Paper 15-2019

Counting the Days Until …

Robert Ellsworth, Ellsworth Stewart Consulting Inc.

ABSTRACT

The paper covers setting up base SAS to do date calculations based on business days. The paper walks through creating a business day interval and working with intck and intnx to count and increment dates based on business days.

INTRODUCTION

Working with date and datetime fields in SAS. SAS provides some powerful date functions. We are going to look at 2 of these functions Intnx and Intck.

FUNCTIONS

- Intnx function allows you to add or subtract intervals from a given date or datetime.
- Intck function allows you to calculate the number of intervals between 2 dates or datetimes.

INTERVALS

Intnx and intck get their power with the use of intervals. There are both system defined and user defined intervals.

- Default date intervals are day, week, weekday, tenday, semimonth, month, qtr, semiyear, and year.
- Default datetime intervals are dtday, dtweek, dtweekday, dttenday, dtsemimonth, dtmonth, dtqtr, dtsemiyear, and dtyear.
- Default time intervals are second, minute, and hour.

FIND NEXT PERIOD

How do you find the last day of the prior month based on today's date.

```
mth_end= intnx('month',today(),-1,'e');
```

Figure 1 Example last day of last month

INTNX PARAMETERS

What do the parameters for intnx in the above example mean.

- Parameter 1 is the interval. (month)
- Parameter 2 is the date to increment. (today)
- Parameter 3 is the increment. (-1 for last month)
- Parameter 4 is the alignment in the period. (e -end)

INTNX ALIGNMENT

Alignments used in intnx are:

- B –First date in the period
- M –Middle date in the period.
- E –Last date in the period.
- S –Same day in the period.
- Default alignment is b.

FIND THE DIFFERENCE IN 2 DATES

How do you calculate the number of months between the start and end date of an activity on a row in the data set.

```
len_in_mths = intck('month',start_dt,end_dt,'c');
```

INTCK PARAMETERS

What do the parameters for intck in the above example mean.

- Parameter 1 is the interval. (month)
- Parameter 2 is the start date. (start_dt)
- Parameter 3 is the end date. (end_dt)
- Parameter 4 is the method. (c -continuous)

INTCK METHOD

Methods used are:

- C –Continuous
  - will use the start date as the beginning of the interval
  - will count complete intervals between the start and end dates
- D –Discrete
  - will count the interval boundaries crossed.
- Default method is discrete.

MODIFING INTERVALS BOUNDARIES

A suffix can be used to modify the interval definition.

The format is intervalX.Y

- X is the number of base intervals
- Y is the starting point in the interval.

EXAMPLES

- Day3 is a 3 day interval.
- Week.7 is a weekly interval starting on Saturday instead of Sunday.

USING SUFFIX MODIFIER

How do you calculate the current fiscal quarter. The fiscal year end is Oct 31. First calculate the last day of the prior fiscal year. then count the number of quarters since then.

```
fyear_end= intnx('year.11',today(),-1,'e');
qtr= intck('qtr.2',fyear_end,today());
```

Figure 2 code that calculates current fiscal quarter

MODIFIED INTERVALS

In the above example the interval year.11 means a year interval starting in November, the 11th month. The interval qtr.2 means 3 month interval starting on Feb 1st, the 2nd month. No method is specified so discrete is used.

INTERVALS OTHER THAN THE DEFAULTS

Given a banking day is any day that is not a weekend or holiday.

How do you find the last banking day in the current month?

```
last_bday_of_mth= intnx('BankingDays',intnx('month',dt,-1,'e'),0);
```

Figure 3 Calculate the last banking day of the month

How do you count the number of banking days in the current month?

```
bd_cnt= intck('BankingDays',intnx('month',dt,-1,'e'),
                                    intnx('month',dt,0,'e'));
```

Figure 4 Calculate the number of banking days in the month

CREATING A CUSTOM INTERVAL

To create a custom interval you need to:

- Build a data set with the boundaries of the custom interval.
- Using the options statement define the custom interval to SAS

BUILDING THE DATA SET

To build a banking days dataset we use SAS holidays function and nwkdomfunction to determine the dates of the holidays for year. we loop through all the weekdays from the start year to the end year, output the banking dates into the dataset.

```
data BankDayDS(keep=BEGIN);
  startyear= 1998;
  stopyear= 2022;
  do year = startyear to stopyear;
    /*get holidays for the year shifting those that land on a weekend*/
    %moveday(NEWYEAR);
    %movefwdday(CANADA);
    %movefwdday(VETERANS);
    %moveday(CHRISTMAS);
    %moveday2(BOXING);
    nwkdays= INTCK('WEEKDAY',MdY(1,1,year),mdy(12,31,year));
    do i= 0 to nwkdays;
      /*loop through the weekdays outputting non holidays */
      BEGIN = INTNX('WEEKDAY',MdY(1,1,year),i);
      if BEGIN ne NEWYEAR and
         BEGIN ne HOLIDAY("EASTER",year)-2 and
         BEGIN ne CANADA and /*canadaday*/
         BEGIN ne HOLIDAY("LABOR",year) and
         BEGIN ne HOLIDAY("VICTORIA",year) and
         BEGIN ne VETERANS and /*remembrance day*/
         BEGIN ne HOLIDAY("THANKSGIVINGCANADA",year) and
         BEGIN ne CHRISTMAS and
         BEGIN ne BOXING and
         begin ne nwkdom(1, 2, 8, year) and /*civic hoiday*/
         (begin ne nwkdom(3, 2, 2, year) or year < 2013) /*family day*/
          Then output;
    end;
    format BEGIN DATE.;
  end;
run;
```

Figure 5 Build banking day dataset

```
%macro moveday(day);
  &day = HOLIDAY("&day",year);
  if weekday(&day) = 1 then &day + 1; /*SUNDAY*/
  else if weekday(&day) = 7 then &day + (-1); /*SATURDAY*/
%mend;
```

Figure 6 Shift holiday off to Friday or Monday

```
%macro moveday2(day);
  &day = HOLIDAY("&day",year);
  if weekday(&day) in (1,2) then &day + 1; /*SUNDAY OR MONDAY*/
  else if weekday(&day) = 7 then &day + 2; /*SATURDAY*/
%mend;
```

Figure 7 Shift holiday to Monday

```
%macro movefwdday(day);
&day = HOLIDAY("&day",year);
if weekday(&day) in (1) then &day + 1; /*SUNDAY*/
else if weekday(&day) = 7 then &day + 2; /*SATURDAY*/
%mend;
```

Figure 8 Shift holiday to Monday or Tuesday

Obs	BEGIN
5019	02JAN18
5020	03JAN18
5021	04JAN18
5022	05JAN18
5023	08JAN18
5024	09JAN18
5025	10JAN18
5026	11JAN18
5027	12JAN18
5028	15JAN18

Figure 9 Subset of banking day dataset

THE BEGINNING AND END

Business Day:	X	X			X	X
Date:	11-Jan-18	12-Jan-18	13-Jan-18	14-Jan-18	15-Jan-18	16-Jan-18
Day:	Thursday	Friday	Saturday	Sunday	Monday	Tuesday

Figure 10 Banking day chart

The alignment for banking day is dependent on the boundaries defined (rows) in the dataset. The beginning is the calculated by moving back in time until you reach a boundary. Example if you start with Jan 13th and ask for the beginning of the current banking day period you will get Jan 12th. The end is calculated by moving forward in time until the next day is a boundary. Example start on Jan 12th and ask for the end of the current period you will get Jan 14th.

DEFINING THE BANKING DAY INTERVAL

Once the banking day data set is created, it can be used to create a banking day interval. Here is the code to define it in your program

```
libname dgamacro '/td/edw/general/general1/edwcore/dga_macro/';
options intervalds=(BankingDays=dgamacro.BankDayDS);
```

Figure 11 Define banking day interval

USING THE BANKING DAY INTERVAL

The following are examples date calculations using the banking day interval.

```
first_bday_of_mth= intnx('BankingDays',intnx('month',dt,-1,'e'),1);
```

Figure 12 The first banking day of the month

```
last_bday_of_mth= intnx('BankingDays',intnx('month',dt,-1,'e'),0);
```

Figure 13 The last banking day of the prior month

```
first_bday_of_week= intnx('BankingDays',intnx('week',dt,0,'b'),1);
```

Figure 14 The first banking day of the week

```
last_bday_of_week= intnx('BankingDays',intnx('week',dt,0,'e'),0);
```

Figure 15 The last banking day of the week

```
libnamedgamacro'/td/edw/general/general1/edwcore/dga_macro/';
optionsintervalds=(BankingDays=dgamacro.BankDayDS);
data_null_;
  dt= '3aug2015'd;
  first_bday_of_mth= intnx('BankingDays',intnx('month',dt,-1,'e'),1);
  put first_bday_of_mth= date9.;

  last_bday_of_mth= intnx('BankingDays',intnx('month',dt,-1,'e'),0);
  put last_bday_of_mth= date9.;

  first_bday_of_week= intnx('BankingDays',intnx('week',dt,0,'b'),1);
  put first_bday_of_week= date9.;

  last_bday_of_week= intnx('BankingDays',intnx('week',dt,0,'e'),0);
  put last_bday_of_week= date9.;
run;

Result
first_bday_of_mth=04AUG2015
last_bday_of_mth=31JUL2015
first_bday_of_week=04AUG2015
last_bday_of_week=07AUG2015
```

Figure 16 Calculating different banking days

Calculating Number of Banking Days

CALCULATE BANKING DAYS BETWEEN 2 DATES.

We can use the new banking day interval and intck to count banking days between 2 dates. We need to be careful when counting banking days if the start day is a banking day it is not counted.

```
bd_cnt= intck('BankingDays',st_dt,end_dt);
```

Figure 17 Banking Days Between 2 dates

```
   bd_cnt=
intck('BankingDays',intnx('week',dt,0,'b'),intnx('week',dt,0,'e'));
```

Figure 18 Banking Days in the week

```
   bd_cnt= intck('BankingDays',intnx('month',dt,-
1,'e'),intnx('month',dt,0,'e'));
```

Figure 19 Banking Days in the month

```
   data_null_;
     /* days between 2 dates */
     st_dt= '28jun15'd;
     end_dt= '4jul15'd;
     bd_cnt= intck('BankingDays',st_dt,end_dt);
     put bd_cnt;

     /* days in the week */
     dt= '28jul2015'd;
     bd_cnt= intck('BankingDays',intnx('week',dt,0,'b'),
                                 intnx('week',dt,0,'e'));
     put bd_cnt;

     /* days in the month */
     dt= '28may2015'd;
     bd_cnt= intck('BankingDays',intnx('month',dt,-1,'e'),
                                intnx('month',dt,0,'e'));
     put bd_cnt;
   run;

   Result
   4
   5
   20
```

CONTACT INFORMATION

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

Robert Ellsworth
Ellsworth Stewart Consulting Inc.
416-414-1172
rob@escorp.ca