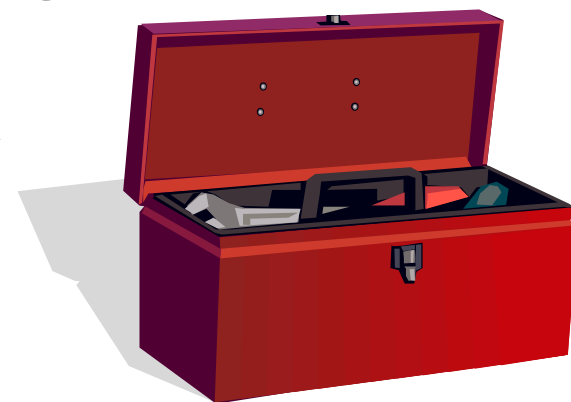


# Two Useful Macros for Your SAS<sup>®</sup> Tool Box



**Barbara A. Barrett, M.A.**  
**barrettb@bms.com**

**Bristol-Myers Squibb Company**  
**Wallingford, CT**



**PARSE.SAS**

# Purpose:

To make processing easier in a later step of a program by...

...counting the number of items in a string and assigning the value to a macro variable

...putting each item in the string into individual macro variables



# Purpose cont'd:



*For example...*

```
proc sql;  
    select distinct(lbcode) into: ex_strg  
separated by ',' from labs ;  
quit;
```

```
...ex_strg = HB,HCT,PLAT,RBC
```

# Parameters:

**STRG** - required - input string of words

**PREFX** - optional - default=v - characters to be used at start of macro variable names e.g. &v1, &v2 ...  
or if prefix=var then &var1, &var2 ...

**QUOT** - optional - y/n - default=n - quote contents of output macro variables.

**DELIM** - optional - default is blank - delimiter used on input string to separate items.



```
%MACRO PARSE(STRG=, QUOT=n, PREFIX=, DELIM=);
```

```
  %if &prefix= %then %do;  
    %let prefix=v;  
  %end;
```

***Assign default prefix if needed.***

```
data _null_;  
  %global &prefix.n;  
  i=0 ;  
  do until ( x = " );  
    i=i+1;  
    x=scan("&strg",i,"&delim");  
  end ;  
  call symput("&prefix.n",i-1);  
run;
```

***Count number of items in string.***

```
data _null_ ;  
  %do j=1 %to &&prefix.n;  
    %global &prefix&j ;  
    %if %upcase("&quot;)=Y %then %do;  
      call symput("&prefix&j",quote(scan("&strg",&j,"&delim")));  
    %end ;  
    %else %do;  
      call symput("&prefix&j",scan("&strg",&j,"&delim"));  
    %end;  
  %end ;  
run;
```

***Assign items to macro variables.***

```
%MEND PARSE
```

```
%MACRO PARSE(STRG=, QUOT=n, PREFIX=, DELIM=);
```

```
  %if &prefix= %then %do;  
    %let prefix=v;  
  %end;
```

***Assign default prefix if needed.***

```
data _null_ ;  
  %global &prefix.n;  
  i=0 ;  
  do until ( x = ");  
    i=i+1;  
    x=scan("&strg",i,"&delim");  
  end ;  
  call symput("&prefix.n",i-1);  
run;
```

***Count number of items in string.***

```
data _null_ ;  
  %do j=1 %to &&prefix.n;  
    %global &prefix&j ;  
    %if %upcase("&quot;)=Y %then %do;  
      call symput("&prefix&j",quote(scan("&strg",&j,"&delim")));  
    %end ;  
    %else %do;  
      call symput("&prefix&j",scan("&strg",&j,"&delim"));  
    %end;  
  %end ;  
run;
```

***Assign items to macro variables.***

```
%MEND PARSE
```

```
%MACRO PARSE(STRG=, QUOT=n, PREFIX=, DELIM=);
```

```
  %if &prefix= %then %do;  
    %let prefix=v;  
  %end;
```

*Assign default prefix if needed.*

```
data _null_;  
  %global &prefix.n;  
  i=0 ;  
  do until ( x = " );  
    i=i+1;  
    x=scan("&strg",i,"&delim");  
  end ;  
  call symput("&prefix.n",i-1);  
run;
```

*Count number of items in string.*

```
data _null_ ;  
  %do j=1 %to &&prefix.n;  
    %global &prefix&j ;  
    %if %upcase("&quot;")=Y %then %do;  
      call symput("&prefix&j",quote(scan("&strg",&j,"&delim")));  
    %end ;  
    %else %do;  
      call symput("&prefix&j",scan("&strg",&j,"&delim"));  
    %end;  
  %end ;  
run;
```

*Assign items to macro variables.*

```
%MEND PARSE
```

# Example 1: Macro Call with Default Parameters

```
%parse(strg=RED BLUE GREEN);
```



# Example 1: LOG

```
MPRINT(PARSE): data _null_;
MPRINT(PARSE): i=0;
MPRINT(PARSE): do until ( x = ");
MPRINT(PARSE): i=i+1;
MPRINT(PARSE): x=scan("RED BLUE GREEN",i,"");
MPRINT(PARSE): end;
MPRINT(PARSE): call symput("vn",i-1);
MPRINT(PARSE): run;
MPRINT(PARSE): *** ;
MPRINT(PARSE): *** ASSIGN ITEMS TO MACRO VARIABLES NAMED &PREFIX
FOLLOWED BY SEQUENCE IN STRING i.e. 1 TO &PREFIX.N ;
MPRINT(PARSE): ;
MPRINT(PARSE): *** IF &QUOTE=Y THEN QUOTE THE CONTENT OF THE MACRO
VARIABLE ;
MPRINT(PARSE): data _null_;
MPRINT(PARSE): call symput("v1",scan("RED BLUE GREEN",1,""));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): call symput("v2",scan("RED BLUE GREEN",2,""));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): call symput("v3",scan("RED BLUE GREEN",3,""));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): run;
```

# Example 1: LOG cont'd

## INPUT PARAMETERS...

... STRG=RED BLUE GREEN

... QUOT=

... PREFIX=v { *Default value for PREFIX*

... DELIM=

## OUTPUT MACRO VARIABLES...

... vn=3

... v1=RED

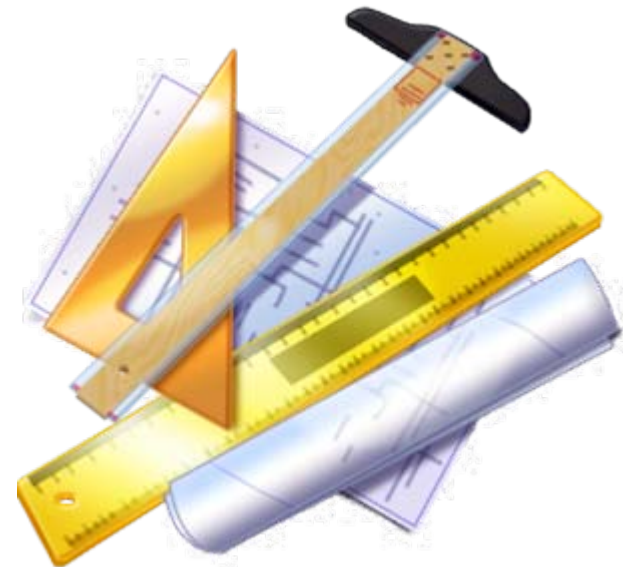
... v2=BLUE

... v3=GREEN



# Example 2: Macro Call with All Parameters Specified

```
%parse(strg=%str(RED,BLUE ,GREEN),  
      quot=y,  
      prefix=color_  
      delim=%str(,  
      );
```



# Example 2: LOG

```
MPRINT(PARSE): data _null_;
MPRINT(PARSE): i=0;
MPRINT(PARSE): do until ( x = ");
MPRINT(PARSE): i=i+1;
MPRINT(PARSE): x=scan("RED,BLUE ,GREEN",i,"");
MPRINT(PARSE): end;
MPRINT(PARSE): call symput("color_n",i-1);
MPRINT(PARSE): run;
MPRINT(PARSE): *** ;
MPRINT(PARSE): *** ASSIGN ITEMS TO MACRO VARIABLES NAMED &PREFIX FOLLOWED BY
SEQUENCE IN STRING i.e. 1 TO &PREFIX.N ;
MPRINT(PARSE): ;
MPRINT(PARSE): *** IF &QUOT=Y THEN QUOTE THE CONTENT OF THE MACRO VARIABLE ;
MPRINT(PARSE): data _null_;
MPRINT(PARSE): call symput("color_1",quote(scan("RED,BLUE ,GREEN",1,"")));
MPRINT(PARSE): *QUOTED OUTPUT;
MPRINT(PARSE): call symput("color_2",quote(scan("RED,BLUE ,GREEN",2,"")));
MPRINT(PARSE): *QUOTED OUTPUT;
MPRINT(PARSE): call symput("color_3",quote(scan("RED,BLUE ,GREEN",3,"")));
MPRINT(PARSE): *QUOTED OUTPUT;
MPRINT(PARSE): run;
```

# Example 2: LOG cont'd

INPUT PARAMETERS...

```
... STRG=RED,BLUE ,GREEN  
... QUOT=y  
... PREFIX=color_  
... DELIM=,
```

OUTPUT MACRO VARIABLES...

```
... color_n=3  
... color_1="RED"  
... color_2="BLUE "  
... color_3="GREEN"
```



# Example 3: Why Use a Delimiter?

```
%parse(  
  strg=%str(CLARET RED ^INDIGO BLUE  
  ^BOTANICAL GREEN ),  
  quot=n,  
  prefix=car ,  
  delim=%str(^)  
);
```



# Example 3: LOG

```
MPRINT(PARSE): data _null_;
MPRINT(PARSE): i=0;
MPRINT(PARSE): do until ( x = ");
MPRINT(PARSE): i=i+1;
MPRINT(PARSE): x=scan("CLARET RED ^INDIGO BLUE ^BOTANICAL GREEN ",i,"^");
MPRINT(PARSE): end;
MPRINT(PARSE): call symput("carn",i-1);
MPRINT(PARSE): run;
MPRINT(PARSE): *** ;
MPRINT(PARSE): *** ASSIGN ITEMS TO MACRO VARIABLES NAMED &PREFIX FOLLOWED BY
SEQUENCE IN STRING i.e. 1 TO &PREFIX.N ;
MPRINT(PARSE): ;
MPRINT(PARSE): *** IF &QUOTE=Y THEN QUOTE THE CONTENT OF THE MACRO VARIABLE ;
MPRINT(PARSE): data _null_;
MPRINT(PARSE): call symput("car1",scan("CLARET RED ^INDIGO BLUE ^BOTANICAL GREEN ",1,"^"));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): call symput("car2",scan("CLARET RED ^INDIGO BLUE ^BOTANICAL GREEN ",2,"^"));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): call symput("car3",scan("CLARET RED ^INDIGO BLUE ^BOTANICAL GREEN ",3,"^"));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): run;
```

# Example 3: LOG cont'd

## INPUT PARAMETERS...

```
... STRG=CLARET RED ^INDIGO BLUE  
^BOTANICAL GREEN  
... QUOT=n  
... PREFIX=car  
... DELIM=^
```

## OUTPUT MACRO VARIABLES...

```
... carn=3  
... car1=CLARET RED  
... car2=INDIGO BLUE  
... car3=BOTANICAL GREEN
```



# Example 4: Common Pitfall in Specifying STRG Parameter

```
%parse(strg=%str(RED ^BLUE ^ GREEN),  
      quot=n,  
      prefix=var,  
      delim=%str(^)  
);
```



# Example 4: LOG

```
MPRINT(PARSE): data _null_;
MPRINT(PARSE): i=0;
MPRINT(PARSE): do until ( x = ");
MPRINT(PARSE): i=i+1;
MPRINT(PARSE): x=scan("RED ^BLUE ^ GREEN",i,"^");
MPRINT(PARSE): end;
MPRINT(PARSE): call symput("varn",i-1);
MPRINT(PARSE): run;
MPRINT(PARSE): *** ;
MPRINT(PARSE): *** ASSIGN ITEMS TO MACRO VARIABLES NAMED &PREFIX FOLLOWED BY
SEQUENCE IN STRING i.e. 1 TO &PREFIX.N ;
MPRINT(PARSE): ;
MPRINT(PARSE): *** IF &QUOTE=Y THEN QUOTE THE CONTENT OF THE MACRO VARIABLE ;
MPRINT(PARSE): data _null_;
MPRINT(PARSE): call symput("var1",scan("RED ^BLUE ^ GREEN",1,"^"));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): call symput("var2",scan("RED ^BLUE ^ GREEN",2,"^"));
MPRINT(PARSE): * NON-QUOTED OUTPUT;
MPRINT(PARSE): run;
```

# Example 4: LOG cont'd

INPUT PARAMETERS...

... STRG=RED ^BLUE ^ GREEN

... QUOT=n

... PREFIX=var

...DELIM=^

OUTPUT MACRO VARIABLES...

... varn=2

... var1=RED

... var2=BLUE



# Example 4: LOG cont'd

## INPUT PARAMETERS...

... STRG=RED ^BLUE ^ GREEN

... QUOT=n

... PREFIX=var

... DELIM=



*The delimiter is not immediately before the third item, therefore the scan function reads the blank and stops.*

## OUTPUT MACRO VARIABLES...

... varn=2

... var1=RED

... var2=BLUE

# Example 2: LOG cont'd

## INPUT PARAMETERS...

... STRG=RED,BLUE ,GREEN

... QUOT=y

... PREFIX=color\_

... DELIM=,



*In this case, the delimiter is immediately before the last item.*

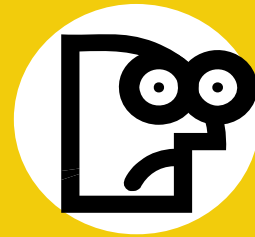
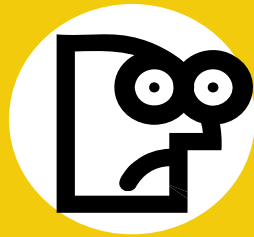
## OUTPUT MACRO VARIABLES...

... color\_n=3

... color\_1="RED"

... color\_2="BLUE "

... color\_3="GREEN"



**It's QUESTION TIME !!**

# EMPTY.SAS

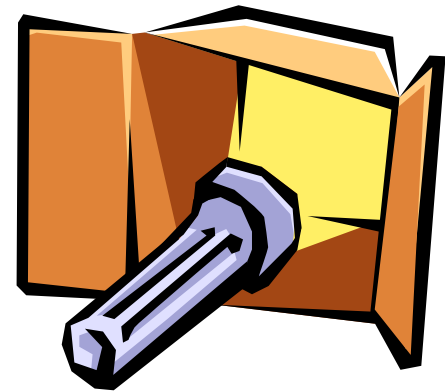


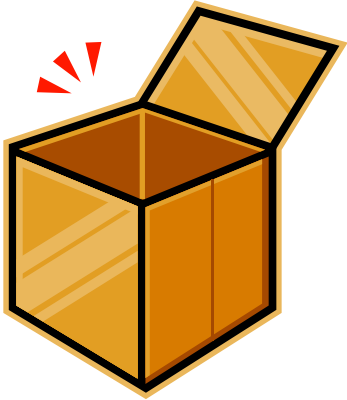
# Purpose:

To create a null report i.e. a report indicating there are no data by...

...creating macro variable NREC containing the number of observations that exist in the dataset to be reported

...writing either a default or customized message to the output file when  $NREC = 0$





# Parameters:

**DS - required - name of dataset to be analyzed**

**TEXT - optional – default='No Observations for This Report' otherwise, text as entered in macro call.**

```
%MACRO EMPTY(DS=,TEXT=);
```

```
%global nrec;  
  
proc sql noprint;  
  select count(*) into: nrec  
  from &ds  ;  
quit;
```



Count number of observations.

```
%if &nrec=0 %then %do;
```



```
  data empty;  
    %if &text= %then %do;  
      %let text=%str(No Observations for This Report);  
    %end;  
    text="&text";  
  run;
```

```
proc report data=empty headskip missing;  
  columns text;  
  define text / ' ';  
  compute before _page_;  
    line put 132*'-';  
  endcomp;  
run;  
%end;
```

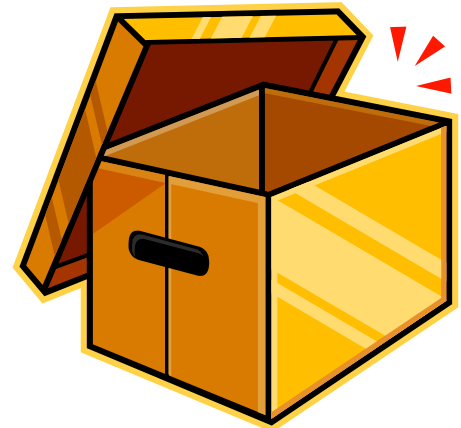


If no observations then write contents of &text to output file.

```
%MEND EMPTY;
```

# Examples 1 & 2: Dataset SH

Obs	ID
1	Superman



# Example 1: Using Default Parameters

```
%MACRO EXAMPL_1;  
data rpt;  
  set sh;  
  where id ne 'Superman';  
run;
```



```
title 'Individuals Able to Leap Tall Buildings in a Single Bound Other than Superman';  
footnote ' Example: Using Default Parameters';
```

```
%empty(ds=rpt);  
run;
```

```
%if &nrec > 0 %then %do;  
  proc report data=rpt headline headskip missing ;  
    columns id;
```

- 
- 
- 

```
%end;  
%MEND EXAMPL_1;
```



# Example 1: Output

Individuals Able to Leap Tall Buildings in a Single Bound Other than Superman

---

No Observations for This Report

---

Example: Using Default Parameters

# Examples 2 & 3: Text Parameter Specified

```
%MACRO EXAMPL_2_3;  
data rpt;  
  set sh;  
  where id ne 'Superman';  
run;
```

```
title1 'Individuals Able to Leap Tall Buildings in a Single Bound Other than  
Superman';  
footnote ' Example: Text Parameter Specified';
```

```
%empty(ds = rpt, text = %str(Superman Still Rules!) );  
run;
```

```
%if &nrec > 0 %then %do;  
  proc report data=rpt headline headskip missing ;  
    columns id;  
      ●  
      ●  
      ●  
run;  
%end;  
%MEND EXAMPL_2_3;
```



# Example 2: Output

Individuals Able to Leap Tall Buildings in a Single Bound Other than Superman

---

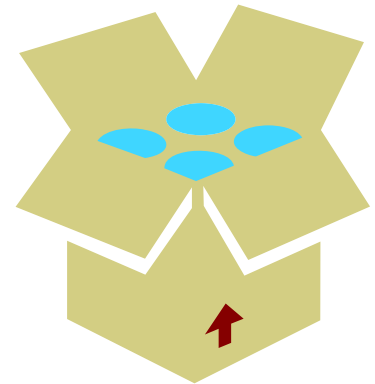
Superman Still Rules!

---

Example: Text Parameter Specified

# Example 3: Dataset

Obs	ID
1	Superman
2	Under Dog



# Example 3: Output

Individuals Able to Leap Tall Buildings in a Single Bound Other than Superman

---

Super Hero

---

Under Dog



---

Example: Text Parameter Specified



# The End.



## More on Delimiters Used by SAS® Scan Function

-This macro uses the SAS® scan function therefore the delimiter must be at the start of the item e.g. `strg=red ^blue ^green`

-Leading delimiters before the first word in the character string do not affect SCAN e.g. `string=^red^blue;`

-If there are two or more contiguous delimiters, SCAN treats them as one. e.g. `string=red^^blue;`

-If you omit *delimiters* in an ASCII environment, SAS uses the following characters: blank . < ( + & ! \$ \* ) ; ^ - / , % |

-In ASCII environments without the ^ character, SCAN uses the ~ character instead.

-If you omit *delimiters* on an EBCDIC environment, SAS uses the following characters: blank . < ( + | & ! \$ \* ) ; ¬ - / , % | ¢