Paper 1300-2017

Reporting Rock Stars or Robots, They'll Never Know:

Using SAS to Automate Report Distribution

Jacob Price, Baylor University

ABSTRACT

Do you ever feel like your office emails the same reports to the same people over, and over and over again? If your customers are anything like mine, you create reports, and lots of them. Our office is using macros, SAS email capabilities and other programming techniques, in conjunction with our trusty contact list, to automate report distribution. Customers now receive the data they need, and only the data they need, on the schedule they request. In addition, not having to send these emails out manually saves our office valuable time and resources that can be used for other initiatives. In this session, we will walk through a few of the SAS techniques we are utilizing to provide better service to our internal and external partners and, hopefully, make us look a little more like rock stars.

INTRODUCTION

The Office of Institutional Research and Testing (IRT) conducts research in order to provide information that supports institutional planning, policy formulation, decision-making, and external reporting at Baylor University. Our customers often ask us to provide this information in the form of reports sent through email, with the report attached as a PDF or an EXCEL file. This allows the recipients to disseminate the information, in a widely used format, to their internal and external partners around campus as they see fit. Unfortunately, this process is slow and occupies time and resources that could be used to complete other requests.

For security purposes, most of our academic departments on campus only have access to record-level data regarding their own areas. To meet this security standard in the past, IRT has had to either manually split out the report data for each department prior to sending, or, more frequently, write and maintain multiple versions of the same program in SAS for each area requesting the information. Then we were manually writing each email and sending them out to the appropriate customers from our office contact list. This, as you can imagine, can become quite tedious as the number of requestors continues to grow for these reports.

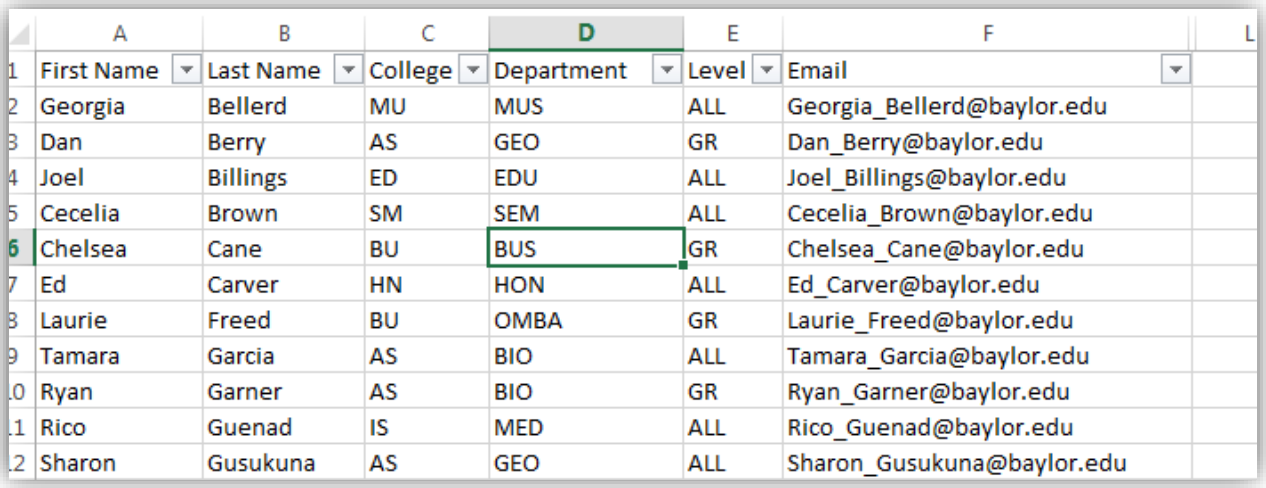
For other, less sensitive report information, we have previously sent multiple offices a cumulative report each time it is run. Invariably, there have been times when there was not relevant data for everyone who receives the report for a given distribution. This leads to people contacting us asking if their information is missing or ignoring the report the next time it is sent, assuming nothing is on there for them to review.

Many times, these requests are time sensitive and need to follow a specific distribution schedule to be as useful and pertinent as possible for our customers. To add another level of difficulty to this, not every customer requests data on the same schedule, even when they request the same reports that we send to other areas. Maintaining multiple delivery schedules without issue can be very taxing and was not a sustainable solution long-term.

Luckily, SAS offers solutions that are helping our office overcome all of these obstacles and make report distribution much more efficient and effective. We are using the SAS macro language, in conjunction with SAS Output Delivery System (ODS) capabilities and the FILENAME statement to deliver our data and fill our customers' business needs. The following paper outlines, in detail, the steps we take to accomplish this task.

RETHINKING THE OLD CONTACT LIST

If your office is like ours, you probably have a spreadsheet or paper list (no judgement here) of people that you communicate with on a regular or semi-regular basis that includes their basic contact information. It probably looks something like this:



	A	B	C	D	E	F	L
1	First Name	Last Name	College	Department	Level	Email	
2	Georgia	Bellerd	MU	MUS	ALL	Georgia_Bellerd@baylor.edu	
3	Dan	Berry	AS	GEO	GR	Dan_Berry@baylor.edu	
4	Joel	Billings	ED	EDU	ALL	Joel_Billings@baylor.edu	
5	Cecelia	Brown	SM	SEM	ALL	Cecelia_Brown@baylor.edu	
6	Chelsea	Cane	BU	BUS	GR	Chelsea_Cane@baylor.edu	
7	Ed	Carver	HN	HON	ALL	Ed_Carver@baylor.edu	
8	Laurie	Freed	BU	OMBA	GR	Laurie_Freed@baylor.edu	
9	Tamara	Garcia	AS	BIO	ALL	Tamara_Garcia@baylor.edu	
10	Ryan	Garner	AS	BIO	GR	Ryan_Garner@baylor.edu	
11	Rico	Guenad	IS	MED	ALL	Rico_Guenad@baylor.edu	
12	Sharon	Gusukuna	AS	GEO	ALL	Sharon_Gusukuna@baylor.edu	

We will take this information and create a SAS dataset out of it. This allows us to set up automated report distribution based on a customer's college, department, and report preferences.

First, we will add IRT staff to the contacts spreadsheet so that they will be in the new dataset. Then variables are added to the spreadsheet to serve as flag indicators for report delivery and to add HTML formatted email signatures for IRT staff to emails. Then 'Y' values can be entered for all of our customers and IRT staff that are going to receive the associated reports denoted by each new flag variable.

The reason this is done in the spreadsheet for this instance rather than in the data set in SAS is to leave the list available to non-SAS users in the office so that they can continue to make updates to the list. Then, the next time the program is run it will update our SAS datasets with any contact changes.

Now we import the contacts list spreadsheet into a permanent SAS library using the PROC IMPORT procedure.

SAS Code:

```
proc import out = present.contacts_list
        datafile = "k:\jacob price\contacts.xlsx"
        dbms = excel replace;
    range = "contacts";
    getnames = yes;
    mixed = no;
    scantext = yes;
    usedate = yes;
    scantime = yes;
run;
```

The final dataset looks like this:

	First_Name	Last_Name	College	Departmen	Level	Email	Signature	Course_Errors	Missing_Instructor	Records_Errors	Studer
15	Marilyn	Hicks	SW	SWO	ALL	Marilyn_Hicks@baylor.edu				Y	Y
16	Stephanie	Hollis	NU	NUR	ALL	Stephanie_Hollis@baylor.edu			Y	Y	Y
17	Charles	King	AS	PSYN	ALL	Charles_King@baylor.edu		Y	Y		
18	Alfred	Lemon	HN	HON	ALL	Alfred_Lemon@baylor.edu					Y
19	Jerri	Maize	LW	LAW	ALL	Jerri_Maize@baylor.edu		Y	Y	Y	Y
20	Chava	McNair	AS	PHY	ALL	Chava_McNair@baylor.edu		Y			
21	Mary	Pinscher	BU	OMBA	GR	Mary_Pinscher@baylor.edu					Y
22	Nina	Porter	AS	PSYN	ALL	Nina_Porter@baylor.edu		Y			
23	Heather	Preen	AS	ENV	ALL	Heather_Preen@baylor.edu		Y			
24	Jacob	Price	IR	IRT	ALL	Jacob_Price@baylor.edu	 <font color = "009900" font face = "Californian FB" p style = "font-size:14pt"> Jacob Price <font color = "003366" p style = "font-size:11pt"> Institutional Research Baylor University One Bear Place #97032 Waco, TX				
25	Lori	Puenney	GR	Graduate	ALL	Lori_Puenney@baylor.edu				Y	Y

After my data is imported, my contacts list is in data set form and is capable of directing reports to our customers. It also comes in handy if a point of contact leaves or retires because we will know, immediately, all of the reports for which IRT is going to need a new contact.

PREPARING THE REPORTING DATA WITH MACRO VALUES

For this demonstration, we are creating and emailing the Missing Instructors Report. The purpose of this report is straightforward. It contains a list of courses that don't have instructors assigned to them yet and need to have one added. We create and send it out multiple times each semester to a wide audience around campus. The number of recipients varies from one run of the report to another because we only want to send the report out to the Colleges that have courses on the list. Using a SAS Macro, we can split up the data and create a spreadsheet for each individual college that contains only their courses.

We start with the already created dataset, "missing_instructors":

VIEWTABLE: Present.Missing_instructors

	CRN	SECTION_NUMBER	COLLEGE	SUBJECT	COURSE_NUMBER	TITLE	ACTUAL_ENROLLMENT
1	31229	01	AS	ANT	3310	Intro Lang & Linguistics	3
2	36229	02	AS	ART	1300	Intro to Art (N-Major) AAI	141
3	19862	02	AS	ART	1310	Drawing I	15
4	35131	02	AS	ART	4100	Field Studies Art History AAI	9
5	32441	02	AS	ART	4321	Advanced Painting	2
6	36862	24	AS	BIO	1106	Mod Concp Bioscience Lab	28
7	12292	B	AS	BIO	3422	Human Physiology	23
8	12299	C	AS	BIO	3422	Human Physiology	23
9	12314	E	AS	BIO	3422	Human Physiology	24

First, begin by to defining our new macro that we will call ***reports_and_emails***.

SAS Code:

```
%macro reports_and_emails;
```

- Note: the code contained within a macro is not color-coded the same way it is normally in the enhanced editor in SAS. It is displayed in this paper with color-coding for demonstration purposes only.

The first thing we do in the ***reports_and_emails*** macro is to count how many records (courses sections, in this case) are actually in "missing_instructors" dataset. Using PROC SQL, the macro variable, &missing_count, is set to the number of observations found.

SAS Code:

```
/* Count the number of sections with no instructor. */
title1 'Current Number of Sections with no Instructor Assigned.';
proc sql;
    select nobs
    into :missing_count
    from sashelp.vtable
    where libname ='PRESENT'
        and memname = 'MISSING_INSTRUCTORS'
    ;
quit;
```

SAS Results Viewer:

Current Number of Sections with no Instructor Assigned.

Number of Physical Observations
157

The &missing_count macro variable serves two purposes:

1. First, it gives me an overall total of how many courses still need instructors added, which I am asked about frequently in the IRT office.
2. Second, this macro variable is used in a conditional statement that determines whether the program continues to create the reports.

Then, we use a conditional statement and the &missing_count macro variable we have created to determine whether to proceed with report creation. We only want to continue if there are records in the dataset.

SAS Code:

```
/* If there are sections with no instructor, move forward. */
%if &missing_count > 0
    %then %do;
```

Once the macro is called and the condition is met (in this case, there are records), the program will compile and execute the code within the %THEN %DO block of code. If the condition is not met (no records), the program vector will move on to the code that follows the %END statement.

Then, we create a temporary copy of the data that is sorted in the way it is needed for the reports.

SAS Code:

```
/*  Sort the data for distribution. */
proc sort data = present.missing_instructors
    out = missing_instructors;
    by college
        subject
        course_number
        section_number;
run;
```

Next, we create a data set called “colleges” in the WORK library that contains each of the Colleges that appear in our data and are going to receive the Missing Instructors Report. We use the first.college variable that is created with the BY statement to only keep one record of each value in the column. This only works because we have previously sorted the data by the “college” variable in the earlier PROC SORT. We also create and populate a “count” variable that will be used later in our process to separate our data for the individual reports.

SAS Code:

```
/*  Create a macro variable of the total number of
colleges missing instructors. */
data colleges;
    set missing_instructors
        (keep = college);
    by college;
    if first.college;
    count + 1;
run;
```

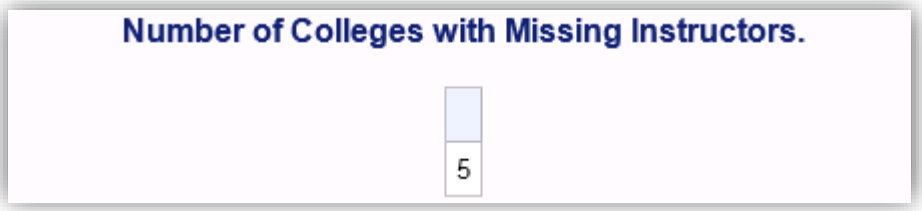
From the “colleges” data set we create a macro value of the total number of unique “college” values in the data, again using PROC SQL.

SAS Code:

```
title1 'Number of Colleges with Missing Instructors.';
proc sql;
    select max(count)
        into :college_count
    from colleges
    ;
quit;
```

- Note: all macro variables created within the reports_and_emails macro are local macros and will not exist outside of the macro unless specifically stated in the program code.

SAS Results Viewer:



Again, this &college_count macro variable serves two purposes:

1. This tells us how many different Colleges have courses with missing instructors.
2. It will be used to set the stop value of an iterative %DO loop later in our code.

CREATING MACRO VARIABLES FOR CONTACTS

Once the "missing_instructors" data set is sorted and the "colleges" data set and &college_count macro variable are created within our ***reports_and_emails*** macro, we are ready to create the following macro variables that will be used to email our reports:

1. &irt_name – The first name of the contact in IRT that is sending the reports.
2. &irt_email – Their email address.
3. &irt_signature – Their email signature.
4. &contact_name – The point of contact first name(s). There can be more than one person.
5. &contact_email – The email address for the report recipient(s).
6. &college – The College for which they are a point of contact.

The internal contact information (those macro variables beginning with "irt") will be static for every email that is sent because each email will go out from the same sender. Once created, these macro variables will remain the same value throughout the ***reports_and_emails*** macro execution. However, the customer (beginning with "contact") and "college" macro variables are dependent on what college we are currently emailing.

The IRT macro values are created first using PROC SQL.

SAS Code:

```
proc sql noprint;
    select first_name,
        compress("'"||email||"'"),
        "'"||signature||"'"
    into :irt_name,
        :irt_email,
        :irt_signature
    from present.contacts_list
    where department = 'IRT'
        and first_name = 'Jacob'
    ;
quit;
```

SAS Results Viewer:

IRT_Name	IRT_Email	IRT_Signature
Jacob	'Jacob_Price@baylor.edu'	' <font color = "009900" font face = "Californian FB" p style = "font-size:14pt"> Jacob Price <font color = "003366" p style = "font-size:11pt"> Institutional Research Baylor University One Bear Place #97032 Waco, TX'

- Note: the NOPRINT option above will keep this data from printing to the results viewer. It is printed here with variable names for demonstration only.

From here, we begin an iterative %DO loop by creating an index macro variable, &i, that will be used to create the customer and college macro values.

SAS Code:

```
%do i = 1 %to &college_count.;
```

We use the &college_count macro variable that was created earlier in our code as the stop for the %DO loop's index variable so that the loop only runs as many times as there are unique "college" values in the data. In this instance, the value of &college_count is five, so the loop will run with values starting at one and continuing to five (with a default interval of one unless you specify otherwise) for &i and then stop processing when the value of &i reaches six.

SAS Code:

```
/*  Set contact email and college macro values. */
title1 'Contacts by College';
proc sql;
    select unique compress("'"||a.email||"'") 'Contact_Email',
        a.college
            format = $unit.
    into:contact_email separated by ' ',
        :college
    from present.contacts_list a
        inner join (missing_instructors b
                        inner join colleges c
                            on b.college = c.college)
            on a.college = c.college
    where missing_instructor = 'Y'
        and c.count = &i.
    ;
quit;
```

The PROC SQL code takes the "missing_instructors" data set and the newly created "colleges" data set and joins them together with an INNER JOIN where the "college" variable has the same value in both

data sets. This is important because it adds the “count” variable that was created earlier to each observation in our “missing_instructors” data sets. Then the “contacts_list” data set is joined to this data with an INNER JOIN, again using the value of “college” on each data set as the join variable. The WHERE clause is used to pull only rows of the “contacts_list” data set that are flagged to receive the Missing Instructor Report. The second piece of the WHERE clause is set to only select the “count” value that is equal to the &i value during the current iteration of the %DO loop.

Example: If this is the first iteration of the loop, then &i = 1 and we will only pull contacts that are from the Art and Sciences (AS).

SAS Results Viewer:

Contacts by College

Contact_Email	College	Count
'Charles_King@baylor.edu'	AS	1
'Lisa_Willis@baylor.edu'	AS	1
'Paulette_Reinhaus@baylor.edu'	AS	1

CREATING THE CUSTOM REPORTS

Now that we have our contact macro variable set we begin to use the &college macro variable to create our report. In this case, the Missing Instructor Report is a spreadsheet that we will create from the “missing_instructors” data set using ODS EXCEL.

First, use a %LET statement to create a macro variable for the .xlsx file name.

SAS Code:

```
%let out = "K:\Jacob Price\Presentations\2017\Missing Instructors - &college..xlsx";
```

We will use the &college macro variable in the file name so that each spreadsheet that is created by our loop will have a different name and will not overwrite the spreadsheet created in previous iterations. When the program finishes we will have a file for every report we create in our designated folder.

Open the ODS EXCEL dialog to set up the file that will be created.

SAS Code:

```
ods excel file = &out
    options (autofilter = 'on'
        frozen_headers = 'yes'
        embedded_titles = 'yes'
        sheet_interval = 'bygroup'
        sheet_name = '#byval(college)'
        absolute row height = '14');
```

The SHEET_NAME option is set to name the spreadsheet tab the value of the ‘college’ variable.

Then we sort the data with PROC SORT and use PROC PRINT to output the data we want to our spreadsheet.

SAS Code:

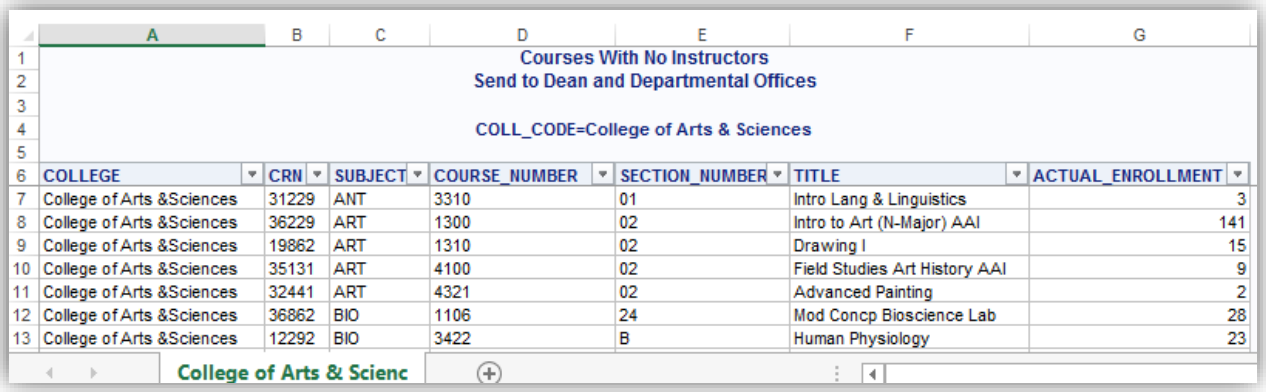
```
proc sort data = missing_instructors;
    by college
        subject
        course_number
        section_number;
run;

proc print data = missing_instructors
    noobs;
    by college;
    format college $unit.;
    var college
        crn
        subject
        course_number
        section_number
        title
        actual_enrollment;
    title1 'Courses With No Instructors';
    title2 "Send to Dean and Departmental Offices";
    where college = "&college.";
run;
```

We use the &college macro variable as a way to filter down to only the records for the current College we are writing the report for. For the first run of the %DO loop this is still Arts and Sciences (AS). We then close the ODS EXCEL process.

SAS Code:

```
ods excel close;
```

The spreadsheet that is created and saved to our file destination for Arts and Sciences looks as follows:



Courses With No Instructors
Send to Dean and Departmental Offices

COLL_CODE=College of Arts & Sciences

COLLEGE	CRN	SUBJECT	COURSE_NUMBER	SECTION_NUMBER	TITLE	ACTUAL_ENROLLMENT
College of Arts &Sciences	31229	ANT	3310	01	Intro Lang & Linguistics	3
College of Arts &Sciences	36229	ART	1300	02	Intro to Art (N-Major) AAI	141
College of Arts &Sciences	19862	ART	1310	02	Drawing I	15
College of Arts &Sciences	35131	ART	4100	02	Field Studies Art History AAI	9
College of Arts &Sciences	32441	ART	4321	02	Advanced Painting	2
College of Arts &Sciences	36862	BIO	1106	24	Mod Concp Bioscience Lab	28
College of Arts &Sciences	12292	BIO	3422	B	Human Physiology	23

College of Arts & Scienc

DELIVERING THE REPORTS

Now we are ready to deliver the reports to our customer(s). The FILENAME statement allows us to send the reports we've made by email, programmatically, using the SMTP (Simple Mail Transfer Protocol) email interface. We need to check to make sure we have access to an SMTP server to send the email from. Then, we have to add a couple of items to our SAS configuration file set before we begin the email process:

1. Note: Close SAS if it is open. You will open SAS again after you have edited and saved the configuration file. The configuration file can be edited with Notepad.
2. In your sasv9.cfg file, enter the following options:

   -emailsys SMTP

   -emailhost your.smtpemail.server.com

   -emailport 25

In the above, *emailhost* is the SMTP server that you use. Check with your email administrator to find out the correct SMTP email server name. Port 25 is by far the most common, but check with your email administrator to be sure.

Now, our report can be sent by email. We also use conditional statements to create a distribution schedule for our clients so that they receive their data at an agreed upon frequency. For the Missing Instructors Report we use days of the week to determine what customer receive the report each time it is ran.

We us the SYMPUT function to take today's date and convert it to a macro variable called &today that populates the current day of the week. The TODAY function inside of the SYMPUT statement creates a numeric date value that we apply the SAS Day of the Week (DOWNAME) format to so that we get the name of the current day (i.e. 'Monday', 'Tuesday', 'Saturday').

SAS Code:

```
/* Set a macro value for the current day of the week. */
data _null_;
    call symput('today',(put(today(), downame.)));
run;
```

This macro is used to determine whether to send the report based on the Colleges' preferences. We use a %IF conditional statement to check the current day with the delivery schedule requested.

SAS Code:

```
/*  Send list on the days requested. */
%if (&today = Thursday
    and "&college" = "AS")
or (&today = Tuesday)
    %then %do;
```

When the condition is true, the program moves forward to the %THEN %DO block. Else, it continues to the code following the %END statement.

Then, we use the FILENAME statement and a DATA step to send our email.

SAS Code:

```
/*  Email reports to IRT & appropriate contacts. */
filename mymail email from = 'Jacob_Price@baylor.edu'
    to = (&contact_email)
    cc = (&irt_email)
    subject = "Missing Instructors &college"
    content_type ='text/html'
    attach =(&out.);
```

The syntax for the FILENAME statement is FILENAME *fileref email* <'*address*'> <*email-options*>;.The *fileref* here is "mymail" (can be anything up to eight bytes). The *address* is the sender's email address. For our purposes, this is populated with the &irt_email macro variable that we created in our code above. The *email-options* that we are using are:

1. TO = the email addresses of the recipient(s). For our program, this is our customer contacts' email addresses that are contained in the &contact_email address.
2. CC = email addresses of any courtesy copy recipients. In this instance, we are copying ourselves on the correspondence with the &irt_email macro variable.
3. SUBJECT = subject of the email.
4. CONTENT_TYPE = Specifies the content type for the body of the message. The default is plain text ('text/plain'). We are using 'text/html' so that we can utilize html code in the body of the email.
5. ATTACH = any attachments to send with the email. For our email, we are attaching the excel spreadsheet that we named in the &out macro variable previously.

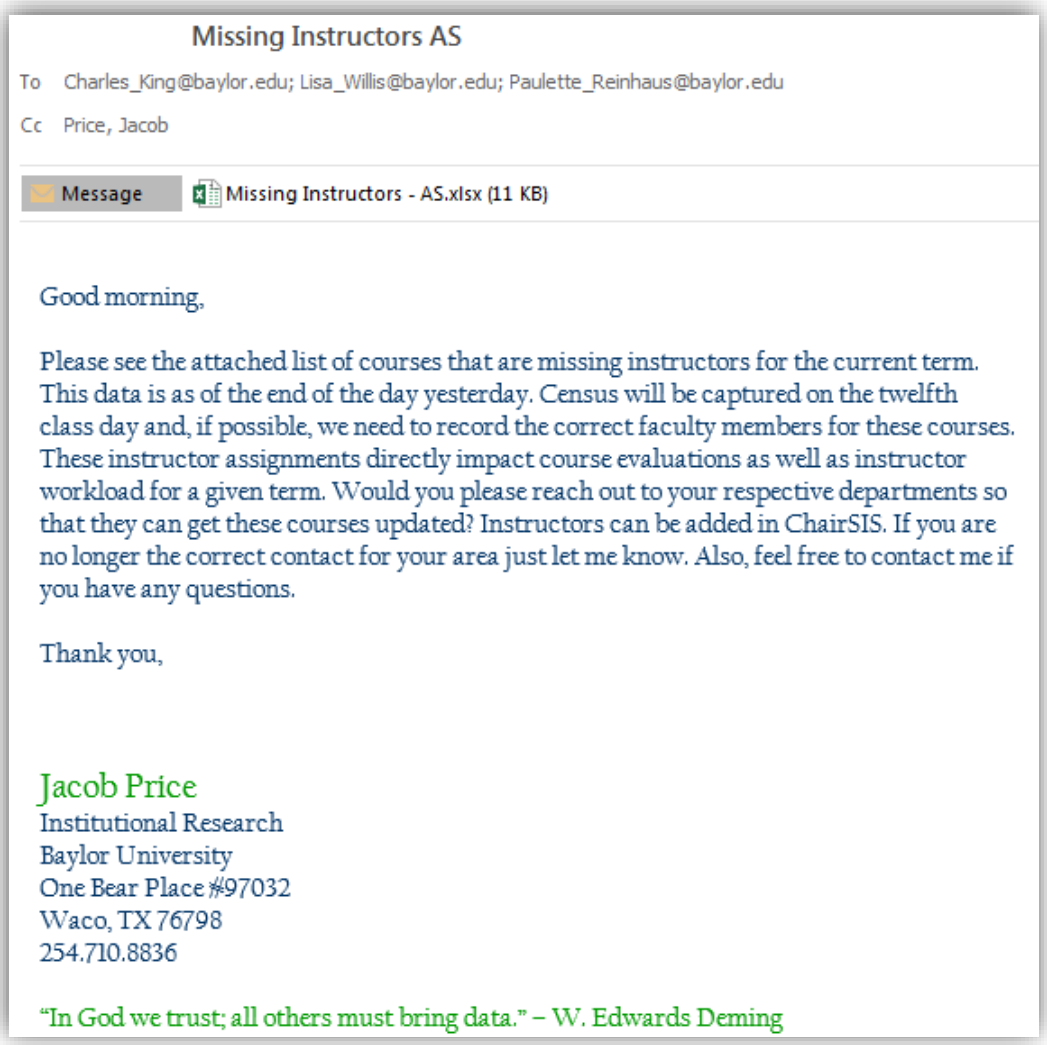
We then use a DATA step and the 'mymail' file to send the email.

SAS Code:

```
data _null_;
file mymail;
put '<font color = "003366" font face = "Californian FB"
    p style = "font-size:11pt">';
put "Good morning,";
put "<br> <br> Please see the attached list of courses that";
put "are missing instructors for the current term. This data";
put "is as of the end of the day yesterday. Census will be";
put "captured on the twelfth class day and, if possible, we";
put "need to record the correct faculty members for these";
put "courses. These instructor assignments directly impact";
put "course evaluations as well as instructor workload for a";
put "given term. Would you please reach out to your";
put "respective departments so that they can get these";
put "courses updated? Instructors can be added in ChairSIS.";
put "If you are no longer the correct contact for your area";
put "just let me know. Also, feel free to contact me if you";
put "have any questions.";
put "<br> <br> Thank you,";
put &irt_signature;
run;
```

The _NULL_ keyword in the place of the data set name will not create a SAS data set. Then the FILE statement is used to call our 'mymail' external file we created above in the FILENAME statement. A series of PUT statements create the body of our email message using HTML formatting. Finally, we will put our &irt_signature macro variable at the end to include our email signature that it has stored.

When the email is delivered, it looks like this:



Missing Instructors AS

To Charles_King@baylor.edu; Lisa_Willis@baylor.edu; Paulette_Reinhaus@baylor.edu

Cc Price, Jacob

Message Missing Instructors - AS.xlsx (11 KB)

Good morning,

Please see the attached list of courses that are missing instructors for the current term. This data is as of the end of the day yesterday. Census will be captured on the twelfth class day and, if possible, we need to record the correct faculty members for these courses. These instructor assignments directly impact course evaluations as well as instructor workload for a given term. Would you please reach out to your respective departments so that they can get these courses updated? Instructors can be added in ChairSIS. If you are no longer the correct contact for your area just let me know. Also, feel free to contact me if you have any questions.

Thank you,

Jacob Price
Institutional Research
Baylor University
One Bear Place #97032
Waco, TX 76798
254.710.8836

"In God we trust; all others must bring data." – W. Edwards Deming

FINISHING UP THE MACRO

Once the email has been sent for the current iteration of our %DO loop we begin to complete our processes with %END statements.

SAS Code:

```
                %end;

        %put _user_;

        %end;

    %end;
```

The innermost %END statement closes the %THEN %DO code block that followed the %IF statement based on the Colleges' scheduled delivery days.

The %PUT _USER_ statement puts all of our current macro variable and their values into our log. This is a good way to see what the macros' values were for each iteration of our %DO loop and can be very helpful when troubleshooting your code. In our case, our macro variables are local to our ***reports_and_emails*** macro, so the %PUT _USER_ statement needs to be located in our macro code in order for us to get their values in the log.

The second %END statement completes the iterative %DO loop that set the value for &i and was used to create the &contact_email and &college macro variables for the customized reports and emails. It will, upon completion loop back up to the top of the %DO loop and run everything inside the loop again using two as the next value for the index variable &i. This will continue until the loop completes after the STOP value. For our current program, that number is five, because that is the value of the &college_count macro variable we used. Once the loop concludes the iteration where &i = 5, the value of &i will go to six and the loop will not iterate again.

The third and final %END statement closes out the first %THEN %DO block that was conditional to our "missing_instructors" data set having at least one observation.

We then end our ***reports_and_emails*** macro with the %MEND statement.

SAS Code:

```
%mend reports_and_emails;
```

The final line of code is our macro call for the ***reports_and_emails*** that actually initializing our macro to run.

SAS Code:

```
%reports_and_emails;
```

CONCLUSION

In this program, we created a SAS data set for our contacts list. Then we wrote a macro named ***reports_and_emails*** that takes our data, sorts through it and creates a different excel spreadsheet for every customer that receives the data. Then we programmatically emailed the report to the individual customers based on the information in our contact list and their preferred delivery days.

This program takes some work to set up, but it has saved our office a ton of time since it has been put in place. In addition, the techniques used here have served great use for other processes that involve emailing reports to our customers, saving our team dozens of emails across a number of projects. They have made our office much more efficient in delivering our reports and have played a big part in helping us to become reporting rock stars in the eyes of our customers.

REFERENCES

FILENAME Statement, EMAIL (SMTP) Access Method

SAS. 2016. SAS 9.4 Statements: Reference. Fifth edition. Chapter 2: Dictionary of SAS Statements, FILENAME Statement, EMAIL (SMTP) Access Method (pp. 112-127). Available at http://support.sas.com/documentation/cdl/en/lestmtsref/69738/PDF/default/lestmtsref.pdf

SAS. 2016. Support Samples and SAS Notes. Usage Note 19767: Using the SAS System to Send SMTP Email. Available at: http://support.sas.com/kb/19/767.html

ODS EXCEL

Huff, Gina. 2016. "An 'Excel'lent Journey: Exploring the New ODS EXCEL Statement." *Proceedings of the 2016 SAS Global Forum Conference.* Available at http://support.sas.com/resources/papers/proceedings16/2780-2016.pdf

SAS. 2015. SAS 9.4 Output Delivery System: User's Guide. Chapter 6: Dictionary of ODS Language Statements, ODS EXCEL Statement (pp. 292-320). Available at http://support.sas.com/documentation/cdl/en/odsug/67921/PDF/default/odsug.pdf

SAS Macro Language

Carpenter, Art. 2004. *Carpenter's Complete Guide to the SAS Macro Language.* Second edition. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Jacob Price
Institutional Research & Testing
Baylor University
254.710.8836
jacob_price@baylor.edu