PAPER SAS4526-2020

ODS: It’s Not Just for Tables Anymore. Using Formatted Text and Lists in Your Reports

Scott Huntley, SAS Institute Inc.

ABSTRACT

You’re used to seeing tables in your ODS reports, but maybe you need that extra something to drive your results home. Did you know that you can add formatted text and lists to your report? With PROC ODSLIST and PROC ODSTEXT, you can add richer descriptions to your reports, or even generate data-driven lists and blocks of text.

The examples in this paper demonstrate several ways to display your output. Whether you need to create a form letter or an infographic, PROC ODSLIST and PROC ODSTEXT can help you generate the report that you want.

INTRODUCTION

The Output Delivery System (ODS) provides several ways for you to create the perfect way to display your SAS output. Traditionally, ODS output contains tables and graphics.

This paper covers two new SAS procedures in first maintenance release of SAS 9.4: PROC ODSLIST and PROC ODSTEXT. PROC ODSLIST creates bulleted lists. PROC ODSTEXT creates text blocks like the statement ODS TEXT= . Both procedures can be used with any ODS destination; however, these procedures are essential to creating content for the ODS destinations PowerPoint and EPUB.

The intended audience is anyone who has a basic understanding of the ODS system. This paper provides code samples to assist users in adding these new features to their existing SAS programs.

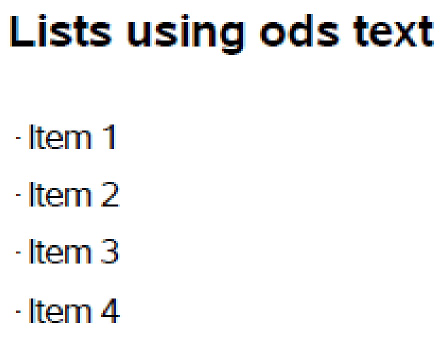
CREATING BULLETED LISTS

Creating a bulleted list is a great way to delineate information. It also helps break up long text strings into readable bits of text. In earlier versions of SAS, you could use titles, footnotes, or ODS TEXT to create a static bulleted list. Here’s an example of using ODS TEXT:

```
options nodate nonumber nocenter;
ods escapechar="^";
ods pdf;
title 'Lists using ods text';
ods text = '^{unicode 00B7} Item 1';
ods text = '^{unicode 00B7} Item 2';
ods text = '^{unicode 00B7} Item 3';
ods text = '^{unicode 00B7} Item 4';
ods _all_ close;
```

In the code above, the ODS inline style Unicode function is used to produce the dot for the list. If you want to indent or add spaces to the bulleted list, you must use the ODS inline style nbspace to adjust the spacing. It's a very manual process and it produces a static list.

Running the code produces this output:



Lists using ods text

· Item 1

· Item 2

· Item 3

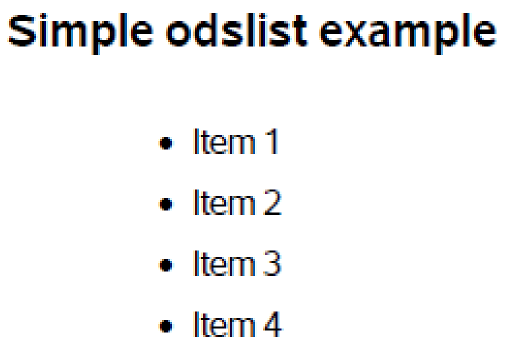
· Item 4

Figure 1. Lists using ODS Text

Here is an example using PROC ODSLIST to create the same bulleted list. The syntax is very simple. Each ITEM statement specifies what gets displayed in the bulleted list.

```
title 'Simple odslist example';
ods pdf;
proc odslist;
   item 'Item 1';
   item 'Item 2';
   item 'Item 3';
   item 'Item 4';
end;
run;
ods _all_ close;
```

The output looks similar, but the results are improved:



Simple odslist example

- Item 1
- Item 2
- Item 3
- Item 4

Figure 2. List with PROC ODSLIST

The bulleted list is indented automatically. The default character to signify the list is the disc. You can easily modify the default character if desired by using the style attribute liststyletype. This paper discusses how to change the default character in the 'Style Adjustments' section.

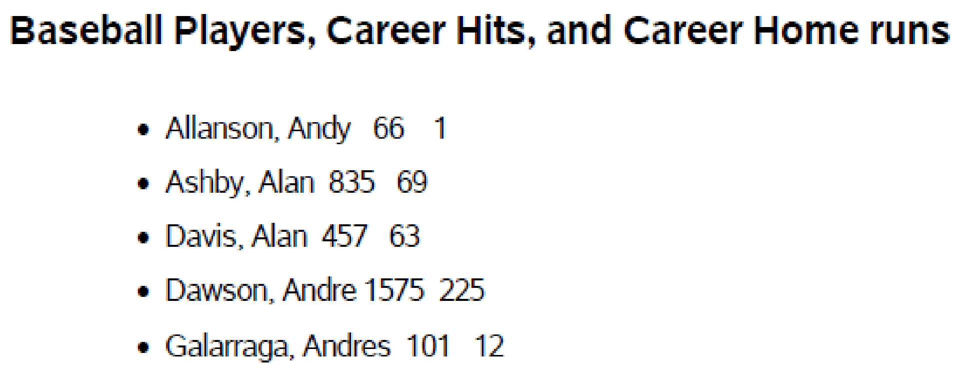
```
 proc odslist;

    item;
    list / style={liststyletype="decimal"};
       item 'Item 1’;
```

SAS can do much more than create simple static bulleted lists with PROC ODSLIST. The ITEM statement in the procedure is a real differentiator. The ITEM statement is a SAS expression, so it can contain both variable names and DATASET functions. You can read from a data set and produce bulleted lists from that data set. The PROC ODSLIST syntax supports the DATA= option. In this example, each player's name, career hits, and home runs are listed out from the SASHELP.BASEBALL data set for the first 5 observations:

```
ods pdf;
title 'Baseball Players, Career Hits, and Career Home runs';
proc odslist data=sashelp.baseball(obs=5);
   item Name || put (CrHits, 5.) || put (CrHome, 5.);
run;
ods _all_ close;
```

Running the code produces this output:



Baseball Players, Career Hits, and Career Home runs

- Allanson, Andy   66   1
- Ashby, Alan 835   69
- Davis, Alan 457   63
- Dawson, Andre 1575 225
- Galarraga, Andres 101   12

Figure 3. List from dataset variables

Notice the use of only one ITEM statement with a complex SAS expression. Each observation of the data set is read and produces one line of output in the PDF file. The variables Name, CrHits, and Crhome are displayed. The numeric variables are formatted to be a numeric with a length of five digits with no decimals.

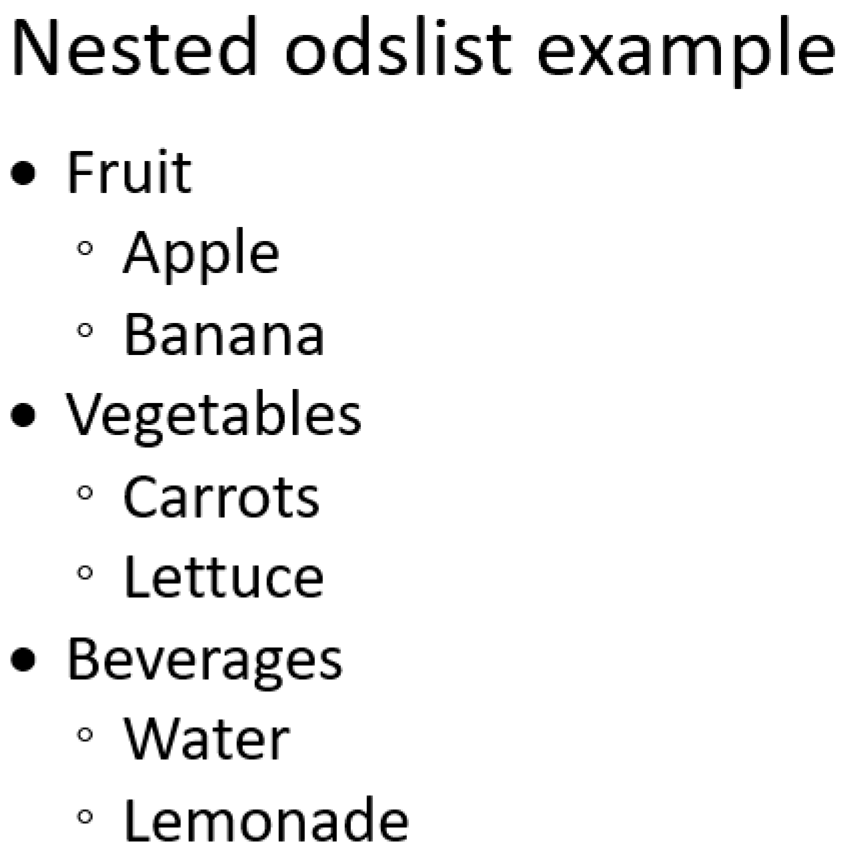
NESTED LISTS

You can further expand the lists even more by introducing nested lists. From the previous example, you can see that an expression was used in the ITEM statement comprised of dataset variables and PUT functions. That means each observation is produced individually. To create a nested list an ITEM/END block of SAS code is needed. There is one restriction on the ITEM statement: ITEM statements that do not specify an expression must end with an END statement. More SAS statements are needed inside the ITEM block. The first statement you need is the P statement. The P statement specifies that a new paragraph is created inside an ITEM block. Multiple P statements can be used in an ITEM block. After each P statement, there can exist a LIST/END block that contains ITEM statements that generate a bulleted list.

Here is a quick example using the P and LIST statements:

```
title 'Nested odslist example';
ods powerpoint;
proc odslist;
item;
   p 'Fruit';
   list;
      item 'Apple';
      item 'Banana';
   end;
end;
item;
   p 'Vegetables';
   list;
      item 'Carrots';
      item 'Lettuce';
   end;
end;
item;
   p 'Beverages';
   list;
     item 'Water';
     item 'Lemonade';
   end;
end;
run;
ods _all_ close;
```

Running the code produces this output:



Nested odslist example

- Fruit
  - Apple
  - Banana
- Vegetables
  - Carrots
  - Lettuce
- Beverages
  - Water
  - Lemonade

Figure 4. Nested List

To recap how the code works, the ITEM/END block creates the outside list. The P statement indicates that item's content. The LIST/END block then creates the inside list to complete the nested list.

STYLE ADJUSTMENTS

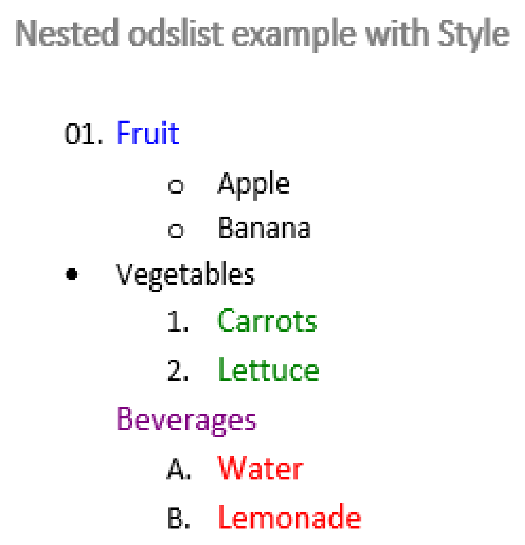
ODS is all about presentation. A big part of the presentation comes from using STYLE and STYLE attributes. Style attribute changes, including color or font size, are achieved the same way as with any other ODS code. PROC ODSLIST can easily use styles to make your lists really standout. A standout for PROC ODSLIST is the style attribute liststyletype. This attribute allows you to change the style of the list indicator. The SAS documentation has an entire section discussing which style attributes are available to which destinations. Here is the code from the nested list example modified to include some basic styles attribute changes.

Here is a quick example of modifying the results with styles using the ODS WORD destination:

```
title 'Nested odslist example with Style';
ods word;

proc odslist;
item / style={liststyletype="decimal_leading_zero”
              fontsize=12pt
              color=blue};
   p 'Fruit';
      list;
         item 'Apple';
         item 'Banana';
      end;
end;
item;
   p 'Vegetables';
      list / style={liststyletype="decimal" fontsize=12pt color=green};
         item 'Carrots';
         item 'Lettuce';
      end;
end;
item / style={liststyletype="none" fontsize=12pt color=purple};
   p 'Beverages';
      list / style={liststyletype="upper_alpha” fontsize=12pt color=red};
         item 'Water';
         item 'Lemonade';
      end;
end;
run;
ods _all_ close;
```

Running the code produces this output:



```
Nested odslist example with Style

01. Fruit
        o  Apple
        o  Banana
 •  Vegetables
        1.  Carrots
        2.  Lettuce
    Beverages
        A.  Water
        B.  Lemonade
```

Figure 5. Nested List with Styles

Notice that six different types of the liststyletype attribute (decimal_leading_zero, circle, disc, decimal, none, upper_alpha) are used in this example. Each of these style attribute types are set at different levels to help show how flexible it can be. The first ITEM/END block uses the DECIMAL_LEADING_ZERO type for its list indicator. The font size and color attributes for the paragraph text are also used. The circle indicator is used for the nested list under the "Fruit" heading. In the next ITEM/END block for "Vegetables," the style attribute is not set for the indicator, so the default disc indicator for the ITEM block is used. The liststyletype attribute for the list items under "Vegetables" is set to decimal and the font color is set to green. On the last ITEM/END block the style attribute is set to none. Therefore, there is no bullet for "Beverages" and the text color is set to purple. The nested LIST/END block has the color set to red and uses style attribute value, "UPPER_ALPHA," as the indicator.

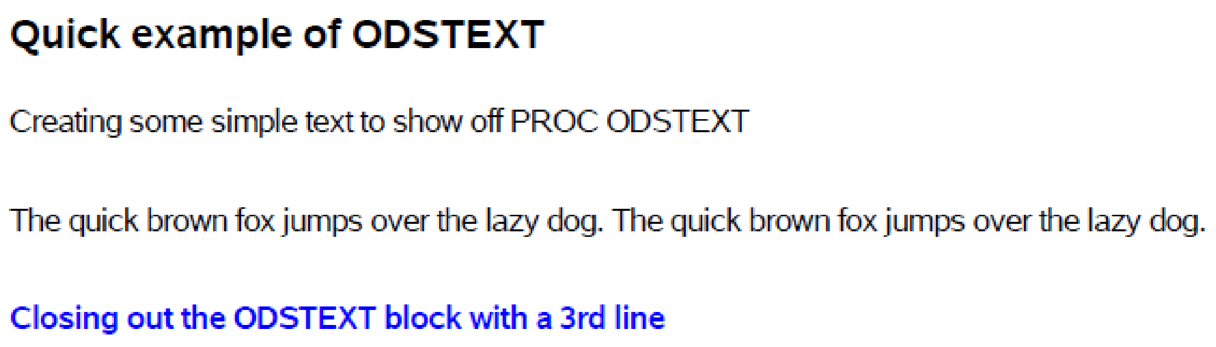
CREATING BLOCKS OF TEXT

Most output created for customer reports has text included to embellish the tables and graphs that are created by SAS procedures. This text can help explain a graphic better or be a part of a larger report that has text embedded throughout. As mentioned earlier, most coders use titles, footnotes, and ODS TEXT to put text into their output. Using the ODS TEXT statement to put one or more text strings in your output is useful but, in my opinion, PROC ODSTEXT makes coding text statements easier. PROC ODSTEXT is used to create text blocks. These text blocks create lists and paragraphs for your output.

Here is a simple example of PROC ODSTEXT:

```
   ods pdf;
   title 'Quick example of ODSTEXT';
   proc odstext;
      p 'Creating simple text to show off PROC ODSTEXT';
      p '';
      p 'The quick brown fox jumps over the lazy dog. The quick brown fox
jumps over the lazy dog.';
      p '';
      p 'Closing out the ODSTEXT block with a 3rd line' / style=[color=blue
         fontweight=bold];
   run;
   ods _all_ close;
```

Running the code would produce this output:


Quick example of ODSTEXT

Creating some simple text to show off PROC ODSTEXT

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Closing out the ODSTEXT block with a 3rd line

Figure 6. Simple ODSTEXT example

In this PROC ODSTEXT statement example, five P statements are used to create the sample text. Notice the P statement with no text is used to create some spacing in the text block to make it more readable. The fifth P statement uses style attributes to make the text bold and blue.

Creating a paragraph of text is one piece of a text block. PROC ODSTEXT can also create bulleted lists. There are some differences in the syntax when compared to PROC ODSLIST. It's important to read the SAS documentation that explains the differences in more detail. Also, the spacing in the lists and the bullet points are slightly different as well.

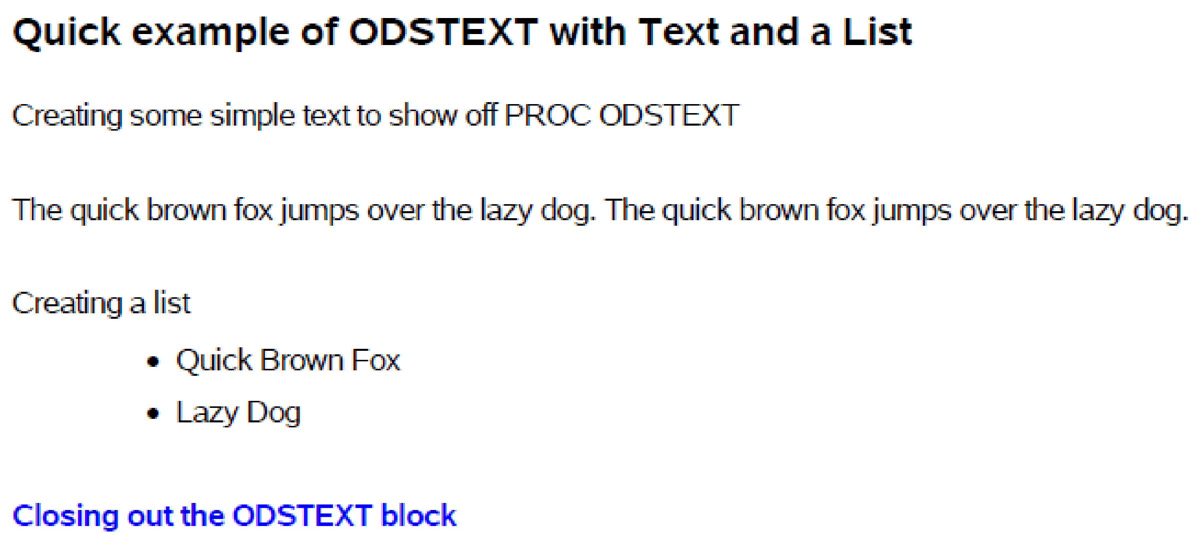
Here is an example of ODSTEXT with text and a bulleted list:

```
ods pdf;
title 'Quick example of ODSTEXT with Text and a List';
proc odstext;
   p 'Creating some simple text to show off PROC ODSTEXT';
   p '';
   p 'The quick brown fox jumps over the lazy dog. The quick brown fox
      jumps over the lazy dog.';
   p '';
   p 'Creating a list';

      list;
         item 'Quick Brown Fox';
         item 'Lazy Dog';
      end;

   p '';
   p 'Closing out the ODSTEXT block' /
         style=[color=blue fontweight=bold];
run;
ods _all_ close;
```

Running the code would produce this output:



Quick example of ODSTEXT with Text and a List

Creating some simple text to show off PROC ODSTEXT

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Creating a list

- Quick Brown Fox
- Lazy Dog

Closing out the ODSTEXT block

Figure 7. ODSTEXT with Text and a List

As mentioned above, be sure to notice the distinct syntax that creates a list within PROC ODSTEXT. The LIST/END block is used to create this list.

Similarly, PROC ODSTEXT supports the DATA= option in order to read SAS datasets. Using SASHELP.BASEBALL, you can expand on the previous example (Figure 3) that used PROC ODSLIST. You can create a form letter that reads a dataset and produces standardized text for each observation. Adding an expression using the CELLSTYLE syntax, you can use logic to make some information stand out. Notice how the players that have more than 400 home runs appear in red and the font is italicized.

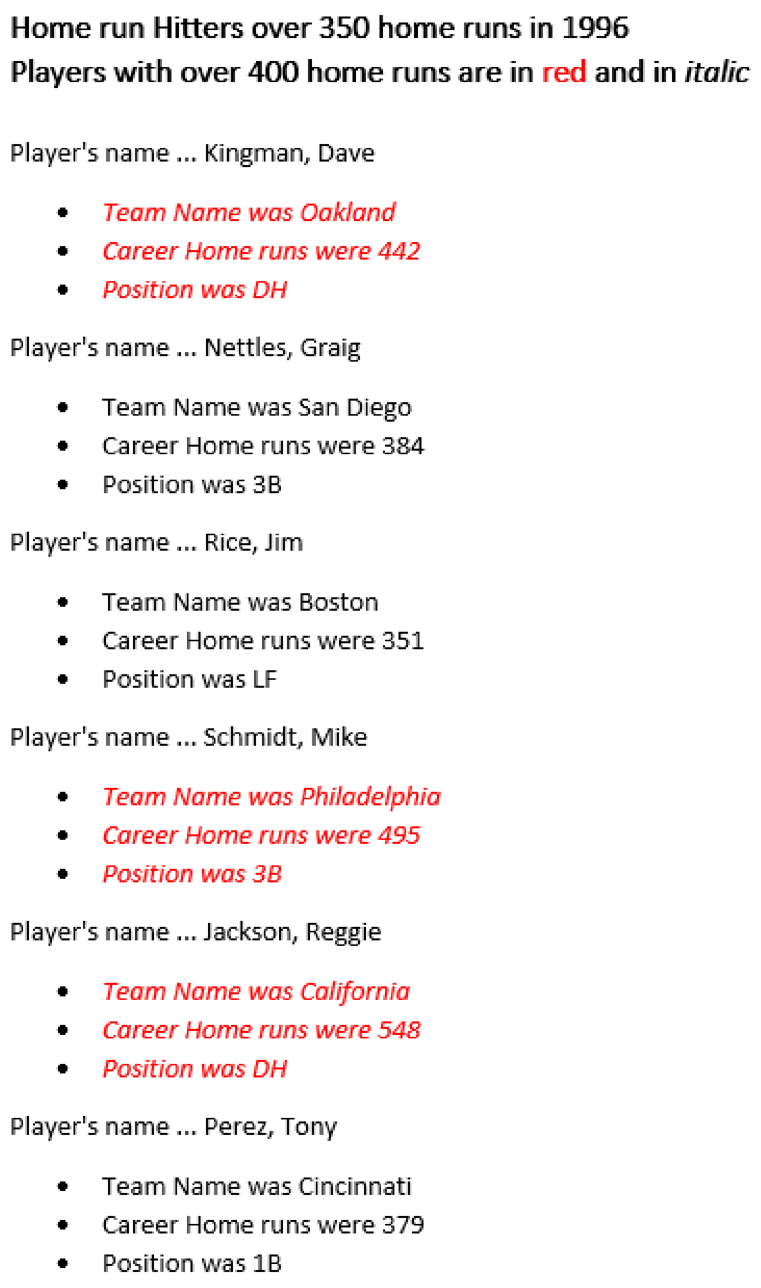
Here is an example of using PROC ODSTEXT to create paragraph text and a bulleted list:

```
ods escapechar = "^";
ods word;
title 'Home run Hitters over 350 home runs in 1996';
title2 'Players with over 400 home runs are in ^{style [color=red]red} and
        in ^{style [fontstyle=italic]italic}';

data test;
   set sashelp.baseball;
   where CrHome > 350;
run;

proc odstext data=test;
p "Player's name ... " || name;
   list;
      cellstyle Crhome > 400 as {fontstyle=italic color=red};
      item 'Team Name was ' || Team;
      item 'Career Home runs were ' || put(CrHome, 3.);
      item 'Position was ' || Position;
   end;
end;
run;
ods _all_ close;
```

Running the code produces this output:



Home run Hitters over 350 home runs in 1996
Players with over 400 home runs are in red and in *italic*

Player's name ... Kingman, Dave

- *Team Name was Oakland*
- *Career Home runs were 442*
- *Position was DH*

Player's name ... Nettles, Graig

- Team Name was San Diego
- Career Home runs were 384
- Position was 3B

Player's name ... Rice, Jim

- Team Name was Boston
- Career Home runs were 351
- Position was LF

Player's name ... Schmidt, Mike

- *Team Name was Philadelphia*
- *Career Home runs were 495*
- *Position was 3B*

Player's name ... Jackson, Reggie

- *Team Name was California*
- *Career Home runs were 548*
- *Position was DH*

Player's name ... Perez, Tony

- Team Name was Cincinnati
- Career Home runs were 379
- Position was 1B

Figure 8. ODSTEXT Reading from a SAS Dataset

The code is condensed and easy to understand. Starting with two title statements containing inline styles to describe the output, the data step code is used to filter the data by selecting players with over 350 career home runs. Then PROC ODSTEXT reads each observation in the dataset in order to display information about each player. PROC ODSTEXT is a perfect way to create a form letter from your SAS datasets.

WHICH PROCEDURE DO I USE?

The big question … How do you decide which procedure to use?

The ODSLIST procedure can only render lists where each list item corresponds to a data set row. If all you need is a quick static or bulleted list, then PROC ODSLIST is your answer.

The ODSTEXT procedure can do anything ODSLIST does and more. PROC ODSTEXT is more flexible because it can easily combine both text and lists.

Experiment with each procedure to determine which is the better solution for you.

CONCLUSION

The Output Delivery System strives to provide numerous ways to create the perfect output you need to show off your SAS work. PROC ODSTEXT and PROC ODSLIST are two new procedures to help you in that endeavor.

ACKNOWLEDGMENTS

The author would like to thank Allison Crutchfield, David Kelley, Bari Lawhorn, and Kevin Smith for their assistance and contributions to this paper.

CONTACT INFORMATION

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

Scott Huntley
SAS Institute Inc.
Scott.Huntley@sas.com