```
%************************************************************************
*             CLIENT:
*           PROTOCOL:
*       PROGRAM NAME:   MR_PAGEBREAK_TABLE.SAS
*       SAS VERSION:    9.1.3, 9.2
*            PURPOSE:   Macro to assist in breaking RTF pages for non-nested tables
*
*        USAGE NOTES:
*
*
*        INPUT FILES:
*       OUTPUT FILES:
*
*             AUTHOR:    (b) (6)
*       DATE CREATED:   13JUN2011
*
*   MODIFICATION LOG:
*            VERSION:   V1.0
************************************************************************
* DATE      INITIALS    MODIFICATION DESCRIPTION
************************************************************************
* 11NOV2013 (b) (6)     [001] Update to fix section protection for tables to work
with artz macros
* 13MAR2014 (b) (6)     [001] Update was not handling casing of section and orderby
variables when determining
*                                                       section order variables.
************************************************************************
*  © 2013 Pharmaceutical Product Development, Inc.
*  All Rights Reserved.
************************************************************************;
%macro MR_PAGEBREAK_TABLE (
                vartx=,
                lines_left_on_page=&G_LINESLEFT,
                in_data=&syslast,
                out_data=,
                escapechar=^,
                orderby=,
                subgrp=%str(),
                section=%str(),
                lineskip=1,
                debug=0,
                section_protection=Y) / store  source des='V1.0.0.13' ;

%PUT ---------------------------------------------- ;
%PUT INFO: (MR_PAGEBREAK_TABLE);
%PUT INFO:  Version 1.0;
%PUT -START---------------------------------------- ;

%**** START PARAMETER CHECKS ******;
```

```
%global MR_PAGEBREAK_TABLE_RC ;



%mu_check_req_parameters
 (  parameters_to_check = section lines_left_on_page in_data );

%if &out_data eq %str() %then %let out_data=&in_data._pb;

%if &orderby eq %str() %then %let orderby=&subgrp &section;
%if &in_data ne %str() %then %let datacheck1 = %substr(&in_data,1);
%else %let datacheck1=0;
%if not %sysfunc(exist(&in_data)) %then %let datacheck1=0;
%let datacheck2 = %substr(&out_data,1);

%let section_protection=%upcase(&section_protection);
%if &section_protection eq YES %then %let section_protection=Y;
%if &section_protection eq NO  %then %let section_protection=N;

%*** this gets rid of the descending and ascending keywords from orderby when just
for checking when variables exist;
%let order_check=%sysfunc(tranwrd(%sysfunc(lowcase(&orderby)),descending,%str()) );
%let order_check=%sysfunc(tranwrd(%sysfunc(lowcase(&order_check)),ascending,%str())
);
/*%put ORDER_CHECK = &order_check;*/

%mu_check_data_and_var_exist(
 data_to_check = &in_data
,vars_to_check_in_all_data = &order_check &vartx &section &subgrp
);



%let MR_PAGEBREAK_TABLE_RC=0;


 %if %sysfunc(verify(&lines_left_on_page,'0123456789'))ne 0 %then %do;
      %let MR_PAGEBREAK_TABLE_RC=1;
 %end;

 %if %sysfunc(verify(&lines_left_on_page,'0123456789')) eq 0 %then %do;
         %if &lines_left_on_page lt 1 %then %let MR_PAGEBREAK_TABLE_RC=1;
 %end;

 %if %sysfunc(verify(&datacheck1,'0123456789'))eq 0  %then %do;
                %let MR_PAGEBREAK_TABLE_RC=2;
 %end;
```

```
 %if %sysfunc(verify(&escapechar,'0123456789'))eq 0  %then %do;
                %let MR_PAGEBREAK_TABLE_RC=3;
 %end;

 %if %length(&escapechar) ne 1  %then %do;
                %let MR_PAGEBREAK_TABLE_RC=4;
 %end;

 %if %sysfunc(verify(&datacheck2,'0123456789'))eq 0  %then %do;
                %let MR_PAGEBREAK_TABLE_RC=5;
 %end;

 %if %sysfunc(verify(&lineskip,'.015')) ne 0 %then %do;
          %let MR_PAGEBREAK_TABLE_RC=6;
 %end;

 %if &MR_PAGEBREAK_TABLE_RC lt 1 %then %do;
                proc contents data=&in_data out=___out noprint;
                run;

                data _null_;
                        length invalid $200;
                        set ___out;
                        retain invalid '';
                        %let i=1;
                        %do %while(%scan(&vartx,&i) ne %str() ) ;
                                if compress(upcase("%scan(&vartx,&i)"))  =
compress(upcase (NAME))  and type ne 2 then do;
                                        call
symput('MR_PAGEBREAK_TABLE_RC',put(7,1.));
                                        if invalid eq '' then invalid =
compress(upcase("%scan(&vartx,&i)"));
                                        else invalid = trim(left(invalid))||",
"||compress(upcase("%scan(&vartx,&i)"));
                                        flag+1;
                                end;
                                %let i=%eval(&i+1);
                        %end;
                        call symput ('MR_INVALID',trim(invalid) );
                run;
 %end;


%**** END PARAMETER CHECKS ******;

%***** START PAGEBREAK CALCULATION ****;
%if &MR_PAGEBREAK_TABLE_RC lt 1 and &MU_CHECK_DATA_AND_VAR_EXIST_RC lt 1 and
&MU_CHECK_REQ_PARAMETERS_RC lt 1 %then %do;
```

```
                *** Code to handle missing flow character variables ****;
                **  takes all character variables and tries to find flow
characters;
                %if &vartx. eq %then %do ;
                        proc contents data = &in_data. out = ___out noprint ;
                        run ;

                        proc sort data=___out tagsort;
                                by varnum;
                        run;

                        proc sql noprint ;
                                select name into :vartx separated by ' '
                                        from ___out
                                        where type eq 2
                                ;
                        quit ;
                        %put NOTE: No variable list supplied for VARTX. Using
VARTX=&vartx.;

                        proc datasets lib=work nolist ;
                                delete ___out ;
                        run ;
                %end ;


                %macro carriage_returns (keepfirst=Y) / des='Defined in
MR_PAGEBREAK_TABLE' ;
                        %*** calculate how many ^n values were inserted to help
determine total number of lines *** ;
                         count_carriage_returns = 0 ;
                         %*** check ordervar variables as to if they are being
displayed which is when first.&__o__. occurs ;
                         %let __count__ = 1 ;
                         %let __o__ = %scan(&ordervar., &__count__., %str( ) );
                          %let iter=0;
                          %do %while ("&__o__." ne "");
                                 %if &iter. eq 1 %then %do ;
                                  %let iter=0;
                                 %end ;

                                 %if &keepfirst=Y %then %do;
                                 if first.&__o__. then do ;
                                 %end;
                                  %*** replace ^n with 'bb'x which is one character
not likely to appear in the data *** ;
                                  trans1 = tranwrd( &__o__., "&escapechar.n", 'bb'x
) ;
                                  %*** remove 'bb'x character from string *** ;
```

```
                                %*** to see how many ^n characters are in &v
check for the difference in length between trans1 and removed 'bb'x character from
string *** ;
                                count_carriage_returns = max( length( strip (
trans1 ) ) - length( compress( strip ( trans1 ), 'bb'x ) ),
                                          count_carriage_returns );
                               %if &keepfirst=Y %then %do;
                           end ;
                               %end;

                               %let iter=1;
                               %let __count__= %eval( &__count__.+1 );
                               %let __o__ = %scan(&ordervar., &__count__., %str(
) );
                            %end ;

                           %let _hhh=1;
                           %*** begin loop to examine each variable supplied in list
&notordervar. *** ;
                           %let v = %scan(&notordervar., &_hhh., %str( ) );
                           %do %while ("&v." ne "");
                                %*** replace ^n with 'bb'x which is one character
not likely to appear in the data *** ;
                                trans1 = tranwrd( &v., "&escapechar.n", 'bb'x ) ;
                                %*** remove 'bb'x character from string *** ;

                                %*** to see how many ^n characters are in &v
check for the difference in length between trans1 and removed 'bb'x character from
string *** ;
                                count_carriage_returns = max( length( strip(
trans1 ) ) - length( compress( strip( trans1 ), 'bb'x ) ),
                                          count_carriage_returns );
                            %let _hhh= %eval( &_hhh.+1 );
                            %let v=%scan(&notordervar., &_hhh., %str( ) );
                           %end ;
                           %*** end loop *** ;
                 %mend;


                %*** Sort Dataset ***;

                proc sort data= &in_data out=&out_data. tagsort ;
                        by &orderby. ;
                run;


                %*** Creates Pagebreak Variable ***;
                data  &out_data.
                          %if &debug = 1 %then %do;
                                       __debug
```

```
                    %end;
                ;

                 length trans1 $ 1000 ;
                 drop trans1 ;
                 retain  __linesleft &lines_left_on_page. pagebrk __line ;


                 set &out_data. ;
                   by &orderby. ;

                 %*build ordervar which is variables report is to be sorted by
except for treatment variable ;
                 %let ordervar=;
                 %let notordervar=;
                 %let _hhh=1;
                 %*** begin loop to examine each variable supplied in list &vartx.
*** ;
                 %let v = %scan(&vartx., 1, %str( ) );
                 %do %while ("&v." ne "");
                    %let _iii=1;
                    %let __o__ = %scan(&orderby., &_iii., %str( ) );
                    %let found=0;
                    %do %while("&__o__" ne "") ;

                         %if "&v." eq "&__o__"  %then %let found=1;

                         %let _iii= %eval( &_iii.+1 );
                         %let __o__=%scan(&orderby., &_iii., %str( ) );
                    %end ;

                    %* quick fix for subgrp test;
                    %if &subgrp.=%str() %then %let subgrp2=#;
                    %else                     %let subgrp2=&subgrp.;

                    %if "&v." ne "&subgrp2." %then %do;
                        %if &found. eq 1  %then %let
ordervar=%trim(%left(&ordervar. &v.));
                        %else                   %let
notordervar=%trim(%left(&notordervar. &v.));
                    %end;

                    %let _hhh= %eval( &_hhh.+1 );
                    %let v=%scan(&vartx., &_hhh., %str( ) );
                 %end ;


                 %carriage_returns (keepfirst=Y);

                 xnlinesx = count_carriage_returns + 1;  %* number of lines used
```

```
for this record -;

                %*** if first treatment reset __line to carriage return count plus
one and increment pagebrk by one *** ;
                %if &subgrp. ne %str() %then %do;
                    if first.&subgrp. then do;
                       __line=( count_carriage_returns + 1 ) ;
                       pagebrk+1;
                    end;
                %end;
                %else %do;
                    if _n_=1 then do;
                       _line=( count_carriage_returns + 1 ) ;
                       pagebrk+1;
                    end;
                %end;
                else do;
                   __line+( count_carriage_returns + 1 ) ; *** else add carriage
returns plus one onto line *** ;
                end;

                %*** take into account a blank line above the breaking variable
(usually patient number for listings) *** ;
                if first.&section. then do;
                    __line+&lineskip.;
                    xnlinesx = xnlinesx + 1;

                    %if &section_protection ne Y %then %do;
                        if __line gt &lines_left_on_page. or ceil(__line) gt
&lines_left_on_page. then do;
                           pagebrk+1;

                           %carriage_returns(keepfirst=N);
                           __line=( count_carriage_returns + 1) +&lineskip.;
                        end;
                    %end;
                 end;


               else do;
                  %if &section_protection ne Y %then %do;

                    %*** if the total number of lines equals or exceeds the
requested number of &lines_left_on_page then increment pagebrk *** ;
                    if __line gt &lines_left_on_page. or ceil(__line) gt
&lines_left_on_page. then do;
                        pagebrk+1;
                        %carriage_returns(keepfirst=N);
                        __line=( count_carriage_returns + 2 );
                   end;
```

```
              %end;
            end;
            %if &debug eq 0 and &section_protection ne Y %then %do;
                 drop  count_carriage_returns;
            %end;
          run;

    %*** SECTION PROTECTION: Creates Counts for how many lines that are in a
section ***;

    %if &section_protection=Y %then %do;

        %let orderby = %upcase(%cmpres(&orderby));
        %let section = %upcase(&section);

        %* -- remove all variables from sort occuring AFTER section variable ----;
        %let sectx   = %sysfunc(indexw(%bquote(&orderby), %str(&section) ));

        %mu_checkrc(condition = &sectx < 1
                   ,msg1      = %str(MR_PAGEBREAK_TABLE: Program could not find
SECTION variable [&section] in sort order ORDERBY)
                   ,msg2      = %str(MR_PAGEBREAK_TABLE: ORDERBY [&orderby])
                   ,rcprefix  = MR_PAGEBREAK_TABLE
                   ,setabort  = Y
                    )

        %if &orderby=&section %then %let sectord= %substr(%bquote(&orderby), 1 ,
%length(&section) );
        %else %let sectord = %substr(%bquote(&orderby), 1 , %eval(&sectx +
%length(&section)) );
        %let sectord = %trim(&sectord);

        %* -- replace spaces with commas for use in proc sql --------------------;
        %let sqlord  = %sysfunc(translate(%bquote(&sectord), %str(,), %str( ) ));

        %* -- The above will also delimit keywords of descending and ascending.  So
these next statements fixes where present *;
        %let sqlord  = %qcmpres(%bquote(&sqlord));
        %let sqlord  = %sysfunc(tranwrd(%bquote(&sqlord),ASCENDING%str(,), %str(
)));  %* -- get rid of ascending --;
        %let sqlord  = %sysfunc(tranwrd(%bquote(&sqlord),DESCENDING%str(,), %str(
)));  %* -- remove descending from var list --;

        %* -- count number of lines used for each section ---;
        proc sql;
          create table sectionlines as
          select pagebrk, &sqlord, sum(xnlinesx) as sectlines
            from &out_data
           group by pagebrk, &sqlord
            ;
```

```
        quit;
        proc sort data=sectionlines;
          by pagebrk &sectord;
        run;


        %* -- Merge counts for each section onto data -------;
        data &out_data(drop=pagebrk cnt  __linesleft __line xnlinesx sectlines);
          merge &out_data sectionlines;
            by pagebrk &sectord;

               if first.pagebrk then do; cnt =0; pageno +1; end;
               if first.&section then do;
                  cnt + sectlines;
                  if ceil(cnt) > &lines_left_on_page. then do;  cnt=sectlines + 1;
pageno+1; end;
               end;
         run;

         proc sort data=&out_data(rename=(pageno=pagebrk));
           by pagebrk &orderby;
         run;

    %end;

        %global maxpage ;
        data &out_data( sortedby=pagenum &orderby. ) ;
         set &out_data. end =last;
             rename pagebrk=pagenum;
             if last then call symputx('maxpage',pagebrk );
        run;


        %PUT ------------------------------------------------- ;
        %PUT INFO: MR_PAGEBREAK_TABLE_RC = &MR_PAGEBREAK_TABLE_RC;
        %PUT ------------------------------------------------- ;

%end;

%else %do;

        %PUT ------------------------------------------------- ;
        %PUT INFO: MR_PAGEBREAK_TABLE_RC = &MR_PAGEBREAK_TABLE_RC;
        %PUT ------------------------------------------------- ;


        %if &MR_PAGEBREAK_TABLE_RC=1 %then %do;
                  %put %str(ALERT_)P: &SYSMACRONAME - Valid values for
LINES_LEFT_ON_PAGE are numeric and greater than 0. Please use...;
                  %put %str(ALERT_)P: &SYSMACRONAME - ...a valid value for
```

```
LINES_LEFT_ON_PAGE and try again. Processing will be stopped!;
         %end;

        %if &MR_PAGEBREAK_TABLE_RC=2 %then %do;
                        %put %str(ALERT_)P: &SYSMACRONAME - IN_DATA must exist, not
be equal to missing, or start with a numeric value.;
                        %put %str(ALERT_)P: &SYSMACRONAME - Please use a valid
value for IN_DATA and try again. Processing will be stopped!;
         %end;

        %if &MR_PAGEBREAK_TABLE_RC=3 %then %do;
                        %put %str(ALERT_)P: &SYSMACRONAME - ESCAPECHAR must not  be
missing or a numeric  value.;
                        %put %str(ALERT_)P: &SYSMACRONAME - Please use a valid
value for ESCAPECHAR and try again. Processing will be stopped!;
         %end;

        %if &MR_PAGEBREAK_TABLE_RC=4 %then %do;
                        %put %str(ALERT_)P: &SYSMACRONAME - Length of ESCAPECHAR
must  be equal to 1;
                        %put %str(ALERT_)P: &SYSMACRONAME - Please use a valid
value for ESCAPECHAR and try again. Processing will be stopped!;
        %end;

        %if &MR_PAGEBREAK_TABLE_RC=5 %then %do;
                        %put %str(ALERT_)P: &SYSMACRONAME - OUT_DATA must not start
with a numeric  value. Please use...;
                        %put %str(ALERT_)P: &SYSMACRONAME - ...a valid value for
OUT_DATA and try again. Processing will be stopped!;
        %end;


        %if &MR_PAGEBREAK_TABLE_RC=6 %then %do;
                  %put %str(ALERT_)P: &SYSMACRONAME - Valid values for the LINESKIP
parameter are 0=(No) 1=(Yes) .5=(Half a line) . Please use...;
                  %put %str(ALERT_)P: &SYSMACRONAME - ...a valid value for LINESKIP
and try again. Processing will be stopped!;
        %end;

        %if &MR_PAGEBREAK_TABLE_RC=7 %then %do;
                  %put %str(ALERT_)P: &SYSMACRONAME - Valid variables for the VARTX
parameter are character;
                  %put %str(ALERT_)P: &SYSMACRONAME - Please remove numeric
variable(s):  &MR_INVALID;
                  %put %str(ALERT_)P: &SYSMACRONAME - Processing will be stopped!;
        %end;



%end;
```

```
%PUT ---------------------------------------------- ;
%PUT INFO: (MR_PAGEBREAK_TABLE);
%PUT -END------------------------------------------ ;

%mend MR_PAGEBREAK_TABLE ;
```