Paper 240-2012

"There's an App for That": It's Called SAS® ODS! Mobile Data Entry and Reporting via SAS ODS

Michael Drutar, SAS Institute Inc., Cary, NC

ABSTRACT

Most mobile reporting allows users only to receive reports; however, it is often necessary to INPUT data on a mobile device. This can be accomplished by leveraging Google Docs spreadsheets and SAS ODS. This paper's example explains how a cross country coach records a runner's race times (in real time) on a Google spreadsheet using his mobile device. His office computer has a SAS® job which, via the FILENAME statement, reads the updated Google spreadsheet into SAS. SAS processes this data and sends an e-mail containing an embedded HTML dashboard of Runner KPI dials, charts and/or tables. The coach receives this report almost instantaneously. Because it is an embedded e-mail, this solution works on nearly any mobile device (for example, iPhone or Android).

INTRODUCTION

Today nearly all companies are putting resources toward ensuring that they can deliver to mobile devices. This poses several problems: designing content that can be viewed on all devices (iPhone, Android), hiring new developers that can deploy the new content to their respective device, and so on. This process could take months, if not years. Because of these issues, nearly everyone is looking for a quick and easy solution to get their content onto the mobile devices.

A simple and effective solution to this problem is found in the concept of the HTML embedded e-mail. Nearly all e-mail messages sent today are structured as HTML code that e-mail clients (such as Microsoft Outlook) render as Web pages. Hence, your e-mail client, be it on your mobile device or on your desktop computer, is essentially a Web browser. This means that e-mail messages can display HTML code that is embedding within the e-mail. Recently, more and more companies have taken to communicating with their customers via an image rich, embedded e-mail rather than just static text e-mailing campaigns. An example of this occurs when grocery stores send out "weekly specials" e-mail to their customers. The e-mail typically has the company's logo image and a list of items that are currently on sale, along with accompanying images.

For the SAS programmer, this new trend of sending information via embedded HTML e-mails is great news. The SAS ODS (Output Delivery System) is a go-to tool for transforming SAS output to an HTML file. A programmer can easily take data, and after analysis create some charts and graphs from the data. Then by using the ODS system, the programmer can generate an HTML file that combines all the SAS output into one clean, easy to read view in the form of an HTML file or pdf. The next step is to take the ODS content and send it via e-mail.

Since mobile devices all have e-mail clients that will display HTML messages, a SAS programmer can generate ODS content that is ready to be consumed by clients on the go. The programmer must first send an e-mail message using the FILENAME statement. The use of the FILENAME statement allows the programmer to embed SAS procedural output directly within the HTML of the e-mail message. Then, by leveraging the style option within the PROC REPORT statement, the programmer can generate custom HTML, which therefore can include SAS graphics.

The HTML embedded e-mail is a solution for the SAS programmer who wants to simply push content to mobile devices. However, as the mobile market continues to move at an amazingly quick pace, recently the requirement for users to not only receive information but also input information onto their mobile devices has arisen. More so, customers are expecting that the information they input should be reflected almost immediately on the application they are using. Examples of this are Facebook status updates, Twitter, and so on. There is a strong relationship between the most used mobile applications and those that enable the user to input data as well as receive data.

In the case of SAS, mobile users might desire to update a data point on their mobile device and have that update automatically update a SAS data set. For the SAS developer this is quite a hurdle. The developer has to somehow have a user interface that users can access from their mobile devices, have inputs (such as text boxes) that the user can both place new data into, or edit existing data values. The user must then be able to save the changes, and send them to the SAS system to which they are communicating. Even if the SAS developer was to create such an application that users could do the input tasks described above, he/she would also have to develop a different version of the application for use by different mobile devices (iPhone, Android).

Fortunately, Google Docs spreadsheets have already done the heavy lifting when it comes to solving the issue of allowing customers to make input data, or make edits to data on mobile devices. Even better, if the Google spreadsheet is saved with the appropriate permissions and format, one can import a Google spreadsheet directly from the Web into SAS.

GOOGLE SPREADSHEETS

Google offers many services online. These services include an e-mail application, a calendar application, and even a documents application. Within the documents application a user can create presentations, word documents, and spreadsheets. The spreadsheets application allows users to create new data files, upload data files (from a variety of formats) from their local machine, and publish their data files for sharing.

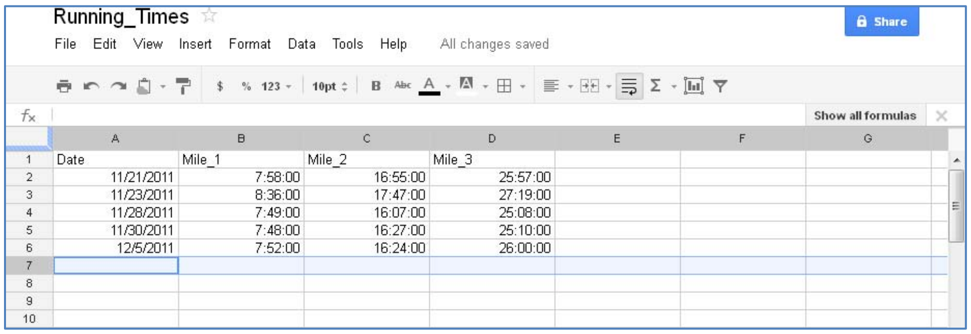
One big advantage to using Google's services is that they are "mobile compatible." That is, Google has gone to great efforts to ensure that their applications can be used by almost any mobile device. For example, a Google spreadsheet renders very nicely on both an Android or Apple mobile device. Not only can users view and edit a spreadsheet using the built in Internet browser on their device, but Google has also published a well-built Google Documents App in both the Android App Store and Apple App store. The ability for users to access, change, and publish spreadsheet files on-the-go from their phones and tables makes the Google spreadsheet a perfect fit for any person or organization that needs an online database.

GOOGLE SPREADSHEETS USAGE

This paper will demonstrate the usage of Google spreadsheets via two examples. Both examples capture and report running times. The first example is for just one individual who is keeping a daily running log for a 5K (5 Kilometers) distance run. The second example will demonstrate how a cross country coach can log his team's individual runner's times for analysis within SAS.

For our first example, the individual user logs in to Google Docs at http://docs.google.com and clicks the **Create** button on the top left of the screen. The user then selects **spreadsheet** to start a new spreadsheet file within the Google Docs application. The user is then presented with a blank spreadsheet and can begin data entry.

The user can now begin entering the data associated with his or her running. In this example, the data consists of 4 columns: Date, Mile_1, Mile_2, and Mile_3. It is assumed that the user already has some previous data from running in the past and enters them. The user then renames the spreadsheet "Running_Times."



	A	B	C	D	E	F	G
1	Date	Mile_1	Mile_2	Mile_3			
2	11/21/2011	7:58:00	16:55:00	25:57:00			
3	11/23/2011	8:36:00	17:47:00	27:19:00			
4	11/28/2011	7:49:00	16:07:00	25:08:00			
5	11/30/2011	7:48:00	16:27:00	25:10:00			
6	12/5/2011	7:52:00	16:24:00	26:00:00			
7							
8							
9							
10							

Figure 1. Google Spreadsheet Main Page

The spreadsheet is now ready for the user to update with running times directly from the race track on his or her mobile device. The user can simply take their mobile device and by either the device's Internet browser or using the Google Docs App enter new data immediately after finishing a race. The screenshot below demonstrates mileage splits being added for December 7 via the Google Docs App:

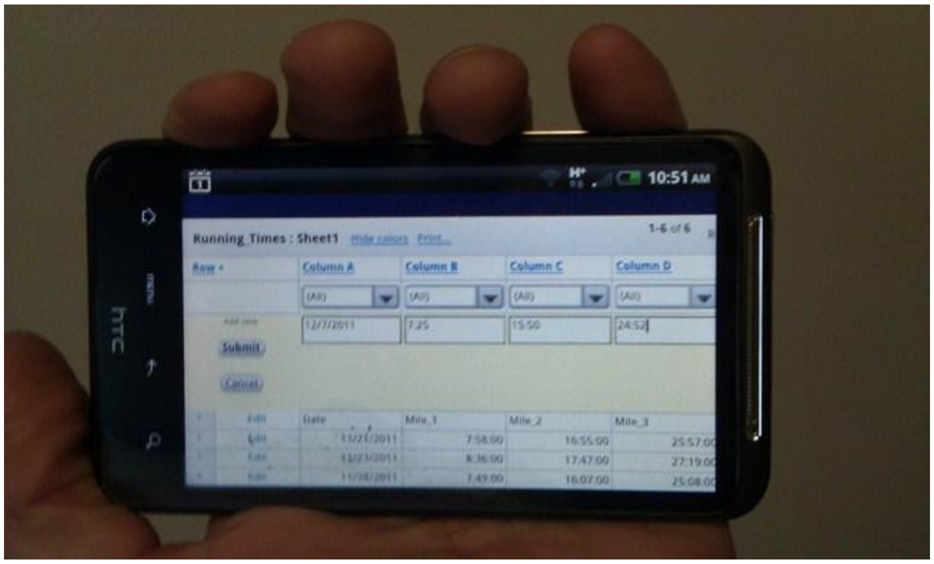


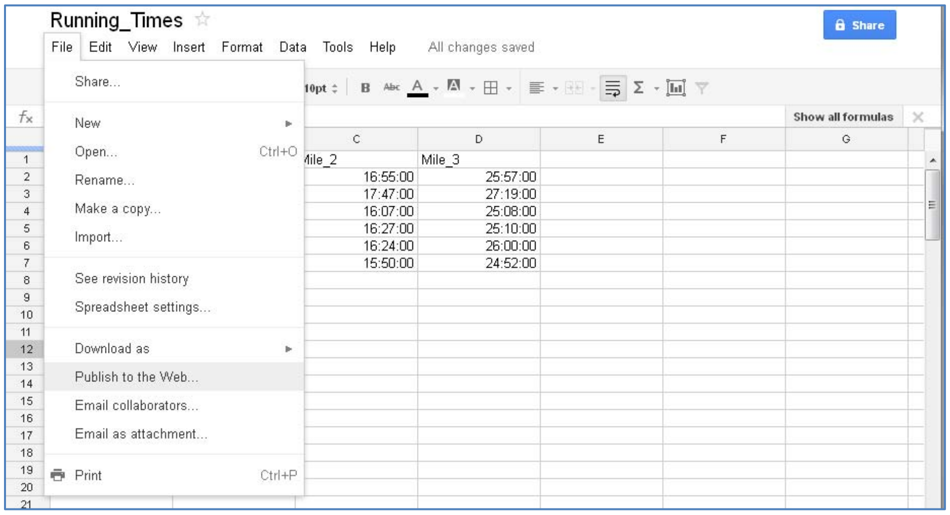
Figure 2. Entering Data Into a Google Spreadsheet From a Mobile Device

As you can see, it is very easy to update a Google Spreadsheet from a mobile device, and there is great value in having this. The runner essentially now has an online database provided by Google. The next step is to do analysis with SAS using the Google spreadsheet as a data source.

SAS AND GOOGLE SPREADSHEETS

It is already known that SAS is the best tool in the business when it comes to reading data files, regardless of the format. One of the first concepts taught to new SAS programmers is the ability to import data files using PROC IMPORT via the FILENAME statement. The FILENAME statement can point to a variety of file types, one of them being a Google spreadsheet URL.

However, before the SAS FILENAME statement can successfully connect to a Google spreadsheet, a few settings need to be changed on the spreadsheet. First a user must publish the spreadsheet. This is achieved by opening the spreadsheet file using Google Docs and clicking the **File** option from the top menu. Then the user chooses **Publish to the Web**.



	C	D	E	F	G
1	Mile_2	Mile_3			
2	16:55:00	25:57:00			
3	17:47:00	27:19:00			
4	16:07:00	25:08:00			
5	16:27:00	25:10:00			
6	16:24:00	26:00:00			
7	15:50:00	24:52:00			

Figure 3. Publishing Your Google Spreadsheet to the Web

On the next dialog box, the user selects **Start Publishing**. Underneath **Get a link to the published data**, Google now displays the direct link to where the Google spreadsheet file is published. However, there is also an option to choose which file type the link represents. Since it is quite easy for SAS to import a CSV file, the user chooses this option:

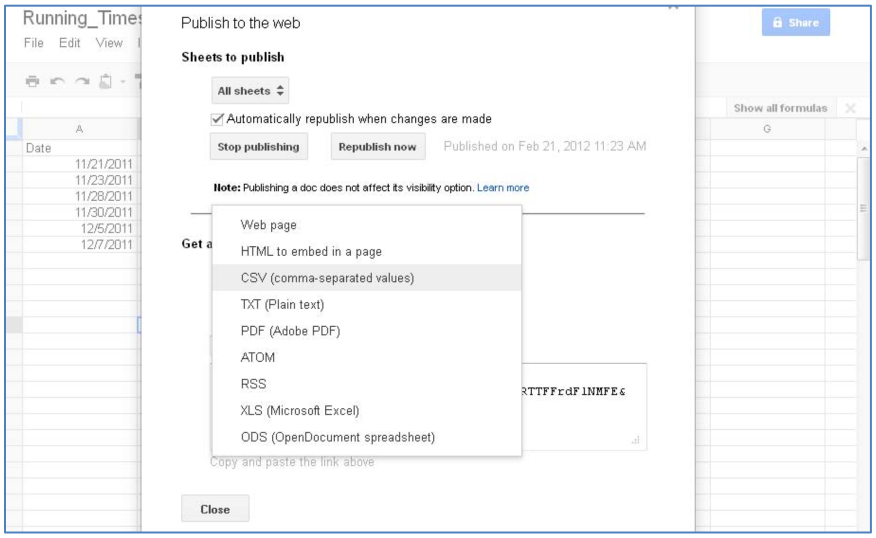
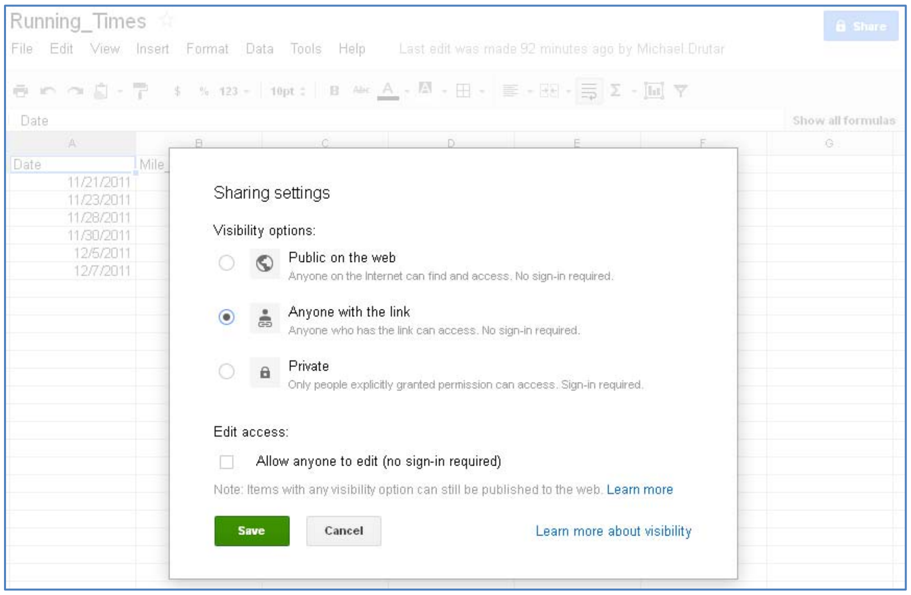


Figure 4. Choosing CSV as the Published File Type

The user can now view the full link to the newly published file that contains the running data. This will be the link to which the user can leverage the FILENAME statement to import the Google spreadsheet and create a SAS data set. Therefore, the user should save this link.

However, before SAS can access the published file, the Google spreadsheet needs to be shared. This is achieved by clicking the **Share** button at the top right of the Google spreadsheet page. Once clicked, a dialog box appears explaining the current level of access for the spreadsheet. The user now must click the **Change** link to the right of the text **Private - Only the people listed below can access**. Now a Share Settings dialog box appears. In order to allow SAS to read the newly published CSV file on the Web, the user must choose **Anyone with the link** as shown in the screenshot below:


Sharing settings

Visibility options:

( ) Public on the web
Anyone on the Internet can find and access. No sign-in required.

(●) Anyone with the link
Anyone who has the link can access. No sign-in required.

( ) Private
Only people explicitly granted permission can access. Sign-in required.

Edit access:

☐ Allow anyone to edit (no sign-in required)

Note: Items with any visibility option can still be published to the web. Learn more

Running_Times
File Edit View Insert Format Data Tools Help
Last edit was made 92 minutes ago by Michael Drutar
Share
Show all formulas
Date
Mile
11/21/2011
11/23/2011
11/28/2011
11/30/2011
12/5/2011
12/7/2011
Save
Cancel
Learn more about visibility

Figure 5. Setting Appropriate Share Settings

IMPORTING GOOGLE SPREADSHEETS INTO SAS

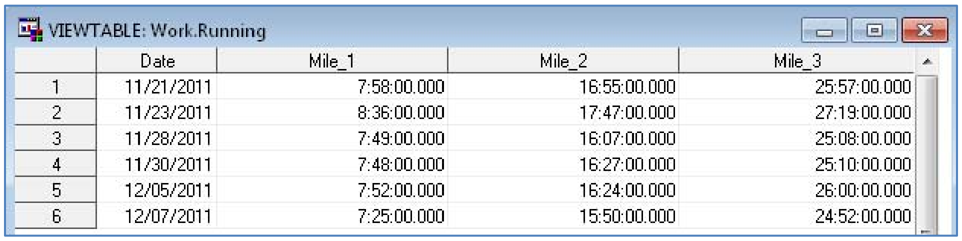
Now that the appropriate sharing and publish settings on the Google spreadsheet have been set, the SAS FILENAME statement can read the file. This is achieved by submitting a FILENAME statement using the URL access method. For the external file, the user specifies the link to the published as CSV Google spreadsheet. For this example that tracks running times, we will call the FILENAME "running:"

```
FILENAME running url
'https://docs.google.com/spreadsheet/pub?key=0Aq78lU4RvCx3dE1ZDERDGlItVERLWkRTTFFrdFlN
MFE&output=csv' debug;
```

Now that the FILENAME statement has been established, the programmer can use PROC IMPORT to import the file. Here is where the concept of using Google's "Publish As CSV" link is key. Since the link is pointing to nothing more than a CSV file on the Web, all the great features that PROC IMPORT offers while importing a CSV file apply to the Google spreadsheet. In this example, the programmer takes advantage of the GETNAMES=YES and DATAROW=2 options to create the data set "running" from the file "running."

```
   proc import datafile=running out=running
                DBMS=CSV REPLACE;
         GETNAMES=Yes;
         DATAROW=2;
   run;
```

The programmer now can submit the FILENAME statement and PROC IMPORT code. SAS imports the file and creates the data set with the data from the Google spreadsheet.


VIEWTABLE: Work.Running

	Date	Mile_1	Mile_2	Mile_3
1	11/21/2011	7:58:00.000	16:55:00.000	25:57:00.000
2	11/23/2011	8:36:00.000	17:47:00.000	27:19:00.000
3	11/28/2011	7:49:00.000	16:07:00.000	25:08:00.000
4	11/30/2011	7:48:00.000	16:27:00.000	25:10:00.000
5	12/05/2011	7:52:00.000	16:24:00.000	26:00:00.000
6	12/07/2011	7:25:00.000	15:50:00.000	24:52:00.000

Figure 6. Google Spreadsheet Imported as a SAS Data Set

The programmer has now successfully imported a Google spreadsheet directly into SAS and can begin analysis.

SENDING SAS OUTPUT TO MOBILE DEVICES

As described in the introduction, a solution to the problem of sending SAS output to mobile devices is found in sending embedded HTML e-mails. Nearly all mobile devices have an e-mail client, which can read an HTML-based e-mail message. Hence, a SAS programmer can generate output from SAS and send it to any mobile device.

E-mailing procedural output from SAS is nothing new. For review, one can read the section titled "Sending Procedure Output as E-Mail" at this Web page on support.sas.com:
http://support.sas.com/documentation/cdl/en/hosto390/61886/HTML/default/viewer.htm#a001412669.htm.

The example from support.sas.com shows how one can use the PROC PRINT procedure and embed its output directly into the HTML body of an e-mail. This embedded e-mail renders very nicely on mobile devices. However, while PROC PRINT is an excellent procedure to print data, it can sometimes be inhibiting. If the programmer uses PROC REPORT in this e-mail example, much more customization is available. Below is an enhanced adaptation from the example from support.sas.com using PROC REPORT to e-mail an ODS listing of the running data set that was imported from the Google spreadsheet. Also, the date column has been converted to a character column.

```
FILENAME outbox email
  to = (‘anybody@company.com’)
  subject = "Running Times Dataset"
  content_type = "text/HTML";
ods listing close;
ods HTML body = outbox style = styles.minimal;
proc report data = running;
column Date Mile_1 Mile_2 Mile_3;
run;
quit;
ods HTML close;
ODS LISTING;
```

Now that the programmer can successfully e-mail a list of the running data set to a recipient, the next obvious question is how to send images created by SAS procedures such as PROC GCHART. One approach is to create one or more images with text and use PROC GREPLAY to create one final image and attach this to the e-mail message. However, this approach poses two potential issues. First, if an image is e-mailed, it doesn’t always render on mobile devices. In other words, the image can be attached to an e-mail; but that doesn’t achieve the goal of EMBEDDING the image directly into the body of the e-mail message. Secondly, if an image is generated with several plots and charts, it will have a defined height and width, which renders differently on each mobile device, as different mobile devices have different screen sizes.

The solution to both of these issues is to generate custom HTML within the PROC REPORT statement. By using HTML tables with heights and widths defined as percentages of the display space, the embedded e-mail will dynamically resize to whatever size screen it appears on. More so, the custom generated HTML can display images where the image source can exist anywhere. Hence, the SAS programmer can generate GCHART output as an external image file (such as a PNG) and place these images on a Web server. The programmer can then embed placeholders for those images within the e-mail.

The ability to generate custom HTML within the PROC REPORT statement lies within the overall style statement. One option within the style statement is the PREHTML option. The PREHTML option allows the programmer to place custom HTML into the style statement. When PROC REPORT displays its results, the custom HTML that the programmer placed in the PREHTML option is displayed before the table itself if printed. For the current example of the daily running report, the SAS programmer first reads in the Google spreadsheet. The programmer then generates a graph using PROC GCHART, and places the resulting image that PROC GCHART creates into his/her Web server’s directory. The programmer then runs a FILENAME statement with a PROC REPORT statement in it to send the resulting PROC REPORT output in an e-mail. However, the PROC REPORT statement uses the PREHTML option and the programmer has written custom HTML to display the image generated by PROC GCHART (which currently sits in the Web server directory). The custom HTML places the image and its title in a table that has a 100% width.

```
%let name=bar1;
FILENAME odsout '<- path to webserver ->';
goptions xpixels=480 ypixels=300 device=png noborder cback=white;
ods listing close;
ods HTML path=odsout body="&name..htm" style=minimal;
title;
footnote;

proc gchart data=running1; vbar Date / sumvar=Mile_3 clipref frame type=sum
  coutline=black raxis=axis1 maxis=axis2
  des="" name="&name";
run; quit;

FILENAME outbox email
  to = ('anybody@company.com')
  subject = "Running Times Bar Chart And Table"
  content_type = "text/HTML";
ods listing close;
ods HTML body = outbox style = styles.minimal;

proc report data = running STYLE = [PREHTML = %UNQUOTE(%STR(%')
  <table width="100%" border="0" cellspacing="0" cellpadding="0"><tr>
  <td align="center"><strong>3 Mile Race Times</strong></td></tr>
  <tr><td align="center">
  <img src="http://webserveraddress/bar1.png" width="480" height="300"
align="middle" />
  </td></tr></table>
  %STR(%'))];
column Date Mile_1 Mile_2 Mile_3;
run;
quit;
ods HTML close;
ODS LISTING;
```

The data analysis delivered by the above is quite simplistic. However, with a few more advanced PROC GCHART outputs, along with a better formatted table displayed by PROC REPORT, the information delivered by the e-mail can be quite useful. More so, since the programmer is able to place any type of custom HTML code within the style statement of the PROC REPORT procedure, the overall look of the HTML delivered by the e-mail can get quite fancy. This will be shown in the next example.

AUTOMATION

The example above shows end-to-end how a programmer can import a data set from Google spreadsheets, generate analysis, and then distribute analysis to recipients. The next question becomes how to automate the process so that when the runner inputs a new time using their mobile device they will automatically get a resulting HTML embedded e-mail back. This is achieved by SAS job scheduling.

The programmer could have a SAS job that reads in the Google spreadsheet and does a PROC COMPARE with a local copy of the data set. If the results of the PROC COMPARE show that there is a new line in the Google spreadsheet that doesn't exist on the local copy of the data set, the SAS program will execute the code from the example above. This results in the embedded e-mail being sent to the recipient(s). After which the program will replace the local copy of the data set with the version of the data set that is currently in the Google spreadsheet. The program will then continue to run every five minutes. If it finds no difference between the Google spreadsheet data set and the local data set, it ends. Therefore, the sending of embedded HTML e-mails with current data is automated.

CROSS COUNTRY COACH EXAMPLE

The second example in this paper involves a cross country coach recording team runners' race times (in real time) on a Google spreadsheet using his mobile device. As with the example above, the Google spreadsheet onto which the coach is recording is being shard and is being published to a Web CSV file. The coach has recorded the runners' times for the past 3 races and is entering the times for the 4th race. For each race, he records the following: the race date, the race name, the runner's name, the runners 1st, 2nd, overall 5K race time, and the temperature.

The coach completes entering the runners' race times. The scheduled SAS job on the computer at his/her office sees that there are new records on the Google spreadsheet that don't exist in the local copy. The job then starts analyzing the data. Three main types of analysis plots are generated.

First, a KPI dial image is created for the team's average first mile split as compared to the previous three races. This image is named "mile_1_dial.png" and is placed in the Web server's directory. The same KPI dial image is created for the team's second mile and overall race time splits. These images are titled "mile_2_dial.png" and "race_dial.png" respectively.

Second, a bar-line chart of the team's average overall race time versus temperature for the last 4 races is generated. In this chart, the bars are a representation of the average finish time for each race. The bars are sorted by the date the race was won. The line plot piece of the bar chart represents the temperature for the day the race was run. With this bar-line chart, the coach can look for relationships between how fast the team is running and the temperature. The output file for the bar-line chart is saved into the Web server's directory and is titled "time_vs_temp.png."

Finally, a line plot is created to show how the team averages in splits. The line plot shows how the team's average mile time change as the race wears on. The image output from this plot is also placed in the Web server's directory and named "splits.png."

Finally, for the table that PROC REPORT actually prints, the program finds the fastest and slowest runners for the current race, the season average, and the records for the season. The output from SAS for this data set is placed in the WORK library and is called "work.final_analysis".

The programmer next generates custom HTML that will display a cross country dashboard. Next the custom HTML is placed in the PREHTML section of PROC REPORT and the data set work.final_analysis is used as the source. The resulting SAS code for sending the PROC REPORT output along with the custom HTML is shown below:

```
FILENAME anyname EMAIL
  FROM = 'anyone@company.com'
  SENDER = 'anyone@company.com'
  REPLYTO = 'anyone@company.com'
  TO = ('anyone@company.com')
  SUBJECT = "Cross Country Dashboard"
  CONTENT_TYPE = "text/HTML";
ODS NORESULTS;

ODS LISTING CLOSE;
ODS HTML BODY = anyname STYLE = styles.minimal;
OPTIONS NOCENTER;
TITLE;

proc report DATA =final_analysis
  STYLE = [bordercolor=white borderwidth=0 PREHTML = %UNQUOTE(%STR(%')
<body>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td align="center"><h2>Carolina Forest High School Cross Country
Dashboard</h2></td>
  </tr>
  <tr>
    <td align="center"><h4>High School C Invitational - December 14<sup>th</sup>,
2011</h4>
    <p>&nbsp;</p></td>
  </tr>
  <tr>
    <td align="center"><img src="http://webserveraddress/mile_1_dial.png" alt=""
width="182" height="172" /><img src="http://webserveraddress/mile_2_dial.png"
alt="" width="182" height="172" /><img src="http://webserveraddress/race_dial.png"
alt="" width="182" height="172" /></td>
  </tr>
  <tr>
    <td align="center"><img src="http://webserveraddress/time_vs_temp.png" alt=""
width="480" height="300" /></td>
  </tr>
  <tr>
    <td align="center"><img src="http://webserveraddress/splits.png" alt=""
width="480" height="300" /></td>
  </tr>
</table>
</body>
<div align="center">
%STR(%'))
];

RUN;
quit;
ODS HTML CLOSE;
ODS LISTING;
```

This code sends an HTML embedded e-mail, which is shown below:

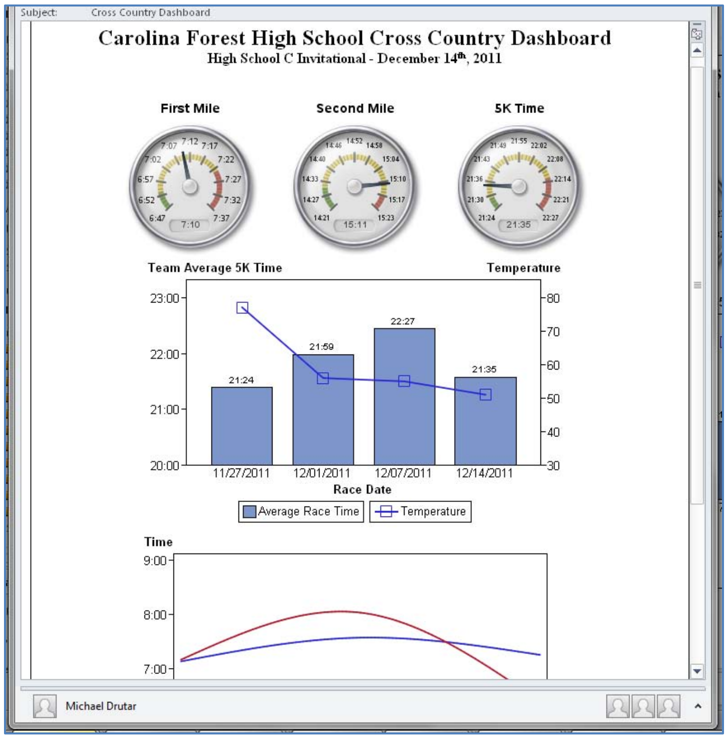


Figure 7. HTML Embedded E-mail of the Cross Country Dashboard

This e-mail essentially renders the same on a desktop computer, mobile phone, or tablet.

CONCLUSION

The need to be able to both input and receive data from mobile devices is becoming more commonplace. SAS programmers must use their current skills to the fullest extent to meet these needs. By leveraging Google Docs spreadsheets and using your ODS programming skills, you can meet these new 'mobile' needs.

REFERENCES

News, Hints, Tips, and Information for SAS® Users. (2008). Retrieved Oct 21, 2011, from http://www.sascommunity.org/mwiki/images/3/36/VIEWS_News_Issue42.pdf

CONTACT INFORMATION

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

Michael Drutar
SAS Campus Drive
SAS Institute Inc.
E-mail: Michael.Drutar@sas.com