

Object-Oriented Reporting Tools

Jennifer Savage, STATPROBE, Inc., Ann Arbor, MI

Abstract

This paper describes the reasons for and the development of object-oriented report tools for clinical data. The primary motivations to develop the **listtool** and **sumtool** macros are to generate reusable code and, therefore, increase programmer and code efficiency. These macros reduce the amount of coding and debugging time required for each individual listing or summary program and promote standard programming techniques. An additional motivation is the desire to have user friendly standard reporting tools without the limitations of the SAS[®] procedures, such as PROC PRINT and PROC REPORT. STATPROBE has developed these macros on a project by project basis to avoid the unwanted inflexibility and restrictions that often accompany over-standardization.

To develop these macros, the framework of the DATA _NULL_ as a reporting tool is abstracted from the individual report programs, standardized, and stored in the **listtool** and **sumtool** macros. Table-specific conditions, titles, and code are provided as parameters in the macro invocation. The individual report programs are set up in a modular fashion. The modules or "blocks" of code include any necessary data manipulation and table-specific code for header, body, and footer of the table. The modular design makes the programs easy to read, understand, debug, and maintain.

Introduction

STATPROBE is a contract research organization that performs data management, programming, and statistical report writing services to multiple clients in the pharmaceutical industry. While each client has their own clinical data collection and report standards, similar classes of data are collected for most clinical studies. In addition, most reports are similar in content and format due to FDA regulations for all report submissions. For this reason, STATPROBE is continually developing standard programming methods to be used for all clients and all studies. These standard programming tools require the classification of output that remains constant for all studies, while maintaining the flexibility to accommodate client specific requests.

There are many benefits of standard clinical data reporting procedures, both for STATPROBE and for clients. The advantages for STATPROBE are code reusability, consistent programming techniques, reduced coding and debugging time, and robust validation. Since STATPROBE has already invested the time in designing and fine tuning standard report

templates, the client is not required to spend time generating a format for the tables. However, the macros are designed to remain flexible enough to accommodate client specific requests.

Design of Standard Reporting Tools

The first requirement for the development of standard data reporting tools is the determination of typical report formats. Once this is achieved, templates for clinical data reports are designed. Then, a programming system is created. Finally, the reporting tools are designed and implemented. These reporting tools should complement the abilities and limitations of the programming system.

STATPROBE has developed a report table template used by all reports for every client and protocol (see Appendix 1). Our programming system has evolved to a high level of efficiency and user-friendliness. STATPROBE programs all reports in SAS. An extensive, evolving macro library has been compiled to support our programming system. Many functions and procedures typically performed in report programming are generalized and abstracted to macros. These macros are compiled in a program that establishes a standard environment for each report program. In this way, the code contained in the actual report programs is very clean and extremely efficient. STATPROBE continually develops and maintains standard clinical data reporting tools. DATA _NULL_ has been chosen as the reporting technique. It is the modular skeleton of the DATA _NULL_ that is abstracted from the individual report programs and placed into the **listtool** and **sumtool** macros.

The standard report macros are designed with both modular elements and object-oriented elements. As STATPROBE continues to develop report programming standards, the system evolves and we increase the utilization of object-orientation. This design allows the macros to be invoked by many different types of tables within one study or similar tables across many studies. It also encourages standard coding techniques. Standard coding techniques are beneficial for many reasons. The benefits include increased reusability of code, reduced programming time, more reliable validation, increased readability, and efficient maintenance.

Another goal considered in the design of the **listtool** and **sumtool** macros is user friendliness. To achieve this goal, STATPROBE designed the macros with as few parameters as possible. In addition, the parameters are descriptively named. This is necessary so that a user can invoke the macro and debug the report program without looking at the **listtool** or

sumtool macro. The parameters are chosen to provide only the information necessary to customize the table to the specific study.

Technical Specifications

Both the standard report macros and the individual report programs are designed in a modular fashion. At this point in the development, STATPROBE has developed one listtool and one sumtool macro for each study. These macros are designed to accommodate all types of reports specific to a study. All programming elements common among the report programs are removed from the individual report programs and converted to more generic code that is used for all tables. This code is centralized in the listtool or sumtool macros. Parameters are created to capture any table specific information.

Figure 1 contains an example of a sumtool macro. This example has parameters that are typically defined in a basic report macro. These include the following:

- the name of the input data set (inds)
- the variable list by which the data is sorted (sortvars)
- the names of the macros containing the table-specific header, body, and footer code (subhead, headcond, lefttext, body, foot)
- the table number used to index the program database, which contains all table titles and footnotes (tblnum)
- an indication of whether the data is to be reported by investigator or overall (byinv)

Additional parameters may be specified depending on the degree to which the tables vary within the study.

```
%Macro report(inds=,sortvars=,subhead=,
              headcond=,lefttext=,body=,tblnum=1,
              foot=,byinv=no);

  Data _null_;
    Set &inds end=eof;
    By &sortvars;
    File print ps=65 ls=180 line=line linesleft=1 n=ps
      header=header;
    Retain topline bottline 0;
    %If %upcase(&byinv)=YES %then %do;
      If _n_>1 and first.inv then link bottom;
    %End;
    Link body;
    If eof then link footer;
  Return;

  Header:
    %head(&pgmname,&tblnum,25);
    Put / %If &inds^= and %upcase(&byinv)=YES
      %then %do;
        inv 1.
      %End;
      %Else %do;
        "OVERALL"
      %End;
    / @&c1 %uline
      %&subhead
    / @&c1 %uline;
    topline=line;
    %&lefttext;
    bottline=line;
```

```
%If &headcond^= %then %do;
  %&headcond;
%End;
Return;

Body:
  %&body;
Return;

Bottom:
  Link footer;
  Put _page_ @;
Return;

Footer:
  Put #bottline @&c1 %uline;
  %footer(&tblnum);
  %If &foot^= %then %do;
    %&foot;
  %End;
Return;

Run;
%Mend report;
```

Figure 1: Example of a sumtool report macro

Figure 2 includes sample skeleton code of an individual summary program that invokes the report macro in Figure 1. The modular design is evident since all of the components of the report program are broken down into blocks of code. In this way, the only programming necessary is code specific to the particular table. When the report macro is invoked these modules are called at the appropriate place.

```
%Include "<path>sumtool.mac";
%Macro subhead;
  < column header put statements >
%Mend subhead;

%Macro lefttext;
  < row category text put statements >
%Mend lefttext;

%Macro body;
  < code to output body of report >
%Mend body;

%report(inds=stats,sortvars=protocol dosegrp patno,
        subhead=subhead,body=body,byinv=yes,
        tblnum=&tblnum);

Run;
```

Figure 2: Skeleton code of a summary program

Future Development

In order to further the development of an object-oriented report system, various types of reports and data structures are categorized into classes. Once this is accomplished, report macros are designed to handle all classes of reports without the need to retool a report system for each project. In the object-oriented paradigm, these report macros are the methods. The objects consist of SAS data sets associated with specific report methods.

Conclusion

In conclusion, STATPROBE has developed extensive modular systems for reporting data. This development is in response to the need to have standardized reporting techniques that are dynamic enough to adapt to client's specific requests. Although the object-oriented aspects of STATPROBE's reporting system are in the early stages of development, we will exploit object-oriented concepts on a much higher scale.

Acknowledgments

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. ® indicates USA registrations.

Author's Address

Jennifer Savage
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 ext. 120

E-mail:

- 72700,1075@compuserve.com
- statprob@oeonline.com

Appendix 1: Example of standard report table template

			PROGRAM NAME	DATE
APPENDIX A.1 (PAGE X OF X)				
MAIN TITLE, DESCRIPTION OF STUDY				
TABLE-SPECIFIC TITLE				
COLUMN 1	COLUMN 2	COLUMN 3		
FOOTNOTE:				