PharmaSUG 2026 - Paper PK-234

Visualizing PK and ADA Data at Scale: A Parameter-Driven SAS Macro for Box Plot Generation

Prasannanjaneyulu Narisetty, Anilkumar Anksapur, Merck & Co., Inc., Rahway, NJ, USA.

ABSTRACT

Effective visualization of Pharmacokinetic (PK) and Anti-drug Antibody (ADA) data is vital for interpreting complex clinical trial data. This presentation introduces a dynamic, user-friendly SAS macro designed to generate high-quality box plots for Immunogenicity analyses, with a particular focus on ADA evaluations.

The macro creates two different visualizations:

1. Antibody Titer by Study Visit
2. ADA Sample Result vs. Predose Serum Concentration by Study Visit and ADA Status

This paper demonstrates how the macro generates two plots through extensive, parameter-driven customization. Users can define the analysis population and input datasets, apply filters, specify X and Y axis variables, incorporate ADA status, and add statistical summaries. The macro verifies that the specified X and Y variables exist in the input dataset and ensures that color specifications remain consistent with data groupings. It also identifies missing cycle values and duplicate X-axis tick labels, and it provides clear, actionable log messages when required inputs are missing or incorrectly defined.

INTRODUCTION

Analysts often need to display distributions of TITER values across visits and compare predose concentrations by ADA sample status (NEGATIVE vs POSITIVE).

These displays are often generated using SAS programs tailored to individual studies, which can result in inconsistent figures, duplicated programming effort, and challenges when reviewing results across multiple studies.

To address these issues, we developed %box0plot0ada, a parameter-driven SAS macro that standardizes box plot creation for ADA-related analysis using Analysis Data Model (ADaM) datasets. The macro promotes consistency across studies and offers flexible parameterization for common analysis scenarios and includes robust inputs validation to help prevent common programming errors.

This paper describes the macro's core parameters, validation features, key code concepts, and illustrates its application using two common PK/ADA visualization examples.

MACRO OVERVIEW

The %box0plot0ada macro is designed to help programming teams efficiently generate consistent, standardized ADA-related box plots with minimal custom coding. Specifically, it:

- Produces standardized box plots for key ADA and PK analyses.
- Uses ADaM inputs (ADSL or ADBASE and ADADA) to define analysis populations and observations.
- Supports parameter-driven control over analysis population and observation filters, X-axis definitions (CYCLE or VISIT), optional ADA status grouping, Y-axis variables (including LOG transformations), layout options, and display of statistics and percentages using XAXISTABLE.

- Performs defensive input validation with clear, actionable SAS log messages.
- Delivers Rich Text Format (RTF) outputs and optional datasets containing computed statistics.

At a high level, %box0plot0ada takes subject-level population data (e.g., ADSL or ADBASE) and observation data (e.g., ADADA), applies user-specified filters, derives plotting variables, and then generates box plots using PROC SGPLOT.

KEY PARAMETERS

The macro uses a set of parameters to control population selection, observation filtering, axis construction, and display options. Key parameters include:

- ***population_from***: ADaM dataset to identify the analysis population (e.g., ADSL).
- ***population_where***: Subset criteria for the analysis population (e.g., ADAEVFL = "Y").
- ***observation_from***: ADaM dataset containing ADA or PK observations (typically ADADA).
- ***observation_where***: Subset criteria for observations (e.g., ADAEVFL = "Y" and ADA status filters).
- ***xvar***: X-axis definition, such as CYCLE, VISIT, CYCLE|ALL, or VISIT|ALL.
- ***yvar***: Y-axis variable. (e.g., log TITERN or CONC).
- ***adastatus***: Indicator (Y or N) to include ADA status on the X-axis.
- ***adastatsvar***: ADA status variable (e.g., CNRRSLT).
- ***display_pct***: Indicator (Y or N) to display n/N and percentages using XAXISTABLE.
- ***stats***: Summary statistics to display (e.g., n nblq mean median sd).
- ***blqvar***: Below the Limit of Quantification (BLOQ) flag variable if pre-derived.
- ***bloq***: Numeric threshold used to derive BLOQ when BLOQ is not pre-defined.
- ***scatter***: Indicator (Y or N) to overlay subject-level points.
- ***boxdata***: ADA status values used to define groups (e.g., NEGATIVE|POSITIVE).
- ***boxlinecolor***: Box line colors for ADA groups (e.g., red|green).

These parameters can be combined to cover common PK/ADA plotting scenarios without modifying the macro code.

DATA REQUIREMENTS

To use %box0plot0ada effectively, several data requirements must be met.

First, the observation-level dataset (e.g., ADADA) should contain one record per subject and visit (or per subject, visit, and ADA status) after all filters are applied. When duplicates occur, the macro automatically selects the most recent record for plotting.

Second, cycle / visit information must be managed consistently. When non-cycle visits are present or when visit names share similar prefixes, it is recommended to define both a numeric VISIT informat and a character VISIT format. This ensures proper ordering and produces clear, uniquely identifiable labels. Using uppercase visit values in format definitions helps prevent inconsistencies.

Third, BLOQ handling should be explicitly defined. The macro can reference a pre-derived BLOQ flag (via *blqvar*) or derive BLOQ status internally based on PCSTRESC, DTYPE, and an optional numeric threshold (*bloq*). Values such as “BLQ,” “BLOQ,” or those beginning with “<” are typically treated as BLOQ.

Finally, population-level and observation-level filters must be clearly specified and parameter-driven. Filters applied to ADSL or ADBASE determine the analysis population, while filters applied to ADADA define which records are included in the final displays.

Graph Ready Dataset structure for Example 1 plot:

USUBJID	VISIT	XVAR	XVARC	YVAR	PCT	PCTVAL	STAT_LBL	STATS
001	CYCLE 1 DAY	01	C1	XX				
002	CYCLE 1 DAY	01	C1	XX				
003	CYCLE 1 DAY	01	C1	XX				
004	CYCLE 1 DAY	01	C1	XX				
005	CYCLE 1 DAY	01	C1	XX				
	CYCLE 1 DAY	01	C1		PCT	(XX.X%)		
	CYCLE 1 DAY	01	C1		PCTVAL	n/N		
	CYCLE 1 DAY	01	C1				AM	XX.X
	CYCLE 1 DAY	01	C1				Median	XX.X
	CYCLE 1 DAY	01	C1				n	XX
001	CYCLE 2 DAY	02	C2	XX				
002	CYCLE 2 DAY	02	C2	XX				
003	CYCLE 2 DAY	02	C2	XX				
004	CYCLE 2 DAY	02	C2	XX				
005	CYCLE 2 DAY	02	C2	XX				
	CYCLE 2 DAY	02	C2		PCT	(XX.X%)		
	CYCLE 2 DAY	02	C2		PCTVAL	n/N		
	CYCLE 2 DAY	02	C2				AM	XX.X
	CYCLE 2 DAY	02	C2				Median	XX.X
	CYCLE 2 DAY	02	C2				n	XX

Graph Ready Dataset structure for Example 2 plot:

USUBJID	VISIT	CNRRSLT	XVAR	XVARC	YVAR	STAT_LBL	STATS
001	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg	XX		
002	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg	XX		
003	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg	XX		
004	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg	XX		
005	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg	XX		
	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg		n	XX
	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg		n blq	XX
	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg		AM	XX.X
	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg		Median	XX.X
	CYCLE 1 DAY 1	NEGATIVE	0101	C1, Neg		SD	XX.X
006	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos	XX		
007	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos	XX		
008	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos	XX		
009	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos	XX		
010	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos	XX		
	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos		n	XX
	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos		n blq	XX
	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos		AM	XX.X
	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos		Median	XX.X
	CYCLE 1 DAY 1	POSITIVE	0102	C1, Pos		SD	XX.X

ERROR CHECKS AND VALIDATIONS

The macro includes built-in validation to catch common issues early in the programming process:

- **Missing variables**: The macro validates that the variables specified in *xvar*, *yvar*, and *adastatsvar* are present in the input datasets. If any required variable is absent, the macro terminates and issues a clear error message in the SAS log.
- **Color definition mismatch**: The macro confirms that the number of colors provided in *boxlinecolor* matches the number of values defined in *boxdata*. If these counts are inconsistent, execution stops and an explanatory message is printed.
- **Non-cycle visits**: When non-cycle visits are detected with missing cycle numbers, the macro requires user-defined VISIT informats and VISIT formats. If these are not supplied, the macro stops and provides instructions, including example INVALUE and VALUE statements to assist in defining the necessary formats.

These checks are intended to prevent the generation of partially drawn plots and avoid misleading, and to guide the user to correct the data or parameter issues.

KEY CODE EXCERPTS — HIGHLIGHTING MACRO FEATURES

1) PARAMETER DEFAULTS AND DEFENSIVE CHECKS

Validate inputs, and halt gracefully with informative messages:

```
%let _xvar1 = %dsVar(ds=&observation_from, var=%scan(&xvar, 1, " "), operator=IN);

%if %quote(&_xvar1)= %then
    %do;
        %put ERROR: (&sysmacroname) X Axis variable &xvar not found in &observation_from.;
        %goto exit_macro;
    %end;
```

2) AXIS CONSTRUCTION FOR CYCLE/VISIT AND ADA STATUS

Composite X keys and labels (e.g., C01, C01,Neg / C01,Pos) and $CYFMT to ensure clean ticks:

```
%let _xvar = cy;
length cy $12;
if index(upcase(visit), "CYCLE") > 0 then
    _cycle = input(scan(substr(visit, index(upcase(visit), "CYCLE")), 2, " "), best.);
else %if "%sysfunc(compress(&visitfmt))" ne "0" %then
    _cycle = input(upcase(visit), visit.);
else
    _cycle = .;
if _cycle ne . then do;
    cy = put(_cycle, z2.);
    cyc = "C" || put(_cycle, z2.);
end;
else do;
    %if "%sysfunc(compress(&visitfmt))" ne "0" %then
        cyc = strip(put(upcase(visit), $visit.));
    %else
        cyc = substr(strip(upcase(visit)), 1, 4);
end;

proc sort data=data_graph out=uni_cy(keep=&_xvar &_xvar.c) nodupkey;
    by &_xvar &_xvar.c;
run;

/* Check duplicate cycle tick labels when the first 4 letters of &_xvar.c values are same
   and missing cycle numbers in X-axis */
%if &dup_lbl > 0 %then %do;
    %if &visitnfmt = 0 %then %do;
        %put ERROR: Define VISIT format for non cycle visits when the first 4 letters of visit value is same;
        %put INFO: Assign the format values as needed;
        %put e.g. %sysfunc(fmtvisit);
        %put "CROSSOVER FOLLOW-UP DAY 30" = "CCFU";
        %put "CROSSOVER END OF TREATMENT" = "CCOT";
        %goto exit_macro;
    %end;
%end;
```

3) BLOQ DERIVATION AND LOG-Y SUPPORT

Derive BLOQ when missing and support LOG <var> to auto-log transform Y:

```
%let _blqvar = %dsVar(ds=&observation_from, var=&blqvar, operator=IN);
%let _pcstresc = %dsVar(ds=&observation_from, var=pcstresc, operator=IN);
%let _dtype    = %dsVar(ds=&observation_from, var=dtype, operator=IN);

if "&_blqvar" = "" then
    do;
        &blqvar = 0;

        if %upcase("&_pcstresc") = PCSTRESC then if upcase(pcstresc) in ('BLQ','BLOQ') or find(pcstresc,'<') then
            &blqvar=1;

        if %upcase("&_dtype") = DTYPE then if &blqvar = 0 and upcase(dtype)='LLOQ' then
            &blqvar=1;

        %if &bloq ne %str() %then
            %do;
                if &blqvar=0 and conc ne . and conc <= &bloq then
                    &blqvar=1;
            %end;
    end;

%if %scan(%upcase(&yvar),1) = LOG %then
    %do;
        %let _tmpy = %scan(%upcase(&yvar), -1);

        if &_tmpy > 0 then
            log&_tmpy = log(&_tmpy);
        %let _yvar_ = log&_tmpy;
    %end;
```

4) DEFINE ATTRMAP DATASET, PROC SGPLOT AND XAXISTABLE RENDERING

Define ATTRMAP dataset:

```
%if %upcase(&adastatus)=%str(Y) %then
    %do;

        data attrmap;
            length value fillcolor linecolor markercolor markersymbol $200.;
            retain id "ADA";

            do _i=1 to &count_c.;
                value=upcase(scan("&boxdata.", _i, "|"));
                fillcolor="white";
                linecolor=scan("&boxlinecolor.", _i, "|");
                markercolor=scan("&markercolor.", _i, "|");
                markersymbol=scan("&markersymbol.", _i, "|");
                markerSize=input(scan("&markersize.", _i, "|"), best.);
                output;
            end;
            drop _i;
        run;

    %end;
```

SGPLOT Procedure:

```
proc sgplot data=_final_gdata
    %if %upcase(&adastatus)=%str(Y) %then
        dattrmap=attrmap;
    noautolegend;
    format &_xvar $cyfmt. ;
    vbox &_yvar_ / category=&_xvar
        %if %upcase(&adastatus)=%str(Y) %then
            group=adastat attrid=ADA;
        %else whiskerpct=<x> meanattrs=(symbol=diamond color=black size=<xx>) boxwidth=<x.x>;;
    %if %upcase(&scatter)=%str(Y) %then
        %do;
            scatter x=&_xvar y=&_yvar_ / jitter jitterwidth=<x.x>
            %if &adastatus=%str(Y) %then
                group=adastat attrid=ADA;
            markerattrs=(size=<xxcm>);
        %end;

    %if %upcase(&display_pct)=%str(Y) %then
        %do;
            xaxistable pctval / x=&_xvar class=pct location=inside nomissingclass
                nolabel valueattrs=(Color=black Family=Arial Size=&num_size);
        %end;

    %if &stats ne %str() %then
        %do;
            xaxistable stats / x=&_xvar class=stat_lbl location=outside nomissingclass
                valueattrs=(Color=black Family=Arial Size=&num_size);
        %end;

    xaxis label="&xaxislbl." discreteorder=data labelattrs=(Color=black
        Family=Arial Size=<xx> Weight=Bold) valueattrs=(Color=black Family=Arial
        Size=<xx> Weight=Bold) fitpolicy=rotate valuesrotate=vertical;
    yaxis label="&yaxislbl." labelattrs=(Color=black Family=Arial Size=<xx>
        Weight=Bold) valueattrs=(Color=black Family=Arial Size=<xx> Weight=Bold);
run;
```

USAGE EXAMPLES

Example 1. Antibody Titer by Study Visit (with percent display):

This example demonstrates how to display log-transformed antibody titers by visit, with percentages displayed below each box.

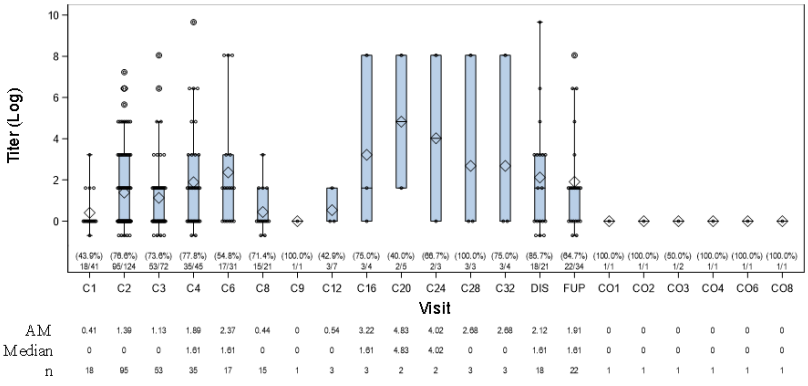
```
proc format;
    invalue visitn
        "FOLLOW-UP" = 35
        "DISCONTINUATION"=36 ;
    value $visit
        "FOLLOW-UP"="FUP"
        "DISCONTINUATION"="DIS" ;
run;

%box0plot0ada(
    population_from   = %str(adam.adbase, adam.adsl),
    population_where  = %str(adbase.usubjid=a.usubjid and adbase.adaevlfl="Y"),
    observation_from  = %str(adam.adada),
    observation_where = %str(adaevlfl = "Y"),
    xvar              = %str(visit),
    xaxislbl          = %str(Visit),
    yaxislbl          = %str(Titer (Log)),
    yvar              = log titern,
    bloq              = 0,
    display_pct       = Y,
    stats             = mean median n
    );
```

This call produces a standardized box plot of log-transformed antibody titers by visit for ADA evaluable participants, with n and percentages displayed below each visit.

Example 1 Output.

Antibody Titer by Study Visit (ADA Evaluable Participants)



ADA=antidrug antibody; AM = Mean; n=number of evaluable participants.
The dots are the data points, boxes represent 25th to 75th percentiles, the line in the box is the median, the diamond in the box is the mean, and whiskers are 5th and 95th percentiles.
*Numerator#: Number of participants with titer values at sampling timepoint; Denominator#: Total evaluable participants with available ADA samples; Percentage#: 100*Numerator/Denominator.*
Source: [adam-adsl; adbase; adada]

Example 2. ADA Sample Result vs Predose Serum Concentration by Visit and ADA Status:

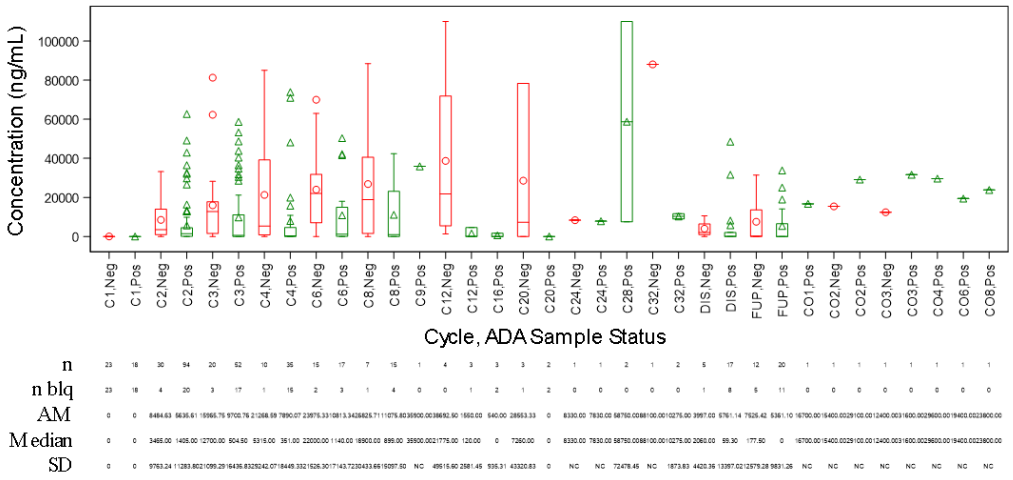
This example produces a plot of predose concentration by visit and ADA status, grouped on the X-axis.

```
%box0plot0ada(
  population_from   = %str(adam.adbase, adam.adsl),
  population_where  = %str(adbase.usubjid=a.usubjid and adbase.adaevlfl="Y"),
  observation_from  = %str(adam.adada),
  observation_where = %str(adaevlfl = "Y" and cnrrslt in ("NEGATIVE" "POSITIVE")),
  xvar              = %str(visit),
  adastatus         = Y,
  adastatsvar       = cnrrslt,
  xaxislbl          = %str(Cycle, ADA Sample Status),
  yaxislbl          = %str(Concentration (ng/mL)),
  yvar              = conc,
  scatter           = N,
  bloq              = 0,
  stats             = n nblq mean median sd
);
```

This call creates a box plot of predose concentrations by visit and ADA status (NEGATIVE versus POSITIVE), summarizing counts, BLOQ counts, and key statistics.

Example 2 Output.

ADA Sample Result vs Predose Serum Concentration
(ADA Evaluable Participants)



ADA = Anti-drug antibody; Neg = Negative; Pos = Positive.
n = number of samples; n blq = number of samples with result < BLQ; AM = Arithmetic mean; SD = Standard deviation; NC = Not Calculated.
Source: [adam-adsl; adbase; adada]

DISCUSSION

The %box0plot0ada macro offers several strengths for PK and ADA visualization:

- **Standardization**: The macro produces consistent, recognizable outputs across protocols and programs when using ADaM inputs, supporting internal standards and external reporting.
- **Flexibility**: It supports multiple X-axis definitions, optional ADA status grouping, log transformations, and configurable layout and style options, allowing reuse across different study designs.

- **Robustness**: Built-in input validation handles missing variables, mismatched color lists, missing formats for non-cycle visits, and invalid page margin values, reducing the risk of generating incorrect or incomplete plots.

At the same time, there are practical considerations and limitations. Data must be sufficiently prepared, including consistent ADA status values and clearly defined BLOQ handling. Non-cycle visits require user-defined formats and informats, which introduces additional setup but yields better control over axis labels and ordering. Use of the "|all" syntax with sparse data can result in SAS warnings and missing boxes, and users should confirm that this behavior is appropriate for the study.

Future enhancements could include additional grouping variables (e.g., treatment arms), automated small-multiples or panel plots by treatment or region.

CONCLUSION

The **%box0plot0ada** macro integrates established best practices for ADA and PK data visualization into a standardized and reusable solution. Through rigorous input validation, automated and consistent axis construction, and flexible layout and styling controls, it supports the rapid generation of publication-quality immunogenicity graphics while strengthening reproducibility across studies.

This standardized framework reduces reliance on study-specific code, streamlines programming workflows, and enables more consistent, transparent, and reliable data review throughout clinical development.

REFERENCES

- SAS ODS Graphics: Procedures Guide. Cary, NC: SAS Institute Inc.
- Base SAS Procedures Guide. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank Merck & Co., Inc., Rahway, NJ, USA. for this opportunity to present at the PharmaSUG conference. Sincere thanks to Amy Gillespie, Jing Su, and Sandeep Meesala for reviewing the paper and providing valuable feedback.

RECOMMENDED READING

Base SAS® Procedures Guide

CONTACT INFORMATION

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

Author Name: Prasannanjaneyulu Narisetty
Company: Merck & Co., Inc., Rahway, NJ, USA.
Email: prasannanjaneyulu.narisetty@merck.com

Author Name: Anilkumar Anksapur
Company: Merck & Co., Inc., Rahway, NJ, USA.
Email: anilkumar.anksapur@merck.com