Paper 040-2012

Sometimes One Needs an Option with Unusual Dates

Arthur S. Tabachneck, Ph.D., myQNA, Inc., Thornhill, Ontario Canada
Matthew Kastin, I-Behavior, Inc., Louisville, Colorado
Xia Ke Shan, Chinese Financial Electrical Company, Beijing, China

ABSTRACT

Do you have date range analytical needs that you haven't been able to solve with the SAS® interval and holiday functions? You have if those needs involved analyzing a date range that didn't start at the beginning, end or middle of a given month, involved holidays other than the US and Canadian holidays that are incorporated in the holiday function, or if you had to analyze data related to virtually any annual sporting event. This paper shows how the intervalds option, introduced in SAS 9.2 phase 2, can be used to meet such needs and includes code for addressing fiscal years in Great Britain, Chinese, Hebrew and Islamic holidays, and for analyzing the various rounds of the NCAA March Madness basketball tournament.

THE PROBLEM

So, what do you do if you have to produce reports or conduct analyses based on data from a fiscal year that begins on April 6th and runs through April 5th of the following year, or if you need to compare the year-to-year performance of advertisements that are disseminated during the annual US college basketball March Madness tournament? Or, similarly, how could you program SAS to automatically send the various expected/appreciated Chinese, Hebrew and/or Islamic holiday email greetings, do year-to-year comparisons of business data that are related to such dates or date ranges, or if you wanted to populate another tool (like Google calendar) with such holidays and events?

THE SOLUTION

As of version 9.2 phase 2, SAS includes a system option, *INTERVALDS,* that provides a way for you to formalize any calendar in a dataset, identify the dataset in an option, and then use the various SAS interval functions to identify, report on and/or analyze any dataset based on the holidays, events or seasons that you have specified in the interval datasets. The present paper describes some potentially useful applications of that not very well known SAS system option, including code for creating calendar related datasets that capture British fiscal years, Chinese, Hebrew and Islamic calendars and holidays, and the various rounds of the NCAA March Madness basketball tournament

One could create formats to accomplish some of the above mentioned tasks but, if you only use formats, you can't take advantage of the various SAS interval functions, like *INTCINDEX, INTCK, INTCYCLE, INTFIT, INTFMT, INTGET, INTINDEX, INTNX, INTSEAS, and INTTEST*. The interval functions, many of which were new in version 9.3, allow one to obtain such things as a cycle index, the number of interval boundaries between two dates, the interval at the next higher seasonal cycle, a time interval that is aligned between two dates, a recommended format, an interval based on three values, a season index, the length of a seasonal cycle and whether an interval is valid and, in the case of the *INTNX function*, increment a date, time, or datetime value by a given time interval.

Our proposed solutions allow the use of those functions and don't require any special math or astrological skills .. just run the code that is provided in this paper.

THE INTERVALDS OPTION

The *INTERVALDS* (or interval dataset) option is a SAS system option that provides you with a way to specify one or more interval name and value pairs, where the names can be any valid SAS name and the values are the names of SAS datasets which contain the definitions of holidays or time periods. The specified intervals, we believe, can be used as arguments to any of the SAS interval functions. We should note that time did not permit our testing of all of the functions and the current (9.3) system documentation only states that the option supports the *INTCK* and *INTNX* functions. However, important to our use in the present paper, the documentation includes examples which use the *INTINDEX* function.

The *INTERVALDS* option is invoked by running the following statement:

```
options intervalds (iName1=dataset1Name <iNameN=datasetNname>);
```

where the interval name(s) (i.e., iName1 to iNameN) can be any valid SAS name and the datasets specified are SAS datasets that contain at least a date, datetime or numeric formatted variable called *BEGIN*, and may include *END* and a *SEASON* variables.

ADDRESSING NON-STANDARD FISCAL YEARS

Assume you had the following SAS dataset containing various dates:

```
data have;
  format dates date9.;
  input dates date9.;
  cards;
14jan2011
5apr2011
6apr2011
5jul2011
6jul2011
14aug2011
;
```

and that you needed to identify the fiscal year, month and quarter that each date represents, for fiscal years that begin on April 6th and end on April 5th. Creating datasets that describe the start of such fiscal years, and the seasons (e.g., months, quarters or years) that the dates represent, is a trivial task that could easily be accomplished by even beginning SAS programmers. Only two fields are needed, namely *BEGIN* and *SEASON*, where *BEGIN* indicates the first day of each fiscal year, month or quarter, and *SEASON* reflects the fiscal year, quarter or month that the dates represent. The following datastep creates month, quarter and year datasets for all years between 1900 and 2100. The *begin* variable represents the actual dates, and the season variable reflects the months, quarters and fiscal years, respectively:

```
data fyds fqds fmds;
  do begin= '06APR1900'd to '05APR2100'd;
    if day(begin) eq 6 then do;
      season=max(month(begin)-3,(month(begin)<4)*(month(begin)+9));
      end=intnx('month', begin, 1, 'same')-1;
      output fmds;
      if month(begin) eq 4 then do;
        season=year(begin);
        end=intnx('year', begin, 1, 'same')-1;
        output fyds;
      end;
      if month(begin) in (4,7,10,1) then do;
        season=max(round(month(begin)/4,1),(month(begin)<4)*4);
        end=intnx('month', begin, 3, 'same')-1;
        output fqds;
      end;
    end;
  end;
  format begin end date.;
run;
```

Note that the range we specified exceeds the desired set of intervals by one day. Insuring that the datasets include at least one cycle beyond the possible intervals of interest appears to be necessary in order for some of the interval functions to work as expected. To identify the fiscal year, quarter and month that each date represents, one only has to set the *INTERVALDS* option to reflect the desired names and datasets that delineate the needed intervals, and then use the *INTINDEX* function to return the desired data:

```
options intervalds=(FiscalMonth=FMDS  FiscalQuarter=FQDS  FiscalYear=FYDS);
data need;
  set have;
  fiscal_year =INTINDEX( 'FiscalYear',    dates );
  fiscal_qtr  =INTINDEX( 'FiscalQuarter', dates );
  fiscal_month=INTINDEX( 'FiscalMonth',   dates );
run;
```

which would result in obtaining the following file:

VIEWTABLE: Work.Need

	dates	fiscal_year	fiscal_qtr	fiscal_month
1	14JAN2011	2010	4	10
2	05APR2011	2010	4	12
3	06APR2011	2011	1	1
4	05JUL2011	2011	1	3
5	06JUL2011	2011	2	4
6	14AUG2011	2011	2	5

Figure 1 – Result of applying intervalds datasets

CREATING INTERVAL DATASETS FOR THE CHINESE, HEBREW AND ISLAMIC CALENDARS, AND THE NCAA MARCH MADNESS TOURNAMENT

Creating intervalds datasets for the other calendars addressed in this paper turned out to be a little more challenging. In addition to being familiar with each calendar's rules, and the various related religious holidays that occur throughout each year, the code also required the use of PROC FCMP to develop specialized math functions and a new function that expands the *holiday* function to include any desired holidays, the hash method to create various lookup tables, and the use of PROC FORMAT and its *cntlin* option to create various formats. Similarly, for the Chinese and March Madness calendars, the code had to include routines that downloaded essential information from the Hong Kong Observatory and Wikipedia, respectively. The code for creating the various interval datasets described in this paper are shown in Appendix I.

As an added bonus, code is included that can be implemented to automatically send email holiday greetings which can include ASCII art. The specific example included in the code sends Hanukkah email greetings on each of the holiday's eight days, including an ASCII art menorah that will have the correct number of candles lit.

FUTURE CHANGES TO THE CODE

While we did our best to only include carefully written and tested code, the code will likely have to be updated to correct for errors or web site changes that we or others might discover. As such, we created a page for the paper on sasCommunity.org. The page includes copies of the four sets of code that are described in this paper, and we will update the files on that page as necessary. Similarly, if you write code that creates additional interval datasets, and are willing to share your efforts, please feel free to add your code to that page. The page can be found at: http://www.sascommunity.org/wiki/Sometimes_One_Needs_an_Option_with_Unusual_Dates.

CONCLUSION

The original purpose of the present paper was to describe the relatively new SAS *intervalds* system option, and to provide examples of how the option could be used to aid the analysis of non-standard interval calendar related data. That task, in itself, turned out to be almost too trivial, and could have been accomplished in just a few paragraphs. However, in writing those paragraphs, we realized that a major obstacle for implementing the option for most non-standard interval calendar related data was the task of creating the necessary interval datasets. As a result, we expanded the paper to include the development of four such datasets and, hopefully, provided sufficient concepts to enable others to generalize the code to meet their own specific needs.

DISCLAIMER

The contents of this paper are the work of the authors and do not necessarily represent the opinions, practices or recommendations of the authors' organizations.

CONTACT INFORMATION

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

Arthur Tabachneck, Ph.D., President
myQNA, Inc.
80 Willowbrook Road
Thornhill, ON L3T 5K9 Canada
E-mail: atabachneck@gmail.com

Matthew Kastin
i-behavior, Inc.
2051 Dogwood Street, Suite 220
Louisville, CO 80027
E-mail: matthew.kastin@gmail.com

Xia Ke Shan
Chinese Financial Electrical Company
Beijing, China
E-mail: xiakeshan@yahoo.com.cn

APPENDIX I

/*CODE FOR CREATING INTERVAL DATASETS FOR THE HEBREW AND ISLAMIC CALENDARS*/

```
/** Hebrew and Islamic Calendars
  *
  * This program creates functions that were needed to convert SAS dates to and from
  * Hebrew and Islamic dates, interval datasets for each calendar, an expanded HOLDIAY
  * function that adds Jewish and Islamic holidays to the SAS Holiday function, and
  * code to send Hanukkah email greetings that include an ascii graphic menorah.
  *
  * If ascii art is desired for other email use, a nice collection is available at:
  * http://www.ascii-art.de/ascii/index.shtml
  *
  * AUTHORS: Matthew Kastin and Arthur Tabachneck
  * DATE: February 14, 2012
*/

proc fcmp outlib=work.func.dates;
  function hebrew_leap(year); /*Check whether Hebrew year is a leap year*/
    return(mod(((year*7)+1),19)<7);
  endsub;

  function hebrew_year_months(year); /*Get number of months in Hebrew year*/
    return(ifn(hebrew_leap(year),13,12));
  endsub;

  function hebrew_delay_1(year); /*Year cannot start on Sunday, Wednesday or Friday*/
    months=floor(((235*year)-234)/19);
    parts=12084+(13753*months);
    day=(months*29)+floor(parts/25920);
    if mod((3*(day+1)),7)<3 then day+1;
    return(day);
  endsub;

  function hebrew_delay_2(year); /*Check for delay due to adjacent year length*/
    last=hebrew_delay_1(year-1);
    present=hebrew_delay_1(year);
    next=hebrew_delay_1(year+1);
    a=ifn(next-present=356,2,ifn(present-last=382,1,0));
    return(a);
  endsub;

  function hebrew_year_days(year); /*Get number of days in given Hebrew year*/
    a=hebrew_to_jd(year+1,7,1)-hebrew_to_jd(year,7,1);
    return(a);
  endsub;

  function hebrew_month_days(year,month); /*Get number of days in given Hebrew month*/
    if month in (2,4,6,10,13) then return(29);
    else if month=12 and not(hebrew_leap(year)) then return(29);
    else if month=8 and (mod(hebrew_year_days(year),10)<>5) then return(29);
    else if month=9 and mod(hebrew_year_days(year),10)=3 then return(29);
    else return(30);
  endsub;

  function hebrew_to_jd(year,month,day); /*Convert Hebrew date to Julian date*/
    hebrew_epoch=347995.5;
    months=hebrew_year_months(year);
    jd=hebrew_epoch+hebrew_delay_1(year)+ hebrew_delay_2(year)+day+1;
    if month<7 then do;
      do mon=7 to months;
        jd+hebrew_month_days(year,mon);
      end;
      do mon=1 to month-1;
```

```
      jd+hebrew_month_days(year,mon);
    end;
  end;
  else do mon=7 to month-1;
    jd+hebrew_month_days(year,mon);
  end;
  return(jd);
endsub;

function hebrew_to_sd(year,month,day); /*Convert Hebrew date to SAS date*/
  return(hebrew_to_jd(year,month,day)-2436934.5);
endsub;

function jd_to_hebrew(jd) $10; /*Convert Julian date to Hebrew date*/
  _jd=floor(jd)+0.5;
  count=floor(((_jd-347995.5)*98496)/35975351);
  year=count-1;
  do while(_jd>=hebrew_to_jd(count,7,1));
    count+1;     year+1;
  end;
  first=ifn(_jd<hebrew_to_jd(year,1,1),7,1);
  month=first-1;
  do while(_jd>hebrew_to_jd(year,first,1));
    first+1;     month+1;
  end;
  day=(_jd-hebrew_to_jd(year,month,1))+1;
  return(catx('-',of year month day));
endsub;

function sd_to_hebrew(sd) $10; /*Convert a SAS date to a Hebrew date*/
  return(jd_to_hebrew(sd+2436934.5));
endsub;

function islamic_leap(year); /*Check whether Islamic year is a leap year*/
  return(mod((year*11)+14,30)<11);
endsub;

function islamic_to_jd(year,month,day); /*Convert Islamic date to Julian date*/
  return((day+ceil(29.5*(month-1))+(year-1)*354+
   floor((3+(11*year))/30)+1948439.5)-1);
endsub;

function islamic_to_sd(year,month,day); /*Convert Islamic date to SAS date*/
  return(islamic_to_jd(year,month,day)-2436934.5);
endsub;

function jd_to_islamic(jd) $10; /*Convert Julian date to Islamic date*/
  _jd=floor(jd)+0.5;
  year=floor(((30*(_jd-1948439.5))+10646)/10631);
  month=min(12,ceil((_jd-(29+islamic_to_jd(year,1,1)))/29.5)+1);
  day=(_jd-islamic_to_jd(year,month,1))+1;
  return(catx('-',of year month day));
endsub;

function sd_to_islamic(sd) $10; /*Convert SAS date to Islamic date*/
  return(jd_to_islamic(sd+2436934.5));
endsub;

function holiday_x(h $,y); /*Expands Holiday function*/
  select(upcase(h));
    when ('ROSH HASHANA') return(hebrew_to_sd(y+3761,7,1));
    when ('YOM KIPPUR') return(hebrew_to_sd(y+3761,7,10));
    when ('SUKKOT') return(hebrew_to_sd(y+3761,7,15));
    when ('SHMINI ATZERET') return(hebrew_to_sd(y+3761,7,22));
    when ('SIMCHAT TORAH','SIMHATH TORAH','SIMKHES TOREH')
     return(hebrew_to_sd(y+3761,7,23));
    when ('HANUKKAH','CHANUKAH','CHANUKKAH','CHANUKA')
```

```
       return(hebrew_to_sd(y+3761,9,25));
      when ('TU BISHVAT') return(hebrew_to_sd(y+3760,11,15));
      when ('PURIM') return(ifn(hebrew_leap(y+3760),
       hebrew_to_sd(y+3760,13,14),hebrew_to_sd(y+3760,12,14)));
      when ('PESACH','PASSOVER','PESAH','PESAKH','PEYSEKH','PAYSOKH')
       return(hebrew_to_sd(y+3760,1,15));
      when ('SEFIRAH',"SEFIRAT HA'OMER") return(hebrew_to_sd(y+3760,1,16));
      when ('YOM HASHOAH','HOLOCAUST REMEMBRANCE DAY')
       return(hebrew_to_sd(y+3760,1,27));
      when ('YOM HAZIKARON','ISRAEL MEMORIAL DAY') return(hebrew_to_sd(y+3760,2,4));
      when ("YOM HA'ATZMAUT",'YOM HAATZMAUT','YOM HA ATZMAUT',
       'ISRAEL INDEPENDENCE DAY') return(hebrew_to_sd(y+3760,2,5));
      when ("LAG BA'OMER","LAG LA'OMER",'LAG BAOMER','LAG LAOMER')
       return(hebrew_to_sd(y+3760,2,18));
      when ('YOM YERUSHALAYIM','JERUSALEM DAY') return(hebrew_to_sd(y+3760,2,28));
      when ('SHAVUOT','SHAVUOS',"SHABHU'OTH") return(hebrew_to_sd(y+3760,3,6));
      when ("TISHA B'AV",'TISHA BAV','TISHA B AV') return(hebrew_to_sd(y+3760,5,9));
      when ("TU B'AV",'TU BAV','TU B AV') return(hebrew_to_sd(y+3760,5,15));
      when ('ISLAMIC NEW YEAR','HIJRI NEW YEAR','RAS AS-SANAH AL-HIJRIYAH')
       return(islamic_to_sd(y-579,1,1));
      when ('DAY OF ASHURA','ASHURA','ASHOURA') return(islamic_to_sd(y-579,1,10));
      when ('MAWLID','MAWLIDU N-NABIYYI','MAWLID AN-NABI','MEVLID','MEVLIT','MULUD')
       return(islamic_to_sd(y-579,3,12));
      when ('LAYLAT AL-MIRAJ','LAILAT AL MIRAJ','SHAB-E-MIRAJ','MIRAC KANDILI')
       return(islamic_to_sd(y-579,7,27));
      when ("MID-SHA'BAN",'MID-SHABAN','MID-SHA BAN','LAYLATUL BARAAH','SHAB-E-BARAT',
       "LAYLATUL BARA'AH",'LAYLATUL BARA AH') return(islamic_to_sd(y-579,8,15));
      when ('RAMADAN','RAMAZAN') return(islamic_to_sd(y-579,9,1));
      when ('LAYLAT AL-QADR',"LAILATUL QADR'",'LAILATUL QADR','SHAB-E-QADR')
       return(islamic_to_sd(y-579,9,27));
      when ('EID-UL-FITR','EID AL-FITR','ID-UL-FITR','ID AL-FITR','EID','IDU I-FITR')
       return(islamic_to_sd(y-579,9,ifn(islamic_leap(y-579),30,29)));
      when ('EID AL-ADHA',"EID AL-ADHA'",'FESTIVAL OF SACRIFICE','GREATER EID')
       return(islamic_to_sd(y-579,12,10));
      when ('BOXING','BOXING DAY') return(holiday('BOXING',y));
      when ('CANADA','CANADA DAY') return(holiday('CANADA',y));
      when ('CANADAOBSERVED') return(holiday('CANADAOBSERVED',y));
      when ('CHRISTMAS') return(holiday('CHRISTMAS',y));
      when ('COLUMBUS','COLUMBUS DAY') return(holiday('COLUMBUS',y));
      when ('EASTER') return(holiday('EASTER',y));
      when ('FATHERS','FATHERS DAY') return(holiday('FATHERS',y));
      when ('HALLOWEEN') return(holiday('HALLOWEEN',y));
      when ('LABOR') return(holiday('LABOR',y));
      when ('MLK','MARTIN LUTHER KING') return(holiday('MLK',y));
      when ('MEMORIAL','MEMORIAL DAY') return(holiday('MEMORIAL',y));
      when ('MOTHERS','MOTHERS DAY') return(holiday('MOTHERS',y));
      when ('NEWYEAR','NEW YEAR','NEW YEARS','NEW YEARS DAY')
       return(holiday('NEWYEAR',y));
      when ('THANKSGIVING','THANSGIVING DAY') return(holiday('THANKSGIVING',y));
      when ('THANKSGIVINGCANADA') return(holiday('THANKSGIVINGCANADA',y));
      when ('USINDEPENDENCE','INDEPENDENCE DAY','JULY 4TH')
       return(holiday('USINDEPENDENCE',y));
      when ('USPRESIDENTS','PRESIDENTS DAY'return(holiday('USPRESIDENTS',y));
      when ('VALENTINES','VALENTINES DAY') return(holiday('VALENTINES',y));
      when ('VETERANS','VETERANS DAY','REMEMBRANCE','REMEMBERANCE','REMEMBRANCE DAY',
       'REMEMBERANCE DAY') return(holiday('VETERANS',y));
      when ('VETERANSUSG') return(holiday('VETERANSUSG',y));
      when ('VETERANSUSPS') return(holiday('VETERANSUSPS',y));
      when ('VICTORIA','VICTORIA DAY') return(holiday('VICTORIA',y));
      otherwise return(.);
    end;
  endsub;
run;

options cmplib=work.func;

proc format; /*Create format to map holidays and their length*/
```

```
  value $jhol2days
  'ROSH HASHANA'    =2  'YOM KIPPUR'       =1  'SUKKOT'                  =6
  'SHMINI ATZERET'  =1  'SIMCHAT TORAH'    =1  'HANUKKAH'                =8
  'TU BISHVAT'      =1  'PURIM'            =1  'PESACH'                  =8
  'SEFIRAH'         =49 'YOM HASHOAH'      =1  'ISRAEL INDEPENDENCE DAY' =1
 'LAG BAOMER'       =1  'YOM YERUSHALAYIM' =1  'SHAVUOT'                 =2
  "TISHA B'AV"      =1  "TU B'AV"          =1  other                     =0;
run;

/* Create dataset and format. JHOLS contains holiday dates and the format provides */
/* the correspondence between interval numbers and holiday names                   */

data s2jhol(drop=end) jhols;
  declare hash hol();
  hol.definekey('begin');
  hol.definedata('holiday','season');
  hol.definedone();

  array h[16] $30 ('ROSH HASHANA'      'YOM KIPPUR'              'SUKKOT'
                   'SHMINI ATZERET'    'SIMCHAT TORAH'           'HANUKKAH'
                   'TU BISHVAT'        'PURIM'                   'PESACH'
                   'YOM HASHOAH'       'ISRAEL INDEPENDENCE DAY' 'LAG BAOMER'
                   'YOM YERUSHALAYIM'  'SHAVUOT' "TISHA B'AV"    "TU B'AV");

  do year=2000 to 2100;
    season=0;
    do i=1 to dim(h);
      holiday=h[i];
      begin=holiday_x(h[i],year);
      season+1;
      hol.add();
      if year=2000 then output s2jhol;
    end;
  end;

  do begin='01JAN2000'd to '31DEC2100'd;
    holiday='';
    end=begin;
    rc=hol.find();
    if (rc = 0) then do;
      end=begin+put(holiday,$jhol2days.)-1;
      output jhols;
    end;
    else do;
      season=0;  output jhols;
    end;
  end;

  keep begin end season holiday;
  format begin end date11.;
run;

data s2jhol;
  retain fmtname 's2jhol' type 'n';
  set s2jhol (drop=begin rename=(season=start holiday=label));
run;

proc format cntlin=s2jhol;  run;

/*Set intervalds option to identify Jewish holidays using the jhols dataset*/
options intervalds=(JewishHolidays=jhols);

%macro sendit(to,subj,from,test=NO);
  /*This macro sends emails on Jewish holidays including ASCII art for Hanukkah*/
  %if &test. eq YES %then %let today=%sysevalf("12DEC2012"d);
  %else %let today=%sysfunc(today());
  %if %sysfunc(intindex(JewishHolidays,&today.)) %then %do;
```

```
    options EMAILSYS=SMTP
            EMAILID="your_email_address"
            EMAILPW="your_email_password"
            EMAILHOST="your_email_smtp_address"
            EMAILAUTHPROTOCOL=LOGIN
            EMAILPORT=587; /* Note: EMAILPORT may have to set to 25 */
    filename mymail email lrecl=256 TYPE='TEXT/HTML';
    data _null_;
      file mymail to=("&to.") subject="&subj.";
      array candles[17] $40. (17*' ');
      put "<BR>";
      put "<B><font face='courier new'>";
      x=intindex('JewishHolidays',&today.);
      msg=cat('Hope you have a Happy ', propcase(strip(put(x,s2jhol.))),'!');
      put msg "<BR> <BR>";
      if put(x,s2jhol.) eq 'HANUKKAH' then do;
        ncandles=2*(&today.-intnx('JewishHolidays',&today.,0))+1;
        if ncandles gt 7 then ncandles+2;
        candles[9]=")";
        msg=catx(' ',of %quote(candles:));
        put "<font face='courier new' color=red>";
        put msg "<BR>";
        do i=1 to ncandles by 2;
          candles[i]=")";
          if i eq 7 then i=9;
        end;
        put "<font face='courier new' color=red>";
        msg=catx(' ',of %quote(candles1-candles8));
        put msg;
        put "<font face='courier new' color=black>";
        put %quote(candles(9));
        put "<font face='courier new' color=red>";
        msg=catx(' ',of %quote(candles10-candles17));
        put msg "<BR>";
        do i=1 to 17 by 2;  candles[i]="|";  end;
        msg=catx(' ',of %quote(candles:));
        put "<font face='courier new' color=black>";
        put msg "<BR>";
        do i=2 to 16 by 2;  candles[i]="_";  end;
        msg=catx(' ',of %quote(candles:));
        put msg "<BR>";
        do i=1 to 17;
          if i eq 8 then i=11;
          candles[i]=' ';
        end;
        msg=catx(' ',of %quote(candles:));
        put msg "<BR>";
      end;
      put "</B></font><BR>";
      put "&from";
      put '</style></body></html>';
    run;
  %end;
%mend;

/* Example usage */
%sendit(EmailAddress,Happy Holidays,YourName,test=YES) /* If only testing */
%sendit(EmailAddress,Happy Holidays,YourName) /* Actual use */

/*or, to send a number of emails*/
data _null_;
  informat email $30.;
  input email &;
  call execute('%sendit('||strip(email)||',Happy Holidays,YourName)');
  cards;
art297@rogers.com
;
```

/*CODE FOR CREATING INTERVAL DATASETS FOR THE CHINESE AGRICULTURAL CALENDAR*/

```
/** This code creates interval datasets for using interval functions for dates from
  * the Chinese Agricultural calendar.  The code uses a lunar calendar from the Hong
  * Kong Obervatory and creates datasets for years, months, and most Chinese festivals
  *
  *AUTHORS: Xia Ke Shan and Arthur Tabachneck
  *DATE: February 13, 2012
*/

proc delete data=dates;
run;

/*The following macro downloads pages from the Hong Kong Observatory which contain all
lunar dates between January 1st, 1901 and December 31st, 2100  */
%macro getcdates;
  %do i=1901 %to 2100;
    filename lunar http
     "http://www.hko.gov.hk/gts/time/calendar/text/T&i.e.txt";
    data temp;
      format Gregorian_date yymmdd10.;
      infile lunar dlm=' ' firstobs=3 truncover expandtabs ;
      input Gregorian_date : ?? yymmdd12. lunar_date & $40.
            day_of_week : $20.            solar_term &  $40.;
      if not missing(Gregorian_date) then output;
    run;

    proc append base=dates data=temp;
    run;
  %end;

  /*The following corrects for dates missing from the Hong Kong Observatory*/
  data all;
    do Gregorian_date='01jan1901'd to '31dec2100'd;
      output;
    end;
  run;

  data dates;
    merge dates all;
    by Gregorian_date;
  run;
%mend getcdates;

%getcdates

data dates;
  set dates;
  /* The Hong Kong Observatory data begins at 01JAN1901 which is in Chinese month 11*/
  retain chinese_month 11;
  if find(upcase(lunar_date),'MONTH') then chinese_month=
   input(compress(lunar_date, ,'kd'),?? best8.);
run;

data dates;
  set dates;
  if chinese_month ne lag(chinese_month) then chinese_day=0;
  chinese_day+1;
  if _n_ eq 1 then chinese_day=11;*start from 01JAN1901;
run;

data Chinese_Calendar;
  set dates;
  if _n_ eq 1 then chinese_year=1900;
  select;
    when(chinese_month=1 and chinese_day=1) do;
```

```
      season=1;
      chinese_year+1;
    end;
    when(chinese_month=1 and chinese_day=15)     season=2;
    when(find(lowcase(solar_term),'bright') and
         find(lowcase(solar_term),'clear' ) )    season=3;
    when(chinese_month=5 and chinese_day=5)      season=4;
    when(chinese_month=7 and chinese_day=7)      season=5;
    when(chinese_month=7 and chinese_day=15)     season=6;
    when(chinese_month=8 and chinese_day=15)     season=7;
    when(chinese_month=9 and chinese_day=9)      season=8;
    when(chinese_month=10 and chinese_day=15)    season=9;
    when(find(lowcase(solar_term),'winter') and
         find(lowcase(solar_term),'solstice') )  season=10;
    when(chinese_month=12 and chinese_day=23)    season=11;
    otherwise                                    season=0;
  end;
run;

/* Create Interval Datasets Chinese_Month, Chinese Holidays and Chinese_Holidays */
data Chinese_Month;
  set Chinese_Calendar(keep=Gregorian_date chinese_month
    rename=(Gregorian_date=begin chinese_month=season));
  by season notsorted;
  if first.season;
  format begin date9.;
run;

data Chinese_Holidays;
  set Chinese_Calendar(keep=Gregorian_date season
    rename=(Gregorian_date=begin));
  by season notsorted;
  if first.season;
  format begin date9.;
run;

data Chinese_Years(drop=chinese_year);
  set Chinese_Calendar(keep=Gregorian_date chinese_year
                       rename=(Gregorian_date=begin));
  select (mod(chinese_year,12));
    when (0)  season=1;
    when (1)  season=2;
    when (2)  season=3;
    when (3)  season=4;
    when (4)  season=5;
    when (5)  season=6;
    when (6)  season=7;
    when (7)  season=8;
    when (8)  season=9;
    when (9)  season=10;
    when (10) season=11;
    when (11) season=12;
    otherwise season=0;
  end;
  format begin date9.;
run;

data Chinese_Years;
  set Chinese_Years;
  by season notsorted;
  if first.season;
  format begin date9.;
run;

/*Create formats for Chinese Holidays and Chinese Years */
proc format;
  value ch(default=40)
```

```
      1='Chinese New Year -- ChunJie'
      2='Lantern Festival -- YuanXiaoJie'
      3='Qingming Festival -- QingMingJie'
      4='Dragon Boat Festival -- DuanWuJie'
      5='Night of Sevens -- QiXi'
      6='Ghost Festival -- ZhongYuanJie'
      7='Mid-Autumn Festival -- ZhongQiuJie'
      8='Double Ninth Festival -- ChongYangJie'
      9='Xia Yuan Festival -- XiaYuanJie'
      10='Winter Solstice Festival -- DongZhi'
      11='Kitchen God Festival -- XiaoNian';

  value cy(default=20)
      1='the Monkey Year'
      2='the Rooster Year'
      3='the Dog Year'
      4='the Pig Year'
      5='the Rat Year'
      6='the Ox Year'
      7='the Tiger Year'
      8='the Rabbit Year'
      9='the Dragon Year'
      10='the Snake Year'
      11='the Horse Year'
      12='the Goat Year';
run;

/*Set the intervalds option to identify Chinese months, holidays and years  */
options intervalds=(cm=Chinese_Month  ch=Chinese_Holidays  cy=Chinese_Years);
```

/*CODE TO CREATE AN INTERVAL DATASET FOR THE NCAA MARCH MADNESS ROUNDS*/

```
/** This code creates an interval dataset that will allow one to use interval
  * functions on dates from the NCAA March Madness Basketball tournament
  *
  * AUTHOR: Arthur Tabachneck
  * DATE: December 12, 2011
*/

proc delete data=dates;
run;

/* The following macro downloads the tournament start dates from Wikipedia  */
%macro getmdates;
  %do i=1985 %to 2012;
    filename ncaa http
     "http://en.wikipedia.org/wiki/&i._NCAA_Men%27s_Division_I_Basketball_Tournament";
    data temp (keep=startdate enddate);
      format startdate enddate date9.;
      informat sdate $21.;
      infile ncaa lrecl=32000;
      input @ "began on " sdate &;
      if sdate ne "" then do;
        x=findc(sdate,',',2,'b');
        startdate=input(substr(sdate,1,x),anydtdte21.);
        if &i. ge 2011 then do;
          startdate=startdate+2;  enddate=startdate+20;
        end;
        else if &i. ge 2001 then do;
          startdate=startdate+1;  enddate=startdate+19;
        end;
        else enddate=startdate+18;
        output;
        stop;
      end;
    run;
```

```
    proc append
      base=dates data=temp;
    run;
  %end;
%mend getmdates;
%getmdates

/* Create an Interval Dataset that captures the dates of each Tournament round */
data roundds (keep=begin season);
  set dates;
  round=0;
  i=0;
  do begin= mdy(1,1,year(startdate)) to mdy(12,31,year(startdate));
    if startdate<=begin<=enddate then do;
      i+1;
      if i in (1,3,8,10,17,19) then round+1;
      if i in (1:4,8:11,17,19) then season=round;
      else season=0;
    end;
    else season=0;
    output;
  end;
  format begin date.;
run;

data roundds (keep=begin season);
  set roundds end=eof;
  output;
  if eof then do;
    begin='01jan2012'd;  season=0;
    output;
  end;
run;

options intervalds=(Rounds=roundDS);
```

/*CODE TO CREATE INTERVAL DATASETS FOR BRITISH FISCAL YEARS*/

```
/**
  *This program is designed to create interval datasets that will allow one to use
  *interval functions on dates from a fiscal year that begins on April 6th and ends
  *on April 5th.
  *
  *
  *AUTHOR: Arthur Tabachneck
  *DATE: September 8, 2011
  *
  */

/**
  * Create Interval Datasets for year, quarter and month
  *
  */

data fyds fqds fmds;
  do begin= '06APR1900'd to '05APR2100'd;
    if day(begin) eq 6 then do;
      season=max(month(begin)-3,(month(begin)<4)*
       (month(begin)+9));
      end=intnx('month', begin, 1, 'same')-1;
      output fmds;
      if month(begin) eq 4 then do;
        season=year(begin);
        end=intnx('year', begin, 1, 'same')-1;
        output fyds;
      end;
```

```
      if month(begin) in (4,7,10,1) then do;
        season=max(round(month(begin)/4,1),
         (month(begin)<4)*4);
        end=intnx('month', begin, 3, 'same')-1;
        output fqds;
      end;
    end;
  end;
  format begin end date.;
run;

/**
  * Set intervalds option
  *
  *  The following line is all that is needed in order to establish FiscalQuarter,
  *  FiscalYear and FiscalMonth as valid intervals using the FQDS, FYDS and FMDS
  *  interval datasets, respectively
  *
  */

options intervalds=(FiscalQuarter=FQDS
                    FiscalYear=FYDS
                    FiscalMonth=FMDS);

/**
  *Test using sashelp.pricedata and the intindex function
  *
  */

data need (keep=date fiscal: next:);
  set sashelp.pricedata;
  fiscal_year=INTINDEX( 'FiscalYear', date );
  fiscal_qtr=INTINDEX( 'FiscalQuarter', date );
  fiscal_month=INTINDEX( 'FiscalMonth', date );
  next_year_start=intnx('FiscalYear',date,1,'b');
  next_year_middle=intnx('FiscalYear',date,1,'m');
  next_year_end=intnx('FiscalYear',date,1,'e');
  format date next: date9.;
run;
```