Paper 076-2018

Never Cut and Paste Results Again:

How to Automate Results with ODS PowerPoint, Metadata and Macros

Ted D. Williams, PharmD, BCPS, Magellan Method

ABSTRACT

How many of our precious hours on this earth are spent cutting and pasting results into PowerPoint presentations? With the SAS® PowerPoint® ODS destination and some basic macro coding, those days are in the past! The system described herein can transform custom results into standard formats requiring no re-coding, no matter how complex the results. A study of Healthcare administrative claims is used to illustrate the aggregation of transactional data into fact tables. SAS® metadata from the built-in DICTIONARY library is then used to create a results dataset (a key-value pair variation). Finally a macro iterates through the results and sends tables and chart to the PowerPoint® ODS in a ready-to-use format.

INTRODUCTION

At the end of any invigorating (not to say frustrating) SAS project, the results must be transformed from datasets into a user-friendly format. This usually involves cutting and pasting results from the results window, excel file, or similar tabular format into PowerPoint. As the programming axiom goes, if you do anything twice, automate it. With the SAS PowerPoint ODS®, cutting and pasting results into PowerPoint can be automated. The PowerPoint ODS is not limited to simple data tables, but can output charts as well as bulleted list of significant results in the underlying data, freeing analysts to do what we do best, interpret data into actionable information.

But how can dissimilar results be automated? Macros are a powerful tool for looping through data and automating repetitive tasks. Leveraging metadata in the DICTIONARY library (specifically the COLUMNS dataset) allows macros to loops through individual variables in a dataset. This allows for a macro which can report on tens of thousands of outcomes. But which variables should the macro report? Two long standing database design tools, fact tables and key-value pairs, provide macros with a standardized format for all sorts of content. Combining the information in the DICTIONARY library with fact tables allows macros to not only automate the output of data, but to select the correct statistical test for the data, as well as the best chart and graphs for the data type.

This paper will demonstrate how SAS programmers with basic macro experience can leverage those skills to send results (tables and charts) directly to PowerPoint using the ODS PowerPoint statement. The power of automating reporting using the DICTONARY.COLUMNS dataset will also be demonstrated. Although much of the data manipulation is performed using PROC SQL, these examples can be easily adapted to use data steps or other dataset manipulation procedures. These techniques can be applied immediately in any environment where PowerPoint is used to share and discuss results. Is there anyone not in that position?

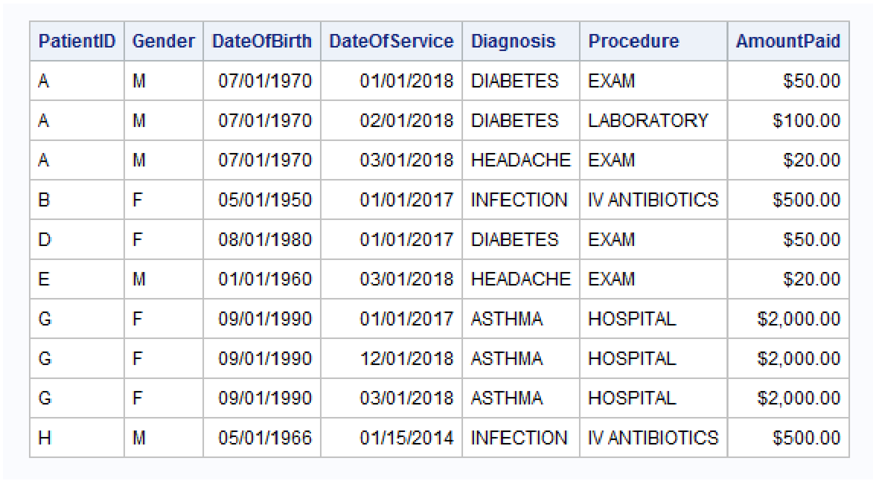
FACT TABLES – EVERYTHING YOU WANTED TO KNOW ABOUT A SINGLE PATIENT

Healthcare data, like much of the Internet of Things (IoT) is stored as transactional records. Transactional dataset have multiple observations for each patient. For example, in Table 1 (created by macro BuildSampleData) Patient “A” has 3 transactions, patient B has 1 transaction, etc. Once the study inclusion and exclusion crtieria have been defined, the transactional data needs to be transformed in to fact tables for analysis.

```
%macro BuildSampleData();

   proc sql ;
         create table Medical (
               PatientID CHAR(100),
               Gender CHAR(1),
               DateOfBirth NUMERIC FORMAT mmddyy10.,
               DateOfService NUMERIC FORMAT mmddyy10.,
               Diagnosis CHAR(100),
               Procedure CHAR(100),
               AmountPaid NUMERIC FORMAT Dollar9.2
         );
         INSERT INTO Medical VALUES
         ("A","M","01JUL1970"d,"01Jan2018"d,"DIABETES","EXAM",50);
         INSERT INTO Medical
         VALUES
         ("A","M","01JUL1970"d,"01Feb2018"d,"DIABETES","LABORATORY",100);
         INSERT INTO Medical VALUES
         ("A","M","01JUL1970"d,"01Mar2018"d,"HEADACHE","EXAM",20);
         INSERT INTO Medical VALUES
         ("B","F","01May1950"d,"01Jan2017"d,"INFECTION","IV
         ANTIBIOTICS",500);
         INSERT INTO Medical VALUES
         ("D","F","01AUG1980"d,"01Jan2017"d,"DIABETES","EXAM",50);
         INSERT INTO Medical VALUES
         ("E","M","01JAN1960"d,"01Mar2018"d,"HEADACHE","EXAM",20);
         INSERT INTO Medical VALUES
         ("G","F","01SEP1990"d,"01JAN2017"d,"ASTHMA","HOSPITAL",2000);
         INSERT INTO Medical VALUES
         ("G","F","01SEP1990"d,"01DEC2018"d,"ASTHMA","HOSPITAL",2000);
         INSERT INTO Medical VALUES
         ("G","F","01SEP1990"d,"01MAR2018"d,"ASTHMA","HOSPITAL",2000);
         INSERT INTO Medical VALUES
         ("H","M","01May1966"d,"15Jan2014"d,"INFECTION","IV
         ANTIBIOTICS",500);
         INSERT INTO Medical VALUES
         ("H","M","01May1966"d,"15Jan2013"d,"DIABETES","EXAM",50);
         INSERT INTO Medical VALUES
         ("H","M","01May1966"d,"15Mar2016"d,"HEADACHE","EXAM",20);
         INSERT INTO Medical VALUES
         ("H","M","01May1966"d,"15JAN2017"d,"ASTHMA","HOSPITAL",2000);
   quit;
   proc sql ;
         create table Pharmacy (
               PatientID CHAR(100),
               Gender CHAR(1),
               DateOfBirth NUMERIC FORMAT mmddyy10.,
               DateOfService NUMERIC FORMAT mmddyy10.,
               DRUG CHAR(100),
               AmountPaid NUMERIC FORMAT Dollar9.
               );
         INSERT INTO Pharmacy VALUES
         ("A","M","01JUL1970"d,"15Jan2018"d,"INSULIN",100);
         INSERT INTO Pharmacy VALUES
         ("A","M","01JUL1970"d,"15Feb2018"d,"INSULIN",100);
```

```
INSERT INTO Pharmacy VALUES
("A","M","01JUL1970"d,"15Mar2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("A","M","01JUL1970"d,"15Apr2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("A","M","01JUL1970"d,"15May2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("A","M","01JUL1970"d,"15Jun2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("C","F","01DEC2000"d,"15Feb2018"d,"BIRTH CONTROL",60);
INSERT INTO Pharmacy VALUES
("C","F","01DEC2000"d,"15MAY2018"d,"BIRTH CONTROL",60);
INSERT INTO Pharmacy VALUES
("C","F","01DEC2000"d,"15AUG2018"d,"BIRTH CONTROL",60);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15MAR2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15MAR2018"d,"BIRTH CONTROL",20);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15APR2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15APR2018"d,"BIRTH CONTROL",20);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15MAY2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15MAY2018"d,"BIRTH CONTROL",20);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15JUN2018"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("D","F","01AUG1980"d,"15JUN2018"d,"BIRTH CONTROL",20);
INSERT INTO Pharmacy VALUES
("G","F","01SEP1990"d,"20JAN2018"d,"INHALER",55);
INSERT INTO Pharmacy VALUES
("G","F","01SEP1990"d,"20FEB2018"d,"INHALER",55);
INSERT INTO Pharmacy VALUES
("G","F","01SEP1990"d,"02JAN2017"d,"INHALER",55);
INSERT INTO Pharmacy VALUES
("G","F","01SEP1990"d,"02FEB2017"d,"INHALER",55);
INSERT INTO Pharmacy VALUES
("G","F","01SEP1990"d,"02MAR2017"d,"INHALER",55);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Jan2014"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Feb2014"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Mar2014"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Apr2014"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22May2015"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Jun2015"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Jan2016"d,"INSULIN",100);
INSERT INTO Pharmacy VALUES
("H","M","01May1966"d,"22Feb2016"d,"INSULIN",100);
```

```
            INSERT INTO Pharmacy VALUES
            ("H","M","01May1966"d,"22Mar2017"d,"INSULIN",100);
            INSERT INTO Pharmacy VALUES
            ("H","M","01May1966"d,"22Apr2017"d,"INSULIN",100);
            INSERT INTO Pharmacy VALUES
            ("H","M","01May1966"d,"22May2018"d,"INSULIN",100);
            INSERT INTO Pharmacy VALUES
            ("H","M","01May1966"d,"2Jun2018"d,"INSULIN",100);
    QUIT;

%mend;

%BuildSampleData();
```

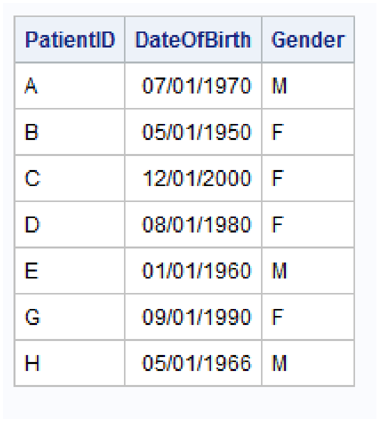


PatientID	Gender	DateOfBirth	DateOfService	Diagnosis	Procedure	AmountPaid
A	M	07/01/1970	01/01/2018	DIABETES	EXAM	$50.00
A	M	07/01/1970	02/01/2018	DIABETES	LABORATORY	$100.00
A	M	07/01/1970	03/01/2018	HEADACHE	EXAM	$20.00
B	F	05/01/1950	01/01/2017	INFECTION	IV ANTIBIOTICS	$500.00
D	F	08/01/1980	01/01/2017	DIABETES	EXAM	$50.00
E	M	01/01/1960	03/01/2018	HEADACHE	EXAM	$20.00
G	F	09/01/1990	01/01/2017	ASTHMA	HOSPITAL	$2,000.00
G	F	09/01/1990	12/01/2018	ASTHMA	HOSPITAL	$2,000.00
G	F	09/01/1990	03/01/2018	ASTHMA	HOSPITAL	$2,000.00
H	M	05/01/1966	01/15/2014	INFECTION	IV ANTIBIOTICS	$500.00

Table 1 - Transactional data with mulitple observations for each subject (PatientID)

Fact tables have been used extensively in data warehouses and provide a convenient way to summarize all transactions from all datasets for each subject. In healthcare studies, static subject characteristics include a patient identifier, date of birth, and gender. The code below creates a unique (DISTINCT) list of members with at least one transaction in either the “Medical” or “Pharmacy” table. The resulting dataset appears in Table 2.

```
%Macro BuildBaseFactTable();
    PROC SQL;
            CREATE TABLE Fact AS
            SELECT DISTINCT PatientID, DateOfBirth , Gender
            FROM Medical
            UNION SELECT  PatientID, DateOfBirth , Gender
            FROM Pharmacy
            ;
    QUIT;
%mend;

%BuildBaseFactTable();
```



PatientID	DateOfBirth	Gender
A	07/01/1970	M
B	05/01/1950	F
C	12/01/2000	F
D	08/01/1980	F
E	01/01/1960	M
G	09/01/1990	F
H	05/01/1966	M

Table 2 – Fact table with one observation per subject (patientID)

DESIGNING A FACT TABLE FOR AUTOMATED RESULTS REPORTING

SAS variable naming requirements present a challenge for automating user-friendly reports. Variable names limited to 32 characters, without spaces, etc. do not make for professional looking presentations. But there is a built-in SAS feature which supports longer names, spaces and numbers: the LABEL property. The LABEL property supports mixed text of up to 256 characters.[1] There are a variety of ways to set the LABEL property using datasteps and PROC SQL. Using the data step ATTRIB statement:

```
ATTRIB DateOfBirth LABEL = "Date Of Birth";
```

A minor downside of using labels is that the obsure the actual variable name when displaying data in the results window. The OPTIONS LABEL; and OPTIONS NOLABEL are extremely helpful when working with labeled variables.[2]

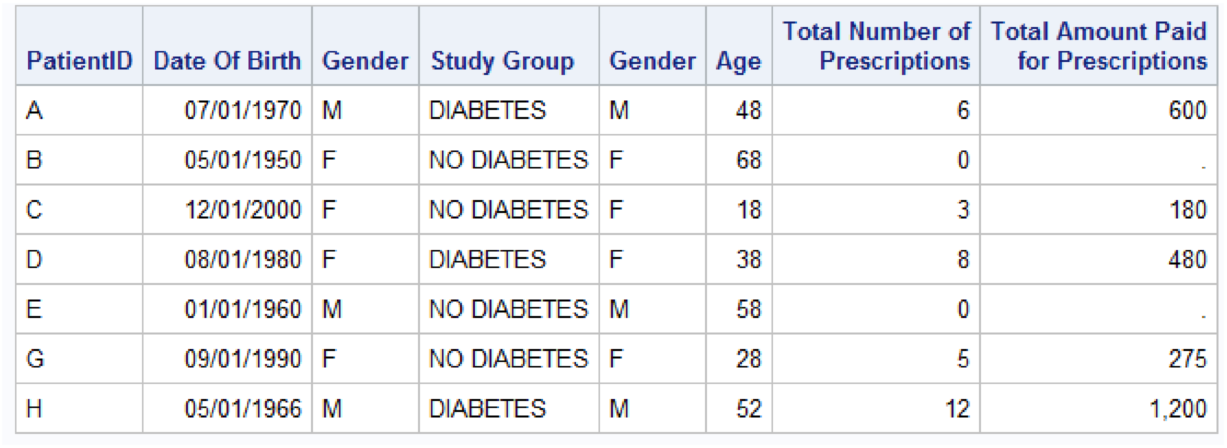
Overloading dataset variable names provides a mechanism for categorizing variables. Overloading, also known as embedding data, is a dubious and ubiquitous practice. Here it is used to classify when and if each variable should be included in the final report. For example, the patient ID is a unique serial number which should not be reported and is not amenable to summarizing. Therefore it should be excluded from the automatic reporting process. Other variables should be clustered together: Gender and age clustered as “baseline” data with counts of transactions and payment amounts clustered as “Follow Up” data. To create these clusters, each variable name will be assigned a prefix of either “BASE_” for baseline or “FLUP_” for follow up data. Variable without one of these prefixes will not be subject to the automated reporting. Any prefix could be used and new prefixes can be added. Although these prefixes are overloading the variable name with an expected value, the prefixes enforce a naming convention which is an accepted programming standard.

Fact tables designed with these considerations enable analysts to create thousands of summary variables. When each of these calculations is encapsulated in an individual macro, a reusable library of outcomes is created and can be shared throughout the team. Simply add or delete a sub-macro call to the AddMeasuresToFactTable macro. The code below illustrate the system and create the fact table listed as table 3.

```
%Macro AddMeasuresToFactTable();

    *Demonstrate LABEL property;
    DATA Fact;
          SET Fact;
          ATTRIB DateOfBirth LABEL = "Date Of Birth";
    RUN;

    %AssignStudyGroup(Dataset=Fact);
    %AssignGender(Dataset=Fact);
    %AssignAge(Dataset=Fact);
    %AssignRxCount(Dataset=Fact);
    %AssignRxPaid(Dataset=Fact);
    %AssignMedPaid(Dataset=Fact);
%mend;

%Macro AssignStudyGroup(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
          SET &Dataset;
          ATTRIB StudyGroup
                LENGTH = $100
                LABEL = "Study Group";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
          UPDATE &Dataset
          SET StudyGroup =
                CASE
                WHEN PatientID IN (
                      SELECT PatientID
                      FROM Medical
                      WHERE Diagnosis = "DIABETES"
                )
                      THEN "DIABETES"
                ELSE "NO DIABETES"
                END
          ;
    QUIT;
%mend;
%Macro AssignAge(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
          SET &Dataset;
          ATTRIB BASE_AGE
                LENGTH = 8
                LABEL = "Age";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
          UPDATE &Dataset
          SET BASE_AGE = INTCK ('year',DateOfBirth, "&SYSDate"d)
          ;
    QUIT;
%mend;
%Macro AssignGender(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
          SET &Dataset;
          ATTRIB BASE_GENDER
                LENGTH = $1
                LABEL = "Gender";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
          UPDATE &Dataset
```

```
            SET BASE_GENDER = Gender
            ;
    QUIT;
%mend;
%Macro AssignRxCount(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
            SET &Dataset;
            ATTRIB FLUP_RxCount
                    LENGTH = 8
                    FORMAT = COMMA9.
                    LABEL = "Total Number of Prescriptions";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
            UPDATE &Dataset f
            SET FLUP_RxCount =
            (
                    SELECT COUNT(AmountPaid)
                    FROM Pharmacy p
                    WHERE  p.PatientID = f.PatientID
            )
            ;
    QUIT;
%mend;
%Macro AssignRxPaid(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
            SET &Dataset;
            ATTRIB FLUP_RxPaid
                    LENGTH = 8
                    FORMAT = COMMA9.
                    LABEL = "Total Amount Paid for Prescriptions";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
            UPDATE &Dataset f
            SET FLUP_RxPaid =
            (
                    SELECT SUM(AmountPaid)
                    FROM Pharmacy p
                    WHERE  p.PatientID = f.PatientID
            )
            ;
    QUIT;
%mend;
%Macro AssignMedPaid(Dataset=);
    *Create Variable with a Label;
    DATA &Dataset;
            SET &Dataset;
            ATTRIB FLUP_MedPaid
                    LENGTH = 8
                    FORMAT = COMMA9.
                    LABEL = "Total Amount Paid for Medical Claims";
    RUN;
    *Populate the Fact table value;
    PROC SQL;
```

```
          UPDATE &Dataset f
          SET FLUP_MedPaid =
          (
                SELECT SUM(AmountPaid)
                FROM Medical p
                WHERE  p.PatientID = f.PatientID
          )
          ;
    QUIT;
%mend;
%AddMeasuresToFactTable();
```



PatientID	Date Of Birth	Gender	Study Group	Gender	Age	Total Number of Prescriptions	Total Amount Paid for Prescriptions
A	07/01/1970	M	DIABETES	M	48	6	600
B	05/01/1950	F	NO DIABETES	F	68	0	.
C	12/01/2000	F	NO DIABETES	F	18	3	180
D	08/01/1980	F	DIABETES	F	38	8	480
E	01/01/1960	M	NO DIABETES	M	58	0	.
G	09/01/1990	F	NO DIABETES	F	28	5	275
H	05/01/1966	M	DIABETES	M	52	12	1,200

Table 3 – Fact table formatted for automation (showing labels)

RESULTS BY STUDY GROUP

After summarizing data for each subject in the fact table, it is time to summarize data for each study group in the results table. Not only will the results dataset sort simple aggregations, but it can store p-values, formatting data, and combine disparate data types in a user friendly format. Most importantly the data from all of the measures calculated above are stored in a format that a macro can use to generate dozens (or hundreds or thousands) of tables, without requiring any recoding. Project specific variable names never appear in processing from here on out. Variable names are transformed into observation values. The power of automation is about to be released!

```
%Macro CreateResultsTable();
    PROC SQL;
          CREATE TABLE Results
          (
                StudyGroup CHAR(100),
                MeasureType CHAR(100),
                MeasureName CHAR(100),
                MeasureField CHAR(100),
                MeasureValue CHAR(100),
                PatientCount NUMERIC,
                PatientPercent NUMERIC,
                Mean NUMERIC,
                Median NUMERIC,
                SD NUMERIC,
                CompoundResult CHAR(100)
          )
          ;
    QUIT;
%mend;
%CreateResultsTable();
```

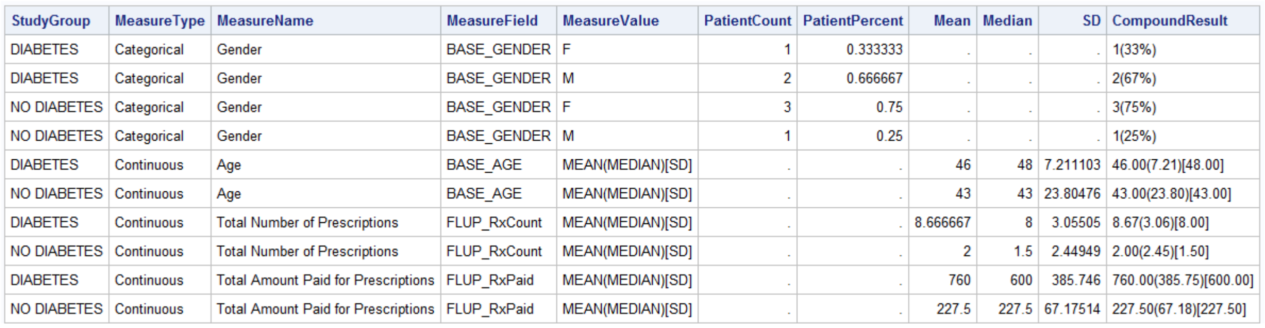
USE METADATA TO AGGREGATE CUSTOM OUTCOMES INTO STANDARD RESULTS

The built in DICTIONARY.COLUMNS dataset contains a list of all of the properties of all of the variables in all of the datasets in all of the active libraries. The name, label, format, and type of each variable in the fact table can be programmatically accessed and use to perform any desired: aggregations, statistical tests, etc. The simple example below, loops through each variable, determines the data type (categorical or numeric), and performs appropriate aggregations (counts for categorical data, measures of variance for numeric data). A production system can (and should) make distinctions between dates and currency, ordinal and nominal data, parametric vs. non-parametric data, etc. P-values can be calculated based on these distinctions and the results stored in the results table.

```
%macro PopulateResultsTable(
    FactLib=WORK,
    FactDS=FACT
);
    *****
    Local Variable Declaration
    ******;
    %Local MeasureCounter;
    %local ctr;

    %CreateResultsTable();

    *Create arrays of macro variables from Dictionary.Columns;
    proc sql ;
          SELECT Name, Type
          INTO
                :MeasureVariable1-:MeasureVariable9999,
                :MeasureType1-:MeasureType9999
          FROM DICTIONARY.Columns
          WHERE LibName = "&FactLib" and MemName = "&FactDS"
          and (Name like "BASE_%" or Name like "FLUP_%")
          ;
    quit;

    %let MeasureCounter = &SqlObs;

    *************************
    Loop through each measure and aggregate as necessary.
    *******************************
    You can consider the data type, the variable format,
    Check the distribution of the data, etc.
    For illustrative purposes, only the data type is checked
    ********************************
    ;
    %DO ctr = 1 %to &MeasureCounter;
          %IF &&MeasureType&ctr = char %then
          %do;
                %AddCategoricalResults(
                      Library=&FactLib,
                      Dataset=&FactDS,
                      Variable=&&MeasureVariable&ctr
                );
          %end;
          %else %if &&MeasureType&ctr = num %then
          %do;
                %AddContinuousResults(
                      Library=&FactLib,
                      Dataset=&FactDS,
                      Variable=&&MeasureVariable&ctr
                );
          %end;
          %else
          %do;
                *Unexpected data type encountered;
          %end;
    %end;
%mend;

%macro AddCategoricalResults(Library=,Dataset=,Variable=);

    *Find the metadata for the specified variable;
    proc sql ;
          SELECT Name, Type, Format, Label
          INTO :MeasureVariable, :MeasureType, :MeasureFormat,:MeasureLabel
          FROM DICTIONARY.Columns
```

```
            WHERE
                  LibName = "&Library" and
                  MemName = "&Dataset" and
                  Name ="&Variable"
            ;
      quit;
      *You can perform any additional manipulations, such as P values;

      *Aggregate as appropriate for the type, format etc.;
      PROC SQL;
            INSERT INTO Results (
                  StudyGroup, MeasureType,
                  MeasureValue, MeasureName, MeasureField,
                  PatientCount,PatientPercent, CompoundResult
            )
            SELECT
                  StudyGroup,
                  "Categorical" as MeasureType,
                  &MeasureVariable as MeasureValue,
                  "&MeasureLabel" as MeasureName,
                  "&MeasureVariable" as MeasureField,
                  COUNT(DISTINCT PatientID) as PatientCount,
                  /*Add a Percentage*/
                  CALCULATED patientcount /
                  (
                        SELECT COUNT(DISTINCT PatientID)
                        FROM FACT b
                        where a.StudyGroup = b.StudyGroup
                  ) as PatientPercent,
                  COMPRESS(
                        PUT(calculated PatientCount,comma8.) ||
                        "(" ||
                        PUT(calculated PatientPercent,Percent8.)
                        || ")"
                  ) as CompoundResult
            FROM &Library..&Dataset a
            WHERE &MeasureVariable is not null
            GROUP BY StudyGroup, &MeasureVariable
            ;
      QUIT;
   %mend;

   %macro AddContinuousResults(Library=,Dataset=,Variable=);
      *Find the metadata for the specified variable;
      proc sql ;
            SELECT Name, Type, Format, Label
            INTO :MeasureVariable,:MeasureType,:MeasureFormat,:MeasureLabel
            FROM DICTIONARY.Columns
            WHERE
                  LibName = "&Library" and
                  MemName = "&Dataset" and
                  Name ="&Variable"
            ;
      quit;
      *You can perform any additional manipulations, such as P values;

      *Aggregate as appropriate for the type, format etc.;
```

```
    PROC SQL;
          INSERT INTO Results (
                StudyGroup, MeasureType,
                MeasureName, MeasureField,
                Mean, Median, SD,
                CompoundResult,MeasureValue
          )
          SELECT
                StudyGroup,
                "Continuous" as MeasureType,
                "&MeasureLabel" as MeasureName,
                "&MeasureVariable" as MeasureField,
                AVG(&MeasureVariable) as Mean,
                MEDIAN(&MeasureVariable) as Median,
                STD(&MeasureVariable) as SD,
                COMPRESS(
                      PUT(calculated Mean,comma8.2) ||
                      "("|| PUT (CALCULATED SD, comma8.2)|| ")"||
                      "["|| PUT (CALCULATED Median, comma8.2) ||"]"
                ) as CompoundResult,
                "MEAN(MEDIAN)[SD]" as MeasureValue
          FROM &Library..&Dataset
          GROUP BY StudyGroup
          ;
    QUIT;
%mend;
%PopulateResultsTable();
```



StudyGroup	MeasureType	MeasureName	MeasureField	MeasureValue	PatientCount	PatientPercent	Mean	Median	SD	CompoundResult
DIABETES	Categorical	Gender	BASE_GENDER	F	1	0.333333	.	.	.	1(33%)
DIABETES	Categorical	Gender	BASE_GENDER	M	2	0.666667	.	.	.	2(67%)
NO DIABETES	Categorical	Gender	BASE_GENDER	F	3	0.75	.	.	.	3(75%)
NO DIABETES	Categorical	Gender	BASE_GENDER	M	1	0.25	.	.	.	1(25%)
DIABETES	Continuous	Age	BASE_AGE	MEAN(MEDIAN)[SD]	.	.	46	48	7.211103	46.00(7.21)[48.00]
NO DIABETES	Continuous	Age	BASE_AGE	MEAN(MEDIAN)[SD]	.	.	43	43	23.80476	43.00(23.80)[43.00]
DIABETES	Continuous	Total Number of Prescriptions	FLUP_RxCount	MEAN(MEDIAN)[SD]	.	.	8.666667	8	3.05505	8.67(3.06)[8.00]
NO DIABETES	Continuous	Total Number of Prescriptions	FLUP_RxCount	MEAN(MEDIAN)[SD]	.	.	2	1.5	2.44949	2.00(2.45)[1.50]
DIABETES	Continuous	Total Amount Paid for Prescriptions	FLUP_RxPaid	MEAN(MEDIAN)[SD]	.	.	760	600	385.746	760.00(385.75)[600.00]
NO DIABETES	Continuous	Total Amount Paid for Prescriptions	FLUP_RxPaid	MEAN(MEDIAN)[SD]	.	.	227.5	227.5	67.17514	227.50(67.18)[227.50]

Table 4 – Results table

A few control variables were added to help during the generation of the PowerPoint file. The data type specific results are converted into a character variable “CompoundResult.” This step eliminates the need to re-determine the data types and values at reporting time. Various formats can be assigned depending on the original data type and format. The MeasureType variable will be used to select a chart type appropriate for the data type.

SEND REPORTS DIRECTLY TO ODS POWERPOINT

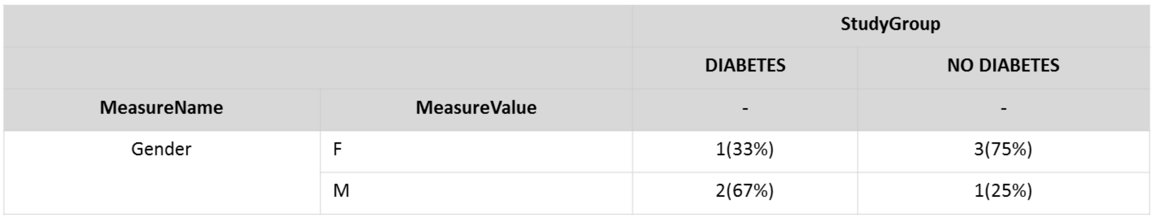
Although outputting to ODS PowerPoint is fairly straightforward, there are some nuances. The simplest way to output results to PowerPower is:

```
ODS PowerPoint ;
    /*Any PROC sending output to the results window*/
ODS PowerPoint Close;
```

Although this framework generates PowerPoint slides, it lacks the nuanced formatting required for professional presentations. Controlling the format of tables and charts via the style properties of PROC REPORT and PROC SGPLOT is the most consistent and specific method. Although the PowerPoint ODS does have style, there are some odd behaviors when these styles interact with these other PROCs.[3] For example, if the font size is set using the PowerPoint ODS, a blank cell in PROC REPORT will not use the PowerPoint ODS Style value, but rather the default text size causing oddly large empty cells. For outputting tables and charts, the built-in layout property value of TitleAndContent does a nice job of aligning the objects and allows PowerPoint tables to adapt gracefully if the slide layout is change, such as applying corporate styles after creation. Setting the PaperSize to 10x5.63 create a wide screen layout.[4] When only outputting table via PROC REPORT, a new page is created after each table. PROC GCHART does not create a new page, so a ODS PowerPoint StartPage=now; statement must be used. Finally, the most troublesome features is how SAS handles wide tables. If the table contents are too wide for the table or slide, the table is split, creating table which are generally not appropriate for client presentations. The simplest way to manage this is with the font size values in PROC REPORT as in the example below.

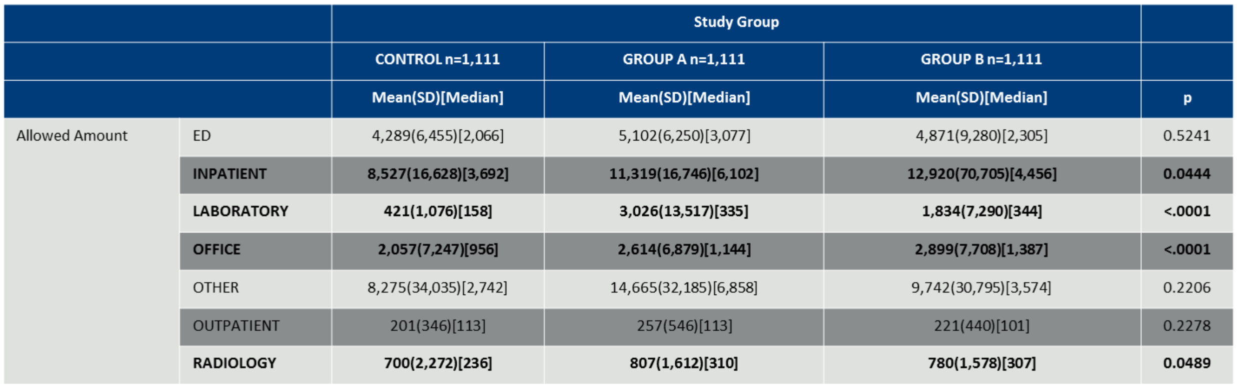
A macro can now be used to generating PowerPoint tables and charts by looping through the contents of the results dataset. The generation of the table using PROC REPORT is encapsulated into a separate macro and can be reused for other purposes beyond PowerPoint generation (Table 5). This is particularly helpful as the formatting of the report becomes more sophisticated (Table 6). The MeasureType variable is used to select the appropriate chart type, bar charts for numeric data, pie charts for categorical data. By defining different MeasureType values in the creation of the results table, more chart options can be added to SendResultsToPPTX using additional %ELSE %IF blocks.

```
%macro SendResultsToPPTX(
    AnyValidFilePath=
);

    %Local MeasureCount;
    %local ctr;


    *Get a list of results and generate the
    appropriate powerpoint elements
    tables, graphs, anything you can
    send to an ODS
    ;
    Proc sql NOPRINT ;
          select distinct
                MeasureName,
                UPCASE( MeasureType)
          INTO :MeasureName1-:MeasureName9999,
          :MeasureType1-:MeasureType9999
          From Results
          ORDER BY 1
          ;
    quit;
    %LEt MeasureCount = &SQLObs;
    %if &MeasureCount >0 %then
    %do;
          *Use papersize to create a wide screen slide;
          *Use built in layout TitleAndContent - makes changing to
           Other styles (e.g. corporate template) easier;
          ODS PowerPoint
                File = &AnyValidFilePath
                STARTPAGE=ON
                layout=TitleAndContent
                ;
                options papersize=(10in 5.63in);
          %Do ctr = 1 %to &MeasureCount;
                Title "&&MeasureName&ctr";
                %ReportDataTable(MeasureName=&MeasureName&ctr);
                %if &&MeasureType&ctr = CATEGORICAL %then
                %do;
                      %CategoricalReport(MeasureName=&MeasureName&ctr);
                %end;
                %else %if &&MeasureType&ctr = CONTINUOUS %then
                %do;
                      %ContinuousReport(MeasureName=&MeasureName&ctr);
                %end;
                *Charts require manually creating a new page;
                ODS Powerpoint StartPage=now;
          %end;
          ODS PowerPoint Close;
    %end;
%mend;
%macro CategoricalReport(MeasureName=);
    *
          Get the number of study groups for the GChart
          ACROSS property
    ;
    PROC SQL NOPRINT;
          SELECT count(distinct StudyGroup)
          INTO :StudyGroupCount
          FROM Results
          WHERE MeasureName ="&MeasureName"
          ;
    QUIT;


    /*Requires SAS/Graph*/
    PROC GCHART DATA=Results(WHERE=(MeasureName="&MeasureName"));
          TITLE "Categorical Data - &MeasureName";
          PIE MeasureValue /
                SUMVAR=PatientCount
                GROUP= StudyGroup
                ACROSS=&StudyGroupCount
                PERCENT=ARROW
                NOHEADING/*suppresses Sum By Variable subheading*/
                ;
    RUN;
    /*Clean up the title*/
    Title;
%mend;
%macro ContinuousReport(MeasureName=);

    /*Requires SAS/Graph*/
    PROC SGPLOT DATA=Results(WHERE=(MeasureName="&MeasureName"));
          TITLE "Continuous Data - &MeasureName";
          VBAR MeasureValue  /*X Axis Bar Groupings */ /
                RESPONSE=Mean /*Y axis Values*/
                GROUP= StudyGroup/*Individual Bars*/
                GROUPDISPLAY=CLUSTER
                STAT=MEAN
          ;
          XAXIS DISPLAY=(NOLABEL NOTICKS);
          YAXIS GRID LABEL= "Mean &MeasureName";
    RUN;
    /*Clean up the title*/
    Title;
%mend;
%macro ReportDataTable(MeasureName=,FontSize=12pt);
    *Although there are ways to control
    font size and position with styles,
    controlling in the table works well
    and controls the cell size even when
    there is no data;
    PROC REPORT
          DATA=Results (WHERE=(MeasureName="&MeasureName"))
          MISSING
          SPANROWS
          style(report)=[
                width=9.5in
          ]
          Style(column)=[
                just=center
                font_size=&FontSize
          ]
          style(header)=[
                just=center
                font_weight=bold
                font_size=&FontSize
          ]
          ;
          COLUMN MeasureName MeasureValue StudyGroup,CompoundResult  ;
          DEFINE CompoundResult /DISPLAY '-';
          DEFINE StudyGroup /ACROSS ;
          DEFINE MeasureValue/Group Style(Column)=[just=left] ;
          DEFINE CompoundResult/GROUP ;
          DEFINE MeasureName  / Group ;
    RUN;
%mend;

%SendResultsToPPTX(AnyValidFilePath="%sysfunc(pathname(work))\WUSS.pptx");
```



		StudyGroup	
		DIABETES	NO DIABETES
MeasureName	MeasureValue	-	-
Gender	F	1(33%)	3(75%)
	M	2(67%)	1(25%)

Table 5 – PowerPoint table for categorical data



		Study Group			
		CONTROL n=1,111	GROUP A n=1,111	GROUP B n=1,111	
		Mean(SD)[Median]	Mean(SD)[Median]	Mean(SD)[Median]	p
Allowed Amount	ED	4,289(6,455)[2,066]	5,102(6,250)[3,077]	4,871(9,280)[2,305]	0.5241
	INPATIENT	**8,527(16,628)[3,692]**	**11,319(16,746)[6,102]**	**12,920(70,705)[4,456]**	**0.0444**
	LABORATORY	**421(1,076)[158]**	**3,026(13,517)[335]**	**1,834(7,290)[344]**	**<.0001**
	OFFICE	**2,057(7,247)[956]**	**2,614(6,879)[1,144]**	**2,899(7,708)[1,387]**	**<.0001**
	OTHER	8,275(34,035)[2,742]	14,665(32,185)[6,858]	9,742(30,795)[3,574]	0.2206
	OUTPATIENT	201(346)[113]	257(546)[113]	221(440)[101]	0.2278
	RADIOLOGY	**700(2,272)[236]**	**807(1,612)[310]**	**780(1,578)[307]**	**0.0489**

Table 6 – Example of PowerPoint table with additional formatting controls, including study group counts in column heads, banded rows, and bolding of significant values (code not included)

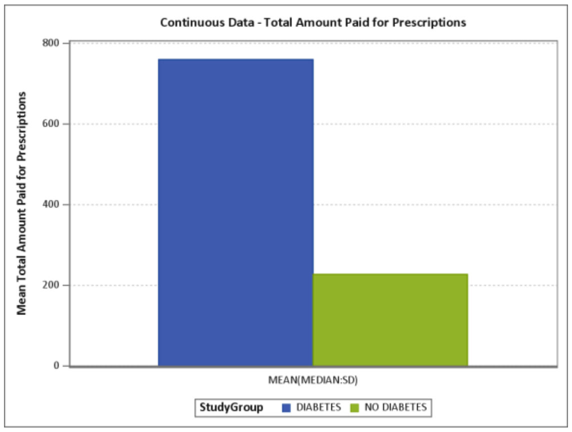


Figure 1 – PowerPoint chart for numeric data

CONCLUSION

The SAS ODS PowerPoint destination, when combine with macro automation, can generate complex, professional looking presentations. Using the built in DICTIONARY.COLUMNS dataset enables SAS programs to automatically generate results table to feed into the ODS PowerPoint system. These two technique can dramatically reduce the most common repetitive tasks associated with data analysis and free analysts to interpretive data.

REFERENCES

1. SAS Variables: SAS Variable Attributes. http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a001103996.htm. Accessed July 5, 2018.

2. SAS System Options: LABEL System Option - 9.2. http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000279126.htm. Accessed July 10, 2018.

3. Eslinger J. The Dynamic Duo: ODS Layout and the ODS Destination for PowerPoint. In: *Proceedings of the*. ; 2016.

4. Hunter T. A First Look at the ODS Destination for PowerPoint. In: *Proceedings of the 2013 SAS Global Forum Conference*. Citeseer; 2013.

ACKNOWLEDGMENTS

I would like to thank Michael Polson of Magellan Method for perennially supporting me optimize my SAS programs. Anne Kangethe, Donna Moore and Servi Barrientos provided invaluable assistance in editing and vetting this paper. Thanks to Art Carpenter for his mentoring and sage advice in preparing this paper. Finally, I would like to thank the volunteers of Western Users of SAS Software (WUSS) for the opportunity to present this paper.

RECOMMENDED READING

CONTACT INFORMATION

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

Ted D. Williams
Magellan Method
88 Silva Lane, Tech 4, Middletown, RI 02842
314-387-4305
tdwilliams1@magellanhealth.com