SESUG Paper 94–2021

Badge in Batch with Honeybadger: Generating Conference Badges with Quick Response (QR) Codes Containing Virtual Contact Cards (vCards) for Automatic Smart Phone Contact List Upload

Troy Martin Hughes

ABSTRACT

Quick Response (QR) codes are widely used to encode information such as uniform record locators (URLs) for websites, flight passenger data on airline tickets, attendee data on concert tickets, or product information on product packaging. The proliferation of QR codes is due in part to the broad dissemination of smart phones and the accessibility of free smart phone applications that scan QR codes. With the ease of QR code scanning has come another common QR code usage—the identification of conference attendees. Conference badges, emblazoned with attendee-specific QR codes, can communicate attendee contact information to other conference goers, including organizers, vendors, potential customers or employers, and others. Conference badges that contain QR codes make it easy for attendees to link up with each other because snapping a photo of a badge can immediately capture contact information (that could not otherwise be printed on the badge itself), thus eliminating the need to exchange outmoded business cards. To that end, this text introduces flexible Base SAS® software that dynamically creates attendee QR codes from a data set containing contact and other information. This data-driven approach could be used to create attendee badges by conference organizers rather than costly third-party vendors. When a badge QR code is scanned at a conference, the attendee's personal information—which can include name, job title, company, phone number, email address, city, state, website, and brief biographical statement—is ported into a variant call format (VCF) file (or vCard) that can be uploaded automatically into a smart phone's contact list. Attendees are able to select what personal information is contained within their QR code and conference organizers are able to customize and configure badge format and content through data-driven software design—an external cascading style sheets (CSS) file that dynamically alters badges without the necessity to modify the underlying code. This end-to-end system offers conference organizers potential cost savings of thousands of dollars—money that can be diverted from costly third-party badge vendors to top-shelf open bars and other necessities.

INTRODUCTION

Quick response (QR) codes, launched in 1994, are barcodes that contain information encoded in boxy black-and-white patterns. Some QR codes are unique to the user. For example, if you've purchased an airline ticket, concert ticket, or sporting event ticket in the past decade, the ticket probably displayed a QR code (unique to you) that was scanned as you entered the event. The QR code in part demonstrates that the ticket is valid and unique, thus preventing would-be thieves from entering the same venue with a photocopy of your ticket. In other cases, a QR code will not be unique to the *user* but rather to the *product*, such as a code used on product packaging that is printed on millions of identical boxes of the same product. For example, in a recent trip to Best Buy (aka, the Amazon showroom), I scanned QR codes on a number of printers, which linked me to online product documentation and reviews.

In addition to other uses, QR codes have become ubiquitous at trade, technical, and other professional conferences. With individualized QR codes emblazoned on attendee badges, conference organizers can ensure that attendees belong in designated events. More advanced QR codes allow conference goers to snap photos of attendee badges or scan them with QR-code readers, and immediately upload the associated contact information into a cellular phone contact list. The use of QR codes on conference badges can tremendously increase the ability of conference goers to interact with each other and to exchange and receive contact information—all without the need to distribute, collect, and scan business cards. And privacy concerns can be mitigated by enabling conference attendees to "opt in" or "opt out" of specific personal metadata (e.g., phone, email) that are included within their personalized QR codes.

This text introduces a data-driven method that generates variant call format (VCF) files (or vCards) that can be uploaded automatically to smart phone contact lists. Contact information is collected when attendees register online for a conference, at which point the user can optionally specify the personal metadata that the badge should contain. User metadata would be subsequently aggregated within some back-end database (not demonstrated in this text), after which only user metadata for which a user has "opted in" would be retained, and finally output to a comma-separated values (CSV) file. This is where the SAS® Honeybadger commences work—converting each user record in this CSV file into a QR code that is placed on individual badges, saved to an HTML file, and printed on 8 ½ by 11 sheets of paper—six badges to a page. This functionality empowers conference organizers to print badges themselves rather than unnecessarily paying a third-party vendor. When conference attendees want to exchange contact information, they only have to focus their smart phone camera on a badge (with QR code reading now default within Android and other phone operating systems) to upload the vCard data into their phones.

Special thanks to Rob Allison for his blog post that wholly inspired this endeavor![i] Rob's code (dated September 12, 2016) formed the basis of this exploration into the QR world.[ii] Honeybadger further relies on the Google Infographics QR code generator that is functional as of 2021, but which has been deprecated.[iii]

ATTENDEE PERSONAL DATA

Personal data for conference attendees is commonly collected during the online conference registration process. Although some of this information is confidential (e.g., credit card number), other information is considered public (e.g., first and last name, company or organization) and is commonly printed on conference badges. And in the gray midlands lie personal information (e.g., phone number, email, city of residence, social media username) that some attendees won't mind releasing publicly whereas others will want these data kept confidential. In other words, some attendees may prefer to be recognized by name only whereas others may prefer to walk around with a business card effectively emblazoned on their chest. Thus, to facilitate privacy at a personalized level, any database system collecting user metadata should ensure that conference attendees are consenting to the specific details of their metadata that will be displayed on their badges—and any user metadata deemed "not releasable" should be omitted from the CSV file from which Honeybadger will generate QR codes and conference badges.

Note that conference organizers are responsible for converting user metadata into the SAS data set format depicted in Table 1. This text does demonstrate the conversion of a CSV file—arguably one of the most canonical and interoperable data structures—into this format, enabling metadata to be ported from Excel.

Information	Variable Name	Format
Prefix	prefix	$5
First Name	firstName	$30
Middle Name	middleName	$30
Last Name	lastName	$30
Suffix	suffix	$5
City	city	$30
State	state	$2
Cell Phone	cellPhone	$12
Work Phone	workPhone	$12
Email Address	email	$40
Website URL	URL	$100
Biography	bio	$100
Company Name	company	$30
Professional Title	jobTitle	$40

Table 1. Attendee SAS Data Set Format (Ported from Unspecified Conference Registration Table)

Note that within the Biography field, carriage returns are changed to “/n” so that line breaks and multiple lines are supported within the vCard NOTE field. Blank lines are also supported, so long as they occur between text blocks (i.e., paragraphs), and are represented by two successive “/n” escape codes.

SAMPLE ATTENDEES DATA SET

The following DATA step creates a sample data set (HONEYBGR.Attendees) that includes some loveable characters from NBC’s hit comedy, *The Office*:

```
%let loc=D:\SAS\honeybadger\;  * USER MUST CHANGE THIS LOCATION!!! *;
libname honeybgr "&loc";

data honeybgr.attendees;
   infile datalines dsd;
   length prefix $5 firstName $30 middleName $30 lastName $30
      suffix $5 city $30 state $2 cellPhone $10 workPhone $10
      email $40 URL $100 bio $100 company $30 jobTitle $40;
   input prefix $ firstName $ middleName $ lastName $
      suffix $ city $ state $ cellPhone $ workPhone $
      email $ URL $ bio $ company $ jobTitle $;
   label prefix='Prefix' firstName='First Name' middleName='Middle Name'
      lastName='Last Name' suffix='Suffix' city='City' state='State'
      cellPhone='Cell Phone' workPhone='Work Phone'
      email='Email' URL='URL' bio='Biography' company='Company' jobTitle='Job
Title';
datalines;
Mr.,Michael,Gary,Scott,,Scranton,PA,,570-344-
1212,mscott@dundermiff.com,https://theoffice.fandom.com/wiki/Michael_Scott,Michael
Scott is the Regional Manager of Dunder Mifflin.,Dunder Mifflin,Regional Manager
Mr.,Jim,,Halpert,,Scranton,PA,,570-344-
1214,jhalpert@dundermiff.com,https://theoffice.fandom.com/wiki/Jim_Halpert,Jim
Halpert has been a paper salesman for far too long.,Dunder Mifflin,Salesman
Mr.,Dwight,K,Schrute,,Scranton,PA,,570-344-
1269,dschrute@dundermiff.com,https://theoffice.fandom.com/wiki/Dwight_Schrute,"Dwig
ht is a fan of Battlestar Galactica, bears, and beets.",Dunder Mifflin,Assistant to
the Regional Manager
Ms.,Pam,,Halpert,,Scranton,PA,,570-344-
1279,phalpert@dundermiff.com,https://theoffice.fandom.com/wiki/Pam_Beesly,"Pam
Halpert has held various roles, including receptionist, salesman, and office
manager.",Dunder Mifflin,Office Manager
Ms.,Angela,,Martin,,Scranton,PA,,570-344-
1206,amartin@dundermiff.com,https://theoffice.fandom.com/wiki/Angela_Martin,Angela
Martin is the head of the Accounting Department and loves her kitties.,Dunder
Mifflin,Accountant
Ms.,Meredith,,Palmer,,Scranton,PA,,570-344-
1208,mpalmer@dundermiff.com,https://theoffice.fandom.com/wiki/Meredith_Palmer,Mered
ith Palmer sleeps with suppliers for steak coupons and discounted products.,Dunder
Mifflin,Supplier Relations
;
```

Note that the &LOC global macro variable must be changed to reflect the user’s SAS system. In reality, conference data would have been maintained in a database, table, SAS data set, Excel spreadsheet, or other application from which data could be ported to SAS; thus, a CSV file is used here to represent the output of this data transfer. This text file represents the finalized, culled data set in which personal information (that attendees wish to remain private, like cell phone numbers) has been removed.

CREATING A VIRTUAL CONTACT CARD (VCARD)

The vCard format specification is defined as an ANSI standard to ensure compatibility across devices.[iv] All possible vCard fields are not incorporated into the QR codes that are utilized; thus, only those fields listed in Table 1 are utilized by the Honeybadger. The metadata content contained within the QR codes represent

the typical contact information (e.g., name, phone number, email, job title, company) that would be printed on a business card, as well as additional (optional) information (e.g., short biography) that might be exchanged in conversation. And, for those conference goers who prioritize privacy, their QR codes can be generated to include only first and last name, with no additional information displayed.

When a conference attendee scans a QR code on a badge, the smart phone recognizes the vCard format, and prompts the user to create (or update) a contact (based on the phone's operating system, make, and model). The contact information is automatically saved to the phone and, if the badge wearer has provided optional biographical information, that information is uploaded as well. vCard files can also contain one URL, so attendees can choose to highlight their LinkedIn, Facebook, Twitter, or other social media handle, or to link to a blog or other URL. In this way, although some conference goers may still choose to distribute business cards at conferences, they can instead rely on their badge to convey this information without the wasted paper—a far more efficient and "greener" solution.

The following DATA step iteratively creates a vCard file for each person in the Attendees data set:

```
data _null_;
   set honeybgr.attendees end=eof;
   length fname $200;
   array line $200 line1-line13;
   fname="&loc" || 'vCard' || strip(put(_n_,8.)) || '.vcf';
   file f filevar=fname;
   line1='BEGIN:VCARD';
   line2='VERSION:4.0';
   line3=cats('N:',lastName,';',firstName,';;',prefix,';');
   line4=catx(' ','FN:',firstName,lastName);
   line5=cats('ORG:',company);
   line6=cats('TITLE:',jobTitle);
   line7=cats('TEL;TYPE=work:tel:+',workPhone);
   line8=cats('TEL;TYPE=cell:tel:+',cellPhone);
   line9=cats('ADR;TYPE=work:',city,';',state);
   line10=cats('EMAIL;TYPE=work:',email);
   line11=cats('URL:',url);
   line12=cats('NOTE:',bio);
   line13='END:VCARD';
   do over line;
      put line;
      end;
   if eof then call symputx('numpeeps',strip(put(_N_,8.)),'g');
run;
```

Thus, given the sample data set, the DATA step creates six vCard files (vCard1.vcf through vCard6.vcf). For example, the first observation (Michael Scott) is saved in the text file vCard1.vcf:

```
BEGIN:VCARD
VERSION:4.0
N:Scott;Michael;;Mr.;
FN: Michael Scott
ORG:Dunder Mifflin
TITLE:Regional Manager
TEL;TYPE=work:tel:+570-344-1212
TEL;TYPE=cell:tel:+
ADR;TYPE=work:Scranton;PA
EMAIL;TYPE=work:mscott@dundermiff.com
URL:https://theoffice.fandom.com/wiki/Michael_Scott
NOTE:Michael Scott is the Regional Manager of Dunder Mifflin.
END:VCARD
```

CREATING A QR CODE FROM A VCARD

The QR macro (thank you again Rob Allison!) contacts the Google API and iteratively converts each text file—in this case, the vCard text files that were dynamically created—into a QR code PNG file:

```
%macro qr(in,out);
data;
   infile &in recfm=f lrecl=4096 length=l;
   length url_encoded $8192;
   keep url_encoded;
   input @1 all $varying4096. l;
   url_encoded='chs=500x500&cht=qr&chl='||urlencode(strip(all))||'&chld=l';
   call symputx('qrtextl',length(url_encoded));
   output;
   stop;
run;

filename qrtext temp recfm=f lrecl=&qrtextl;

data _null_; set;
   file qrtext;
   put url_encoded;
run;

proc delete;
run;

proc http in=qrtext out=&out method='post'
   url='https://chart.googleapis.com/chart?'
   ct='application/x-www-form-urlencoded';
run;
filename qrtext clear;
%mend;
```

The CALLQR macro iterates through the list of vCard files and calls the QR macro to build a QR PNG file from each vCard file:

```
%macro callqr();
%local i;
%do i=1 %to &numpeeps;
   filename vcardin "&loc.vCard&i..vcf";
   filename qrout "&loc.qrcode&i..png";
   %qr(vcardin,qrout);
   filename vcardin clear;
   filename qrout clear;
   %end;
%mend;

%callqr;
```

This invocation of CALLQR creates QRcode1.png through QRcode6.png. Figure 1 demonstrates QRcode1.png, which represents the vCard data for Michael Scott—the first record in the CSV file.

Figure 1. vCard Data for Michael Scott (QRcode1.png)

SCANNING A QR CODE AND ADDING A CONTACT

When a conference goer (such as a conference organizer, security guard, vendor, or other attendee) scans the QR code in Figure 1 using a smart phone, the user is immediately prompted with choices. For example, Figure 2 demonstrates this scan on a Samsung smart phone utilizing Android.

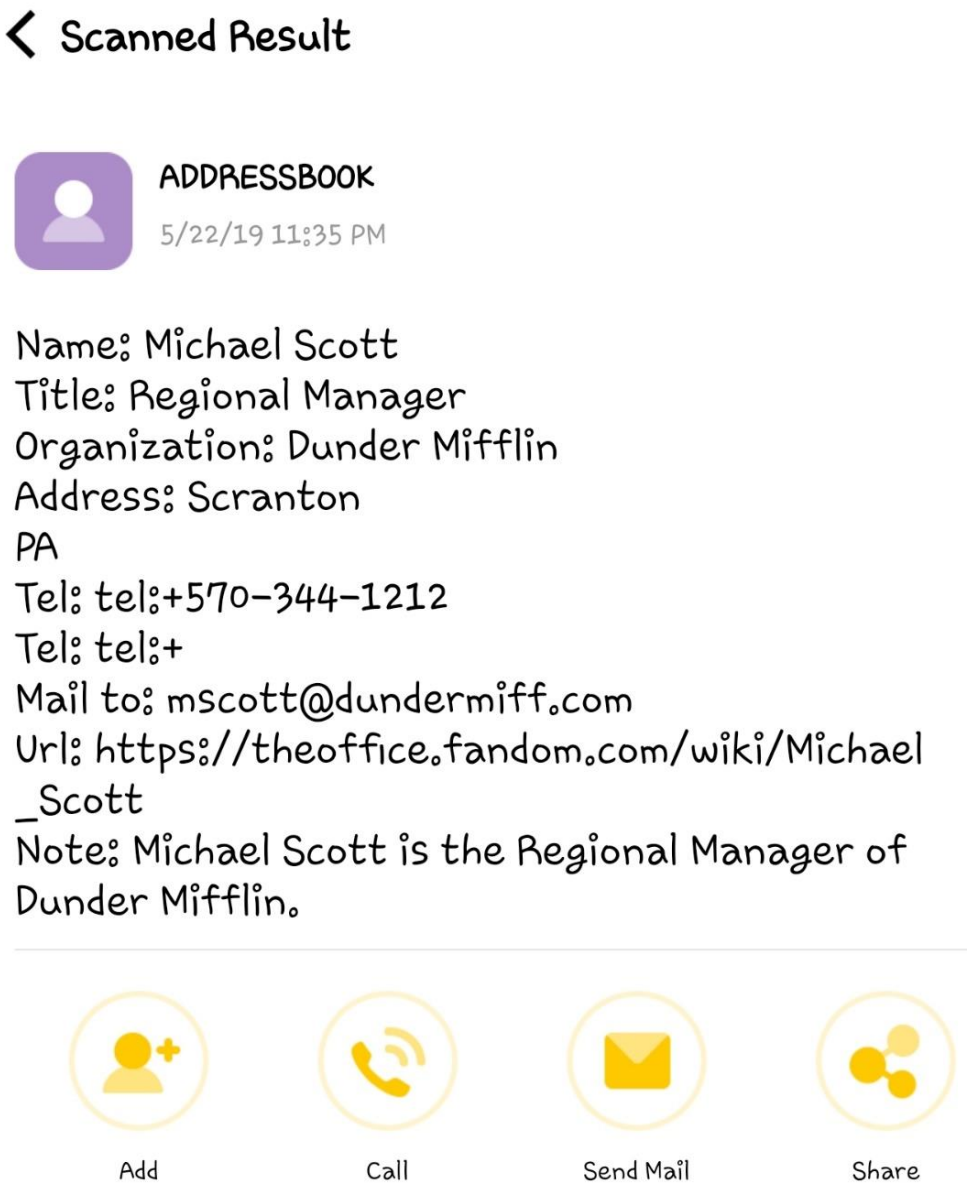

Figure 2. Scanned QR Code Import on a Samsung Smart Phone

Once the vCard information is populated, clicking “Add” will create a contact (i.e., phone book entry) that can be edited. For example, clicking “Add” on Michael Scott is demonstrated in Figure 3.

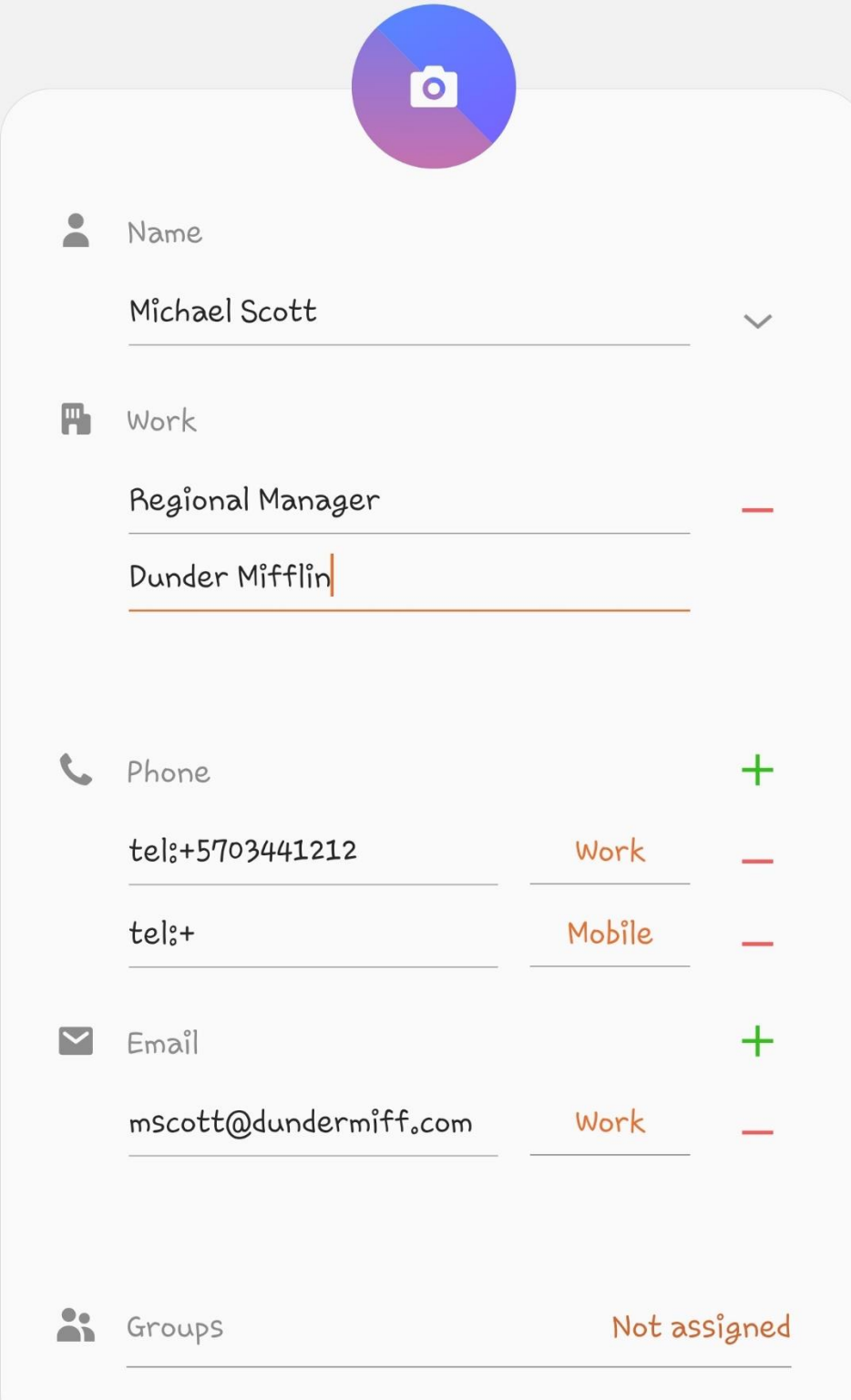

Figure 3. Creating a New Contact

If no edits are required, the contact can be saved immediately. Thus, in five seconds, a conference goer can scan a QR code, import the vCard, save the vCard as a new contact, and have all user information (represented in Table 1) imported into his phone. Figure 4 demonstrates the new contact.

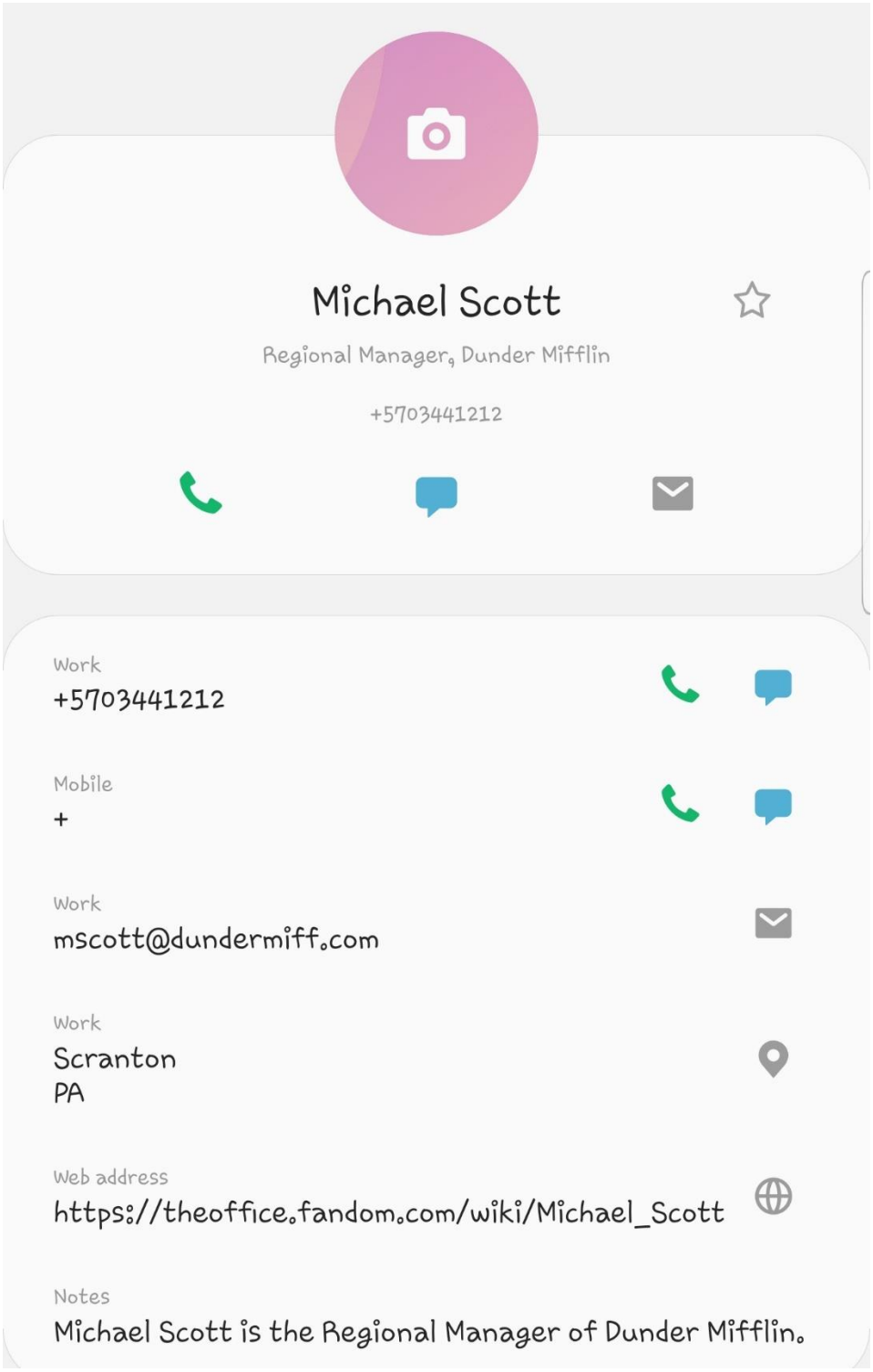

Figure 4. New Contact Saved in Phone

CREATING BADGES

Once the Attendees data set has been populated, the vCard text files have been created, and the QR code images have been created, a conference logo or graphic can be generated or selected to be included on the badge. In this example, the PharmaSUG 2019 logo is cropped from their website and saved as Pharmasug_logo.jpg, as shown in Figure 5. Note that the logo file should be saved to the same folder (&LOC) in which all other files are being saved.

Figure 5. PharmaSUG 2019 Logo

Next, a cascading style sheets (CSS) file must specify the style and formatting for the badges, which can be customized (not shown) by altering this stylesheet file. This CSS file should be saved as Badges.css (in the &LOC folder), and is called by the CSSFILE option in the subsequent SAS DATA step:

```
body {
  margin: 25px;
  background-color: rgb(256,256,256);
  font-family: arial, sans-serif;
  font-size: 14px;
}

div
{
  display: inline-block;
}

table {
  table-layout: fixed;
}

td {
  width:4in;
  height:3in;
  max-width:4in;
  min-width:4in;
  max-height:3in;
  min-height:3in;
  border:1px solid black;
  text-align: center;
  vertical-align: bottom;
}

.f1 {
  font:arial, sans-serif;
  font-size: 20pt;
  font-style: normal;
  font-weight: bold;
}
```

```
.f2 {
  font:arial, sans-serif;
  font-size: 14pt;
  font-style: italic;
  font-weight: normal;
}

.f3 {
  font:times new roman, sans-serif;
  font-size: 10pt;
  font-style: normal
  font-weight: normal;
}

.qr {
  border: 0in;
  padding: 0in;
  display: block;
  float: left;
  vertical-align: bottom;
  margin-bottom: 0px;
  bottom: 0px;
  width: 1.25in;
  height: 1.25in;
}

.logo {
  border: 0in;
  padding: 0in;
  float: right;
  vertical-align: text-bottom;
  left: 2.75in;
  height: 1.25in;
}
```

The final step (for conference organizers) is to run the DATA step that creates the badges as HTML output:

```
data _null_;
   set honeybgr.attendees end=eof;
   array line $200 line1-line5;
   length cssfile qrfile logofile $200;
   cssfile="&loc.badges.css";
   qrfile="&loc.QRcode" || strip(put(_n_,8.)) || '.png';
   logofile="&loc.pharmasug_logo.jpg";
   file "&loc.badges.html";
   if _n_=1 then do;
      put '<!DOCTYPE html>';
      put '<html>';
      put '<head>';
      put '<link rel="stylesheet" href="' cssfile '">';
      put '</head>';
      put '<body>';
      put '<table>';
      end;
   line1='<div class="f1">' || strip(firstName) || ' ' || strip(lastName)
      || '</div><br>';
   line2='<div class="f2">' || strip(jobTitle) || ', ' || strip(company)
      || '</div><br>';
   line3='<div class="f3">' || strip(city) || ', ' || strip(state)
      || '</div><br><br>';
   line4='<img class="qr" src="' || strip(qrfile) || '">';
   line5='<img class="logo" src="' || strip(logofile) || '">';
```

```
   if mod(_n_,2)=1 then put '<tr>';
   put '  <td>';
   do over line;
      put '    ' line;
      end;
   put '  </td>';
   if mod(_n_,2)=0 then put '</tr>';
   if mod(_n_,2)=1 and eof then put '</tr>';
   if eof then put '</table></body></html>';
run;
```

Badges.html is created in the &LOC folder:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="badges.css ">
</head>
<body>
<table>
<tr>
  <td>
    <div class="f1">Michael Scott</div><br>
    <div class="f2">Regional Manager, Dunder Mifflin</div><br>
    <div class="f3">Scranton, PA</div><br><br>
    <img class="qr" src="QRcode1.png">
    <img class="logo" src="pharmasug_logo.jpg">
  </td>
  <td>
    <div class="f1">Jim Halpert</div><br>
    <div class="f2">Salesman, Dunder Mifflin</div><br>
    <div class="f3">Scranton, PA</div><br><br>
    <img class="qr" src="QRcode2.png">
    <img class="logo" src="pharmasug_logo.jpg">
  </td>
</tr>
<tr>
  <td>
    <div class="f1">Dwight Schrute</div><br>
    <div class="f2">Assistant to the Regional Manager, Dunder Mifflin</div><br>
    <div class="f3">Scranton, PA</div><br><br>
    <img class="qr" src="QRcode3.png">
    <img class="logo" src="pharmasug_logo.jpg">
  </td>
  <td>
    <div class="f1">Pam Halpert</div><br>
    <div class="f2">Office Manager, Dunder Mifflin</div><br>
    <div class="f3">Scranton, PA</div><br><br>
    <img class="qr" src="QRcode4.png">
    <img class="logo" src="pharmasug_logo.jpg">
  </td>
</tr>
<tr>
  <td>
    <div class="f1">Angela Martin</div><br>
    <div class="f2">Accountant, Dunder Mifflin</div><br>
    <div class="f3">Scranton, PA</div><br><br>
    <img class="qr" src="QRcode5.png">
    <img class="logo" src="pharmasug_logo.jpg">
  </td>
  <td>
    <div class="f1">Meredith Palmer</div><br>
    <div class="f2">Supplier Relations, Dunder Mifflin</div><br>
```

```
      <div class="f3">Scranton, PA</div><br><br>
      <img class="qr" src="QRcode6.png">
      <img class="logo" src="pharmasug_logo.jpg">
    </td>
  </tr>
  </table></body></html>
```

Note that six badges are printed per page, allowing conference organizers to print multiple pages on conveniently accessible 3” x 4” perforated (or sticky) badge paper. In Figure 6, a black outline (optional) has been added only to differentiate badge edges. This and other stylistic customizations can be made by modifying the Badges.css file. Note that these badges have been shrunk, as actual badges—defined in the HTML file—would cover the entire page and thus extend beyond the margins of this document.

Figure 6. Badges.html

INPUT FILES REQUIRED TO PRODUCE BADGES

An important reminder is to enumerate the three external files that are required to create each badge, and which are also required to view the HTML output file:

1. The conference logo or image, saved as a JPG (e.g., pharmasug_logo.jpg)
2. The stylesheet file that prescribes formatting (e.g., badges.css)
3. Each QR code PNG file (e.g., qrcode1.png through qrcode6.png)

Thus, if the HTML file is to be moved to a different folder after creation, not only the HTML file but also all of the preceding files must be relocated to the same folder. For example, just moving the HTML file (but omitting the other necessary files) will display a very unhappy set of badges that lacks all graphics and formatting, as shown in Figure 7.

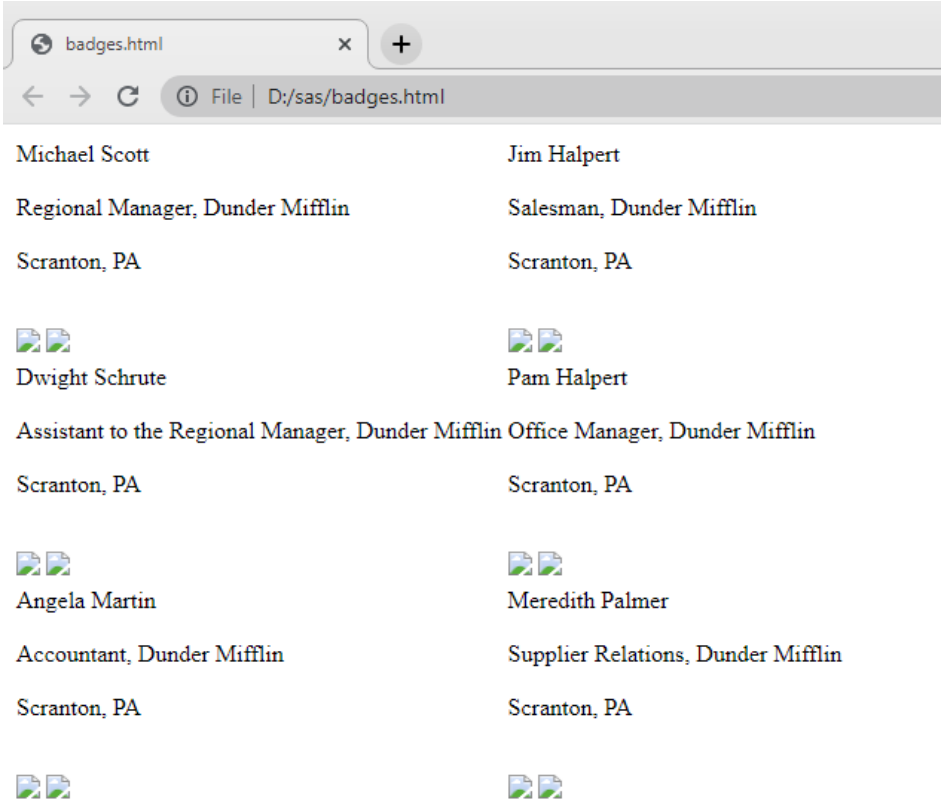

Figure 7. Badges.HTML When QR Codes, PNG Images, and Conference Logo Are Missing

Note that the VCF files do not need to be relocated, as they are only utilized to create the QR codes.

CONCLUSION

This data-driven solution to conference badge creation puts conference organizers squarely in the driver's seat. Honeybadger empowers organizers to control the text printed on each badge, the font and other style elements, and the size and location of the QR code and optional conference logo. Attendees also have control over badge information, and are able to specify what metadata should be made available to conference attendees (in the QR code) versus metadata that should be available only to conference organizers (in the conference registration database). Finally, given the often-exorbitant cost of third-party badge vendors, this home-grown solution offers a huge cost savings to conference organizers who can redirect finances toward more meaningful activities or accoutrement, such as complimentary massages, bouncy castles, and liquor luges.

REFERENCES

[i] Robert Allison. *How to create your own QR codes with SAS!* https://blogs.sas.com/content/sastraining/2016/09/12/how-to-create-your-own-qr-codes-with-sas/.

[ii] Robert Allison. Code, including his SAS macro TO_QR. http://robslink.com/SAS/democd88/qr_robslink.sas.

[iii] Google Charts Infographics: QR Codes. https://developers.google.com/chart/infographics/docs/qr_codes.

[iv] vCard Format Specification. https://tools.ietf.org/html/rfc6350.

CONTACT INFORMATION

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

Name: Troy Martin Hughes
E-mail: troymartinhughes@gmail.com