surface of Sorting
**What is “numerical order”?**

<table>
<thead>
<tr>
<th>Orig RandomPos</th>
<th>Obs Order</th>
<th>NegMiss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85</td>
<td>_</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>M</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>-4</td>
</tr>
<tr>
<td>6</td>
<td>19</td>
<td>-3</td>
</tr>
</tbody>
</table>

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

Looking Beneath the Surface of Sorting
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”?)

# @ % * + < >

Looking Beneath the Surface of Sorting
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 …”

Looking Beneath the Surface of Sorting
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 …"

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 Alphabetic 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

Looking Beneath the Surface of Sorting
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 / Variations in Operating Systems

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 { } ~

Looking Beneath the Surface of Sorting
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 / Variations in 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 !!!

Looking Beneath the Surface of Sorting
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!
### ASCII vs. EBCDIC

#### Sorting Alphanumeric Characters / Variations in Operating Systems

Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>PROC SORT DATA=AscEbc</th>
<th>PROC SORT DATA=AscEbc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASCII</strong></td>
<td><strong>EBCDIC</strong></td>
</tr>
<tr>
<td>BY AByte;</td>
<td>BY OneByte;</td>
</tr>
<tr>
<td>RUN;</td>
<td>RUN;</td>
</tr>
<tr>
<td>Obs</td>
<td>AByte</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>#</td>
</tr>
<tr>
<td>3</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>,</td>
</tr>
<tr>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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
## ASCII vs. EBCDIC

### Sorting Alphanumeric Characters / Variations in Operating Systems

Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>Obs</th>
<th>AByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>f</td>
</tr>
<tr>
<td>3</td>
<td>[</td>
</tr>
<tr>
<td>4</td>
<td>\</td>
</tr>
<tr>
<td>5</td>
<td>{</td>
</tr>
<tr>
<td>6</td>
<td>,</td>
</tr>
<tr>
<td>7</td>
<td>_</td>
</tr>
<tr>
<td>8</td>
<td>&gt;</td>
</tr>
<tr>
<td>9</td>
<td>#</td>
</tr>
<tr>
<td>10</td>
<td>&lt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>AByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;</td>
</tr>
<tr>
<td>2</td>
<td>@</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>\</td>
</tr>
<tr>
<td>6</td>
<td>_</td>
</tr>
<tr>
<td>7</td>
<td>a</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>{</td>
</tr>
<tr>
<td>10</td>
<td>}</td>
</tr>
</tbody>
</table>

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

Looking Beneath the Surface of Sorting

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
- ITALIAN
- POLISH (ver 9.2)
- FINNISH
- DANISH
- NORWEGIAN
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>;
SORTSEQ=SWEDISH;
BY <DESCENDING> Variable1 <<DESCENDING> Variable2 …;
RUN;

Europeans, Central and South Americans, … or SPANISH or NORWEGIAN or DANISH or FINNISH looking beneath the surface of sorting.

How did Scandinavia get first dibs? I suspect … they offered the Powers-That-Be at SAS something that no one else could …
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>
sortseq=swedish;
by <descending> variable1 <descending> variable2 …;
run;
```

Europeans, Central and South Americans, … or SPANISH or NORWEGIAN or DANISH or FINNISH

Sorting Alphanumeric Characters / Variations in Operating Systems

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!)

Looking Beneath the Surface of Sorting
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;
  SORTSEQ=SWEDISH;
  BY <DESCENDING> Variable1 <<DESCENDING> Variable2 …>
RUN;
```

Europeans, Central and South Americans, … or SPANISH or NORWEGIAN or DANISH or FINNISH (ver 9.2)

Sorting Alphanumeric Characters / Variations in Operating Systems

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!)

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?)
```

Collating Sequences

Looking Beneath the Surface of Sorting
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.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Order</th>
<th>Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>MZX</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>MJv</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>mVX</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>MPY</td>
</tr>
<tr>
<td>62</td>
<td>62</td>
<td>mCA</td>
</tr>
<tr>
<td>140</td>
<td>140</td>
<td>MfE</td>
</tr>
<tr>
<td>143</td>
<td>143</td>
<td>MWz</td>
</tr>
<tr>
<td>159</td>
<td>159</td>
<td>Mda</td>
</tr>
<tr>
<td>187</td>
<td>187</td>
<td>MEq</td>
</tr>
<tr>
<td>191</td>
<td>191</td>
<td>moU</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Three Orig Letter</th>
<th>Obs</th>
<th>Order</th>
<th>Mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MZx</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MJv</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>mVX</td>
<td>42</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>MPY</td>
<td>44</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>mCA</td>
<td>62</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>MfE</td>
<td>140</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>MWz</td>
<td>143</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>Mda</td>
<td>159</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>MEq</td>
<td>187</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>moU</td>
<td>191</td>
<td>191</td>
<td></td>
</tr>
</tbody>
</table>

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

```sas
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;
```

Looking Beneath the Surface of Sorting
Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Order</th>
<th>Mixed</th>
<th>Letter</th>
<th>Caps</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>MZx</td>
<td>MZX</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>MJv</td>
<td>MJV</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>mVX</td>
<td>MVX</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>MPY</td>
<td>MPY</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>62</td>
<td>mCA</td>
<td>MCA</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>140</td>
<td>MfE</td>
<td>MFE</td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>143</td>
<td>MWz</td>
<td>MWZ</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>159</td>
<td>Mda</td>
<td>MDA</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>187</td>
<td>MEq</td>
<td>MEQ</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>191</td>
<td>moU</td>
<td>MOU</td>
<td></td>
</tr>
</tbody>
</table>

Looking Beneath the Surface of Sorting
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.

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

PROC SORT DATA=Temp2 SampData SORTSEQ=linguistic,
  BY ThreeLetterCaps ThreeLetterMixed;
RUN;

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

Looking Beneath the Surface of Sorting
Let’s find out via experimentation.

```sas
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

Looking Beneath the Surface of Sorting
Let's find out via experimentation.

Looking Beneath the Surface of Sorting
Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Order</th>
<th>ThreeLetterMixed</th>
<th>ThreeLetterCaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>mCA</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>Mda</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>42</td>
<td>MEq</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>44</td>
<td>MfE</td>
</tr>
<tr>
<td>62</td>
<td>62</td>
<td>62</td>
<td>MJv</td>
</tr>
<tr>
<td>140</td>
<td>140</td>
<td>140</td>
<td>moU</td>
</tr>
<tr>
<td>143</td>
<td>143</td>
<td>143</td>
<td>MPY</td>
</tr>
<tr>
<td>159</td>
<td>159</td>
<td>159</td>
<td>mVX</td>
</tr>
<tr>
<td>187</td>
<td>187</td>
<td>187</td>
<td>MWz</td>
</tr>
<tr>
<td>191</td>
<td>191</td>
<td>191</td>
<td>MZx</td>
</tr>
</tbody>
</table>

Looking Beneath the Surface of Sorting

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

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>14</th>
<th>43</th>
<th>4</th>
<th>1</th>
<th>27</th>
</tr>
</thead>
</table>

(if they’re stored as numbers)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>14</th>
<th>27</th>
<th>43</th>
</tr>
</thead>
</table>

Looking Beneath the Surface of Sorting
Let’s find out via experimentation.

<table>
<thead>
<tr>
<th>Obs</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1411 12th Street</td>
</tr>
<tr>
<td>2</td>
<td>14 121st Street</td>
</tr>
<tr>
<td>3</td>
<td>141 12th Street Apt 2</td>
</tr>
<tr>
<td>4</td>
<td>14 12th Street</td>
</tr>
<tr>
<td>5</td>
<td>141 121st Street</td>
</tr>
<tr>
<td>6</td>
<td>141 12th Street Apt 11</td>
</tr>
<tr>
<td>7</td>
<td>141 12th Street Apt 1</td>
</tr>
</tbody>
</table>

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

Looking Beneath the Surface of Sorting
Let’s find out via experimentation.

Let’s find out via experimentation.

Numbers inside character data

Sorting Alphanumeric Characters / Variations in Operating Systems

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

<table>
<thead>
<tr>
<th>Obs</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1411 12th Street</td>
</tr>
<tr>
<td>2</td>
<td>14 121st Street</td>
</tr>
<tr>
<td>3</td>
<td>141 12th Street</td>
</tr>
<tr>
<td>4</td>
<td>14 12th Street</td>
</tr>
<tr>
<td>5</td>
<td>141 121st Street</td>
</tr>
<tr>
<td>6</td>
<td>141 12th Street</td>
</tr>
<tr>
<td>7</td>
<td>1411 12th Street</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14 12th Street</td>
</tr>
<tr>
<td>2</td>
<td>14 121st Street</td>
</tr>
<tr>
<td>3</td>
<td>141 12th Street</td>
</tr>
<tr>
<td>4</td>
<td>141 12th Street</td>
</tr>
<tr>
<td>5</td>
<td>141 12th Street</td>
</tr>
<tr>
<td>6</td>
<td>141 121st Street</td>
</tr>
<tr>
<td>7</td>
<td>1411 12th Street</td>
</tr>
</tbody>
</table>
Sorting can take a lot of disk space!

Aside: Picture a simple DATA Step:

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

DATA Temp;
  SET Temp;
RUN;
```

Looking Beneath the Surface of Sorting
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?
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;
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

Looking Beneath the Surface of Sorting
Looking at PROC CONTENTS ...

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>WORK.T3</th>
<th>Observations</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>5</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>Tuesday, March 17,</td>
<td>Observation Length</td>
<td>336</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Tuesday, March 17,</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
<td>Sorted</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>wlatin1 Western (Wi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Looking Beneath the Surface of Sorting
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

Looking at PROC CONTENTS …

<table>
<thead>
<tr>
<th>The CONTENTS Procedure</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>WORK.T3</td>
</tr>
<tr>
<td>Member Type</td>
<td>DATA</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
</tr>
<tr>
<td>Created</td>
<td>Tuesday, March 17</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Tuesday, March 17</td>
</tr>
<tr>
<td>Protection</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td>WINDOWS_32</td>
</tr>
<tr>
<td>Encoding</td>
<td>wlatin1 Western (Windows)</td>
</tr>
<tr>
<td>... ... ... ... ... ... ... ... ... ... ... ... ...</td>
<td></td>
</tr>
</tbody>
</table>

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!!

Looking Beneath the Surface of Sorting
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!

```sas
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

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>WORK.TEMP</th>
<th>Observations</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>6</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>Tuesday, March 17,</td>
<td>Observation Length</td>
<td>344</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Tuesday, March 17,</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td></td>
<td>Compressed</td>
<td>NO</td>
</tr>
<tr>
<td>Data Set Type</td>
<td></td>
<td>Sorted</td>
<td>YES</td>
</tr>
</tbody>
</table>

Sort Information

<table>
<thead>
<tr>
<th>Sortedby</th>
<th>RandomBool RandomPosNegMiss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validated</td>
<td>YES</td>
</tr>
<tr>
<td>Character Set</td>
<td>ANSI</td>
</tr>
</tbody>
</table>
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 …)
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
Selecting 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.)
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 take a lot of disk space!
6) What if we don’t bring over all the records!

NODUPKEYS

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!

NODUPKEYS
Only keeps first observation if every variable in the sort sequence on consecutive observations has the 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”!)
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.)
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.
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.
What if I have two or more observations with the same values for their sort variables?

**EQUALS** is usually the preferred alternative. (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.
What if I have two or more observations with the same values for their sort variables? 

Your choice between these two options will determine whether EQUI 

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

NOEQUALS will save processing time and CPU.
What if I have two or more observations with the same values for their sort variables? EQUALS is usually the preferred alternative. NOEQUALS will save processing time and CPU. (which is probably why it is the default!)

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

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

Use of this keyword overrides the system option SORTEQUALS / SORTNOEQUALS.
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?)
Alternatives to Sorting

BY vs. CLASS

MERGE vs. JOIN

Indexes & Hashing

Looking Beneath the Surface of Sorting
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.
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!
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
SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries. \(^{(R)}\) indicates USA registration.