Paper PP05

Titles are a Programmer's Best Friend

Title Driven Display Automation

Gregory Nunn, Veramed Limited, Twickenham, United Kingdom

ABSTRACT

When creating outputs for clinical reporting, much of the information a Clinical Programmer requires is contained within the title. An experienced programmer will break the title down into components and use these components to visualise the output. So why don't we use these components to produce the output?

This paper looks at building a bridge between an output title and an existing standardized reporting macro. By breaking the title down into recognisable components that can be used to populate macro variables, we can produce programs which create a 'best guess' at the output, alongside editable SAS code. Furthermore, by utilizing methods such as artificial intelligence, these programs could be adapted over time incorporating changes made to the tables. In this paper we explore the feasibility of these ideas and consider potential benefits and drawbacks.

INTRODUCTION

In clinical reporting, the title is first indication as to what each output should be. The basic structure and content can be determined from the title and an experienced programmer will break the title down into components and use these components to visualise the output. When looking at an entire set of outputs for clinical reporting, a high percentage of them follow a clear structure, especially the basic summary tables. This structure can be broken down into key components, which can then be used to automate our outputs. As programmers, we read these components and are able to deduce not only which dataset is required, but also which parameters, variables and other critical characteristics of the output. However, this is often the last time the title is used within the output production.

This paper will look at how using the words within the title can potentially benefit the programming team and reduce the amount of time spent making outputs. By breaking the titles down into their components, programmers can utilize them to create more generic, adaptable programs.

BREAKING THE TITLE DOWN

GENERAL STRUCTURE

In most studies, consistency is required throughout the titles to allow for comparisons to be made between similar outputs. This consistency allows each title can be broken down into a set of unique components which describe the nature of the table. Although this is not an extensive list, most titles can be made up of up to 6 key components. Not all of these components will be applicable to all tables, but each title must contain the variable to summarise.

An example general structure for the title of summary tables is displayed below. Although this would not allow for all possible titles, many study population and safety titles can conform to this structure.

Summary of {treatment phase} [Variable] {by subgroup} {at/up to time-period} {for population} {for treatment group}

This template can be adapted as desired, but these basic components will remain the same. In more complex studies more components may be required to cover a larger array of titles, but they could be analysed in similar methods described.

COMPONENT BREAKDOWN

Embedded in every title is the variable or set of variables to be summarised. This should have a direct relationship to a dataset, for example "*Vital Signs*" will always directly link to ADVS. In this instance, the type of variables to be summarised can also be deduced as vital signs are usually continuous variables. More complex variables may indicate a subset of dataset, whether that be certain parameters or certain variables within the dataset.

Although selecting the variable from the title programmatically may seem challenging, all that is required is a search for exact phrases which will be defined within the statistical analysis plan (SAP).

```
select;
  when (index(upcase(title),"VITAL SIGNS")) do;
    dataset="ADVS";
    var="AVAL";
  end;
  when (index(upcase(title),"ECG VALUES")) do;
    dataset="ADEG";
    var="AVAL";
  end;
  when (index(upcase(title),"ECG FINDINGS")) do;
    dataset="ADEG";
    var="AVALC";
  end;
  when (index(upcase(title),"AGE RANGES")) do;
    dataset="ADSL";
    var= AGEGR";
  end;
  otherwise %put Unable to deduce variable component. Dataset not set;
end;
```

The code above gives an indication as to how the variable can be selected. Once this has been selected, numerous characteristics can be set, for example the variable within each dataset, as above. Depending on the variable, it can be deduced which type of summary is required, continuous or categorical. Although the above code indicates a one-to-one relationship with a variable, should a title include *"Change from Baseline*" this immediately indicates that either CHG or CHGC should be summarised instead of AVAL or AVALC respectively. Rather than performed multiple checked for both "*Vital Signs*" and "*Change from Baseline in Vital Signs*", a simple check can be performed as below.

```
if (upcase(index(title)), "CHANGE FROM BASELINE") then var=tranwrd(var,"AVAL","CHG");
```

The next most commonly used component is time period, which can come in two forms. For a table which covers a broad range of visits, the title will usually indicate which treatment phase the table should be subset on. Once again, the standard text for study phases will be predefined

with the SAP, so all that is required is a check to see whether a phase is specified and if so which.

```
if (upcase(index(title)), 'PRE-TREATMENT') then aphase = 'PRE-TREATMENT';
else if (upcase(index(title)), 'ON-TREATMENT') then aphase = 'ON-TREATMENT';
else if (upcase(index(title)), 'POST-TREATMENT') then aphase = 'POST-TREATMENT';
else aphase = ''
```

The example above would work for a simple parallel group study with a single treatment period, but it could be easily expanding to include a wider range of treatment phases. However, the study phase is not the only way a specific time period can be indicated. It might be required that only specific visits are required in the table and this might be indicated to two different ways.

up to {Visit X}/ up to {Week Y}
at {Visit X}/ at {Week Y}

Noticeably this type of component contains key words before the component. Rather than directly scanning the title for the component, it would be beneficial to first check for the presence of these key words and then read in the component. By checking for the presence of the key words, a statement can be built to be used later to subset the dataset, as seen below.

```
do i = 1 to countw(title);
  if scan(title, i) = "up" and scan(title, i+1) = "to" then do;
    visits = "lt " || strip(scan(title, i+3));
  end;
  else if scan(title, i) = "at" then do;
    visits = "eq " || strip(scan(title, i+2);
  end;
end;
```

Should a study require subgroup analysis, this follows a similar component structure. However, rather than the subgroup being a single word, this will often be pre-defined phrases in the SAP. For example:

By {Subgroup}
By Age Group
By Sex
By Geographical Region

Rather than looping through the entire title word by word, one could identify whether the title contains the key word, 'By', and then select the subsequent subgroup.

```
if index(title, 'By') then do;
  subgroup = strip(scan(title, 2, 'By'));
  if subgroup = 'Age Group' then byvar = agegr1;
  else if subgroup = 'Sex' then byvar = sex;
  else if subgroup = 'Geographic Region' then byvar = ctrygr1;
end;
else byvar='';
```

Once the subgroup has been defined, multiple attributes can be set for the output. By setting these attributes to null when no subgroup is present will allow for the subgroup tables to run off the same code as the overall table.

The final two components will only contain study specific text from the SAP. In studies with multiple treatment arms, it might be required that treatment group is in the title. Similarly, for studies with multiple analysis population, the title should contain the population to summarise.

However, in both these cases the absence of the component, the table should default to the standard. Although what the default should be may actually differ table to table, by using previously described components this can still be programmatically set, for example:

```
if index(title, 'Per-Protocol') then popfl = 'PPROTFL';
else do;
  if dataset in ("ADVS" ADEG") then popfl = 'SAFFL';
  else if dataset in ("ADSL" "ADDS") then popfl = 'ITTFL;
end;
```

HOW CAN THESE COMPENENTS BE USED

In order to utilise the title, macros will need to be written to scan for each component and then read in the values. By creating a separate macro for each component of concern, the programming team would then be able to group outputs together and use these component macros to make more general, reusable code. Although tables are often grouped together, this would allow for a wider range of tables to run off one macro, therefore reducing programming time as the only update required for each table is the title.

One way of utilising the components, is converting them into global macro variables for use throughout the code. This will then allow the programmer to use them as and when they are required. For subgroup analysis, the values in the title can be converting into macro variables to be used throughout the code as analysis variables. If there are then multiple different tables using subgroup analysis, the attributes which are controlled by the subgroup would be in a central location, increasing the level of consistency across the outputs.

Alternatively, depending on the values of each component, a different pre-defined macro could be called. For instance, if the value of the variable to be summarised indicates a continuous summary the macro could call a generic proc means, utilising the global macro variables previously set. This could be done internally within the component macro, or the name of the macro to be called could be passed back out of the macro to be used later.

LINKING EVERYTHING TOGETHER

Once a macro has been created for each component, automation of the outputs would be possible by linking them together. The most simplistic way of linking each component together would be a wrapper macro which would call the macro associated with each component. This wrapper macro could then be called itself by each output program and would allow the user to use the global macro variables produced.

To allow for greater automation of outputs, a system can be built which would read in a title and output a SAS file which in turn creates a predicted output. This system would need to perform initial checks on the title to confirm that it is in the desired form, i.e. check that the first two words are *"Summary of"* and check for the existent of the components. By outputting an editable SAS file along with an output, an required adjustments could be made following QC or review.

The incorporation of artificial intelligence into the system will produce more accurate outputs and therefore reduce time spent updating any that are produced. By reading any modifications made to the outputs back into the system, these can be interpreted and then applied in any future outputs with similar or identical titles to fine tune the system. This will lead to more consistent and reproducible outputs across studies.

CONCLUSION

The titles of outputs are key to their creation and should usually be defined before programming begins. By creating macros to analyse the title and read in specific components, programmers would be able to reduce the number of output programs required during a study by capitalizing on the standard structure of the titles. This would in turn, lead to more generic, reusable code and could generate a more consistent approach to outputs across studies. By intertwining the simple macros along with the incorporation of AI, systems can be developed to automate the output production process.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Author Name: Gregory Nunn
Company: Veramed Limited
Address: 5th Floor Regal House, 70 London Road
City / Postcode: Twickenham, TW1 3QS, UK
Work Phone: +44 (0) 20 3696 7240
Email: gregory.nunn@veramed.co.uk
Web: www.veramed.co.uk