*SAS program written by Elisabeth Winkler, Cancer Prevention Research Centre, School of Public Health, University of Queensland version dated 24/11/2015;*  
*With substantial thanks to Danielle Bodicoat, Charlotte Edwardson and Kishan Bakrania;*  
*email: e.winkler@sph.uq.edu.au;*  
*Disclaimer: this code is provided in the hope that it will be helpful, but without any warranty or implied warranty of fitness for a particular purpose;*  
*Sleep non-wear algorithm version B is contained within this program;*  
*Optional steps to aid quality control and for easy use are included in the program*  
-creating csv files: listing valid periods of data to keep, the periods estimated as sleep/non-wear, events-like files with the algorithm classifications  
-creating plots (heatmaps): to aid quality control  
**PLEASE NOTE** - these other features in the program, especially involving the output delivery system (ods) will not work on all SAS versions/set-ups and were included as an example only;  

```sas
%macro algvb(fname, id);
*PART 1: read in the events files, identify relevant identifiers for bout, participant, date and time;
data WORK.events; %let _EFIERR_ = 0; * error detection macro variable;
infile "$dir\$fname..csv" delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2 ;
informat Datetimevar best32.; informat Datacount best32. ; informat Intervalold_s best32. ; informat Activity_code 1.0 ;
informat CumulativeStepCount best32. ; informat ActivityScore_METhr best32. ;
  format Datetimevar best32.; format Datacount best12. ; format Intervalold_s best12. ; format Activity_code 1.0 ;
  format CumulativeStepCount best8. ; format ActivityScore_METhr best12.;
input Datetimevar Datacount Intervalold_s Activity_code CumulativeStepCount ActivityScore_METhr ;
if _ERROR_ then call symputx('_EFIERR_','1'); * error detection macro variable;
run;

data events; set events; if Datetimevar in (0,. ) then delete; *remove rows of 0s from end of some activpal event files;
boutid = _N_; idcode = "$id"; id = input (idcode, best12.);  
sasdatetimenumber = (Datetimevar - 21916)*86400; *convert exceldate to sasdatetime;
CumulativeStepCount_i = CumulativeStepCount; steps_n = CumulativeStepCount - lag(CumulativeStepCount_i); if boutid = 1 then steps_n = CumulativeStepCount ; drop CumulativeStepCount_i ; *non cumulative steps;
run;

*recalculate intervals as time between each bout -> more precise than the rounded data;*  
proc sort data =events; by id descending boutid; run;

data events; set events;
boutidR = _N_;  
sasdatetimenumber_endbout = lag(sasdatetimenumber); if sasdatetimenumber_endbout = . & sasdatetimenumber ne . then
  sasdatetimenumber_endbout = sasdatetimenumber + Intervalold_s; Interval_s = sasdatetimenumber_endbout - sasdatetimenumber;
  basicdate = datepart(sasdatetimenumber); noondate = datepart(sasdatetimenumber + 43200); *dates at beginning of bout for day and for noon to noon window;
  format sasdatetimenumber sasdatetimenumber_endbout datetime. basicdate noondate ddmmyy10.;
run;

*PART 2: The algorithm steps;*
proc sort data =events; by id basicdate boutid; run;

*identifying initial sleep: longest bout >= &longestatleast (e.g. 2h) that occurred in each 24-hr window (defined noon-noon) and any bouts >= &vlongbout (e.g. 5h);
proc means data =events noprint; by id noondate; output out = noondate max(Interval_s)=maxboutdur_nd; run;
data events; merge events noondate; by id noondate; if Interval_s = maxboutdur_nd & (Interval_s/(60*60)) >= &longestatleast then longestbout = 1;
sleepbout = 0; if longestbout = 1 then sleepbout = 1; if (Interval_s/(60*60)) ge &longestatleast then sleepbout = 1;
*initial sleep yes/no;
*calculating the various times in each activity for summation (to identify probable nonwear days) and for searching around sleep;
sitting_s = 0; standing_s = 0; stepping_s = 0; active_s = 0; nonstep_s = 0; prnonstep2h_s = 0; prnonstep30m_s = 0;
if Activity_code = 0 then sitting_s = Interval_s; if Activity_code = 1 then standing_s = Interval_s; if Activity_code = 2 then stepping_s = Interval_s;
if Activity_code in(1,2) then active_s = Interval_s; if Activity_code in(0,1) then nonstep_s = Interval_s;
if Activity_code in(0,1) & (Interval_s/(60*60)) ge &longbout then prnonstep2h_s = Interval_s;
if Activity_code in(0,1) & (Interval_s/(60*60)) ge &mediumbout then prnonstep30m_s = Interval_s;
drop _type_ _freq_; run;

*numbering the periods between one initial sleep bout and the next;
data events; set events; by id;
if first.id then slnum = 0; if sleepbout = 1 then slnum + 1; *periods forwards;
sleepboutall = 0; if sleepbout = 1 then sleepboutall = sleepbout; *sleepboutall refers to all sleep including newly found;
slnumR = -1*(slnum); slnumR_i = lag(slnumR); if sleepbout = 1 then slnumR =slnumR_i ; drop slnumR_i; *periods backwards;
run;

*ALGORITHM STEP 2 -> SEARCHING AROUND INITIAL SLEEP, AFTER (FORWARDS) AND BEFORE (BACKWARDS);
*this version starts at the first bout, continues whether more sleep is not found or not, then later the windows are limited to the relevant periods
- only more sleep following each previously identified sleep bout until more sleep is not found (once);
*FORWARDS;
data psp; set events; by id boutid ;
if first.id then ps_num=0; *post sleep window period number;

if first.id or moresleep or notmoresleep then do;
start=0; *indicator for starting to count the sleep period;
moresleep = 0; *indicator for stopping the window - and it was sleep;
notmoresleep = 0; *indicator for stopping the window - and it was not more sleep;
cnt_steps=0; *number of steps in window;
dur_step_s=0; *duration of stepping in window;
dur_nonstep_s=0; *duration of sitting or standing in window;
dur_prnonstep2h_s = 0; *duration of 2+ hr bouts of sitting or standing in window;
dur_prnonstep30m_s = 0; *duration of 30+ min bouts of sitting or standing in window;
n_sleepbouts = 0; *number of previously identified sleep bouts found in window;
strt_ps_num = boutid; *bout number at start of window;
end_ps_num = boutid; *bout number at end of window;
strt_ps_sb = sasdatetimenum ; *start of window, bout start time;
end_ps_sb = sasdatetimenum; *start of window, bout end time;
strt_ps_eb = sasdatetimenum_endbout; *end of window, bout start time;
end_ps_eb = sasdatetimenum_endbout; *end of window, bout end time;
s_window = 0; * window duration [after sleep/last window to beginning of bout at end of this window];

retain ps_num start moresleep notmoresleep strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb s_window cnt_steps
dur_step_s dur_nonstep_s dur_prnonstep2h_s dur_prnonstep30m_s n_sleepbouts strt_ps_num end_ps_num boutid;

if start = 0 then do;
   strt_ps_eb=sasdatetimenum_endbout; strt_ps_sb=sasdatetimenum; strt_ps_num = boutid;
   start=1;
end;

cnt_steps + steps_n; dur_step_s + stepping_s; dur_sit_s + sitting_s; dur_nonstep_s + nonstep_s;

dur_prnonstep2h_s + prnonstep2h_s ; dur_prnonstep30m_s + prnonstep30m_s ;

if sleepboutall = 1 then n_sleepbouts = n_sleepbouts + 1;
s_window = end_ps_sb - strt_ps_eb;

if ((s_window/60) <= &sleepwindow & (n_sleepbouts > 0 | dur_prnonstep2h_s > 0)) |
( (s_window/60) <= &sleepwindow & (dur_prnonstep30m_s > 0 | n_sleepbouts > 0))|
( (s_window/60) >= &sleepwindow) | last.id then do;
   * found condition for more sleep;
   if ((s_window/60) <= &sleepwindow & (n_sleepbouts > 0 | dur_prnonstep2h_s > 0)) |
((s_window/60) <= &sleepwindow & (dur_prnonstep30m_s > 0 | n_sleepbouts > 0)) & cnt_steps < &steptolerance ) |
cnt_steps = 0
then moresleep=1;
   *met no conditions for finding more sleep;
moresleep=0; notmoresleep = 1- moresleep;
ps_num=ps_num+1;  /*number the potential sleep period;
*output one record for each potential sleep period;
keep id ps_num strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb s_window moresleep notmoresleep cnt_steps dur_step_s
dur_nonstep_s dur_prnonstep2h_s dur_prnonstep30m_s n_sleepbouts strt_ps_num end_ps_num ;
output;
end;
format strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb datetime.;
run;

*decide whether each window is extra sleep or not -> only extra sleep if found more sleep but not if searching forwards
before found any sleep at all;
data psp;set psp; by id; if first.id then periodnum = 0; if lag(n_sleepbouts) > 0 then periodnum+1; if notmoresleep = 1
then stopnum1 = ps_num; run;
proc means data = psp noprint
by id periodnum; where notmoresleep= 1; output out=stops min(stopnum1)=firststop1; run;
data psp; merge psp stops; by id periodnum; extrasleep1 = moresleep; if periodnum = 0 then extrasleep1 = 0; if ps_num
ge firststop1 & firststop1 ne . then extrasleep1 = 0; run;
data psp1; set psp; if periodnum ne 0 & extrasleep1 = 1; run;

*list out all the new sleep bouts found searching forwards and mark those bouts as yes to sleepboutall;
data psplist(keep=id boutid sleepboutF);   set psp1;  by id ps_num;
do i=strt_ps_num to end_ps_num by 1; boutid=i; sleepboutF=1; output; end;
run;
data events; merge events (in = in_events) psplist; by id boutid; if in_events; if (sleepboutF = 1 | sleepbout = 1)
then sleepboutall = 1; run;

* LARGELY THE SAME, BACKWARDS;
proc sort data= events; by id boutidR; run;
data pspR; set events; by id boutidR ; *DIFFERENT BACKWARDS;
if first.id then ps_num=0;

if first.id or moresleep or notmoresleep then do;
    start=0;  moresleep = 0; notmoresleep = 0;  /*DIFFERENT BACKWARDS;
cnt_steps=0; dur_step_s=0; dur_nonstep_s=0;
dur_prnonstep2h_s = 0;  dur_prnonstep30m_s = 0; n_sleepbouts = 0;
strt_ps_num = boutidR;  *DIFFERENT BACKWARDS;
end_ps_num = boutidR;  *DIFFERENT BACKWARDS;
strt_ps_sb = sasdatetimename;  end_ps_sb = sasdatetimename;  strt_ps_eb = sasdatetimename_endbout;
end_ps_eb = sasdatetimename_endbout;  s_window = 0;
end;

retain ps_num start moresleep notmoresleep strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb s_window cnt_steps dur_step_s
dur_nonstep_s dur_prnonstep2h_s dur_prnonstep30m_s n_sleepbouts strt_ps_num end_ps_num boutid ;
if start = 0 then do; strt_ps_eb=sasdatetimename_endbout;  strt_ps_sb=sasdatetimename;  strt_ps_num = boutidR;
start=1; end;  *DIFFERENT BACKWARDS;
end_ps_sb=sasdatetimenumber; end_ps_eb=sasdatetimenumber_endbout; end_ps_num=boutidR; *DIFFERENT BACKWARDS;

cnt_steps + steps_n; dur_step_s + stepping_s; dur_sit_s + sitting_s; dur_nonstep_s + nonstep_s;
dur_prnonstep2h_s + prnonstep2h_s; dur_prnonstep30m_s + prnonstep30m_s;
if sleepboutall=1 then n_sleepbouts = n_sleepbouts + 1;

s_window = (strt_ps_sb - end_ps_eb); *DIFFERENT BACKWARDS: time since beginning of period, with window running backwards;

if ((s_window/60) <= &sleepwindow & (n_sleepbouts > 0 | dur_prnonstep2h_s > 0)) |
((s_window/60) <= &sleepwindow & (dur_prnonstep30m_s > 0 | n_sleepbouts > 0)) |
((s_window/60) >= &sleepwindow)
then do;

if ((s_window/60) <= &sleepwindow & (n_sleepbouts > 0 | dur_prnonstep2h_s > 0)) |
((s_window/60) <= &sleepwindow & (dur_prnonstep30m_s > 0 | n_sleepbouts > 0) & cnt_steps < &steptolerance) |
cnt_steps = 0
then moresleep=1;
else moresleep=0; notmoresleep = 1 - moresleep;

ps_num=ps_num+1;

keep id ps_num strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb s_window moresleep notmoresleep cnt_steps
dur_step_s dur_nonstep_s dur_prnonstep2h_s dur_prnonstep30m_s n_sleepbouts strt_ps_num end_ps_num ;
output;
end;
format strt_ps_eb strt_ps_sb end_ps_eb end_ps_sb datetime.;
run;

*decide whether each window is extra sleep or not -> only extra sleep if found following (backwards) from initial sleep or moresleep without first encountering a failure to find moresleep;
data pspR; set pspR; by id;if first.id then periodnumR = 0; if lag(n_sleepbouts) > 0 then periodnumR+1; if notmoresleep = 1 then stopnum1 = ps_num; run;
proc means data = pspR noprint; by id periodnumR; output out=stopsR min(stopnum1)=firststop1 ; run;
data pspR; merge pspR stopsR; by id periodnumR; extrasleep1 = moresleep; if periodnumR = 0 then extrasleep1 = 0; if ps_num ge firststop1 & firststop1 ne . then extrasleep1 = 0; run;
data psplstR; set pspR; if periodnumR ne 0 & extrasleep1 = 1; run;
data psplstR(keep=id boutidR sleepboutB); set psplstR; by id ps_num;
  do i=strt_ps_num to end_ps_num by 1; boutidR=i; sleepboutB=1; output; end;
run;

proc sort data = psplstR; by id boutidR; run;
proc sort data = events; by id boutidR; run;
data events; merge events (in = in_events) psplstR; by id boutidR; if in_events;
if (sleepboutF = 1 | sleepboutB = 1 | sleepbout = 1) then sleepboutall = 1;
run;

proc sort data = events; by id basicdate boutid; run;

*daily summary to determine valid days -> durations in mins, and values of 0 set to 0 not missing, and percentages of waking wear;
data events; set events;
  *time in each activity during waking wear;
  AW_sitting_t = 0; AW_standing_t = 0; AW_stepping_t = 0; AW_active_t = 0; AW_all_t = 0; AW_steps_n = 0;
  if sleepboutall ne 1 then do; AW_sitting_t = sitting_s; AW_standing_t = sitting_s; AW_stepping_t = stepping_s;
  AW_active_t = active_s; AW_all_t = Interval_s; AW_steps_n = steps_n; end;
  *time in each activity during sleep or nonwear;
  SL_sitting_t = 0; SL_standing_t = 0; SL_stepping_t = 0; SL_active_t = 0; SL_all_t = 0; SL_steps_n = 0;
  if sleepboutall = 1 then do; SL_sitting_t = sitting_s; SL_standing_t = sitting_s; SL_stepping_t = stepping_s;
  SL_active_t = active_s; SL_all_t = Interval_s; SL_steps_n = steps_n; end;
run;

proc means data = events noprint; by id basicdate; output out = summary
  sum(AW_sitting_t)=AW_sitting_t_TOT sum(AW_standing_t)=AW_standing_t_TOT sum(AW_stepping_t)=AW_stepping_t_TOT
  sum(AW_active_t)=AW_active_t_TOT sum(AW_all_t)=AW_all_t_TOT sum(AW_steps_n)=AW_steps_n;
run;

data summary; set summary; drop _freq_ _type_; array durs SL_sitting_t_TOT SL_standing_t_TOT SL_stepping_t_TOT SL_active_t_TOT SL_all_t_TOT AW_sitting_t_TOT
AW_all_t_TOT AW_stepping_t_TOT AW_active_t_TOT AW_all_t_TOT do over durs; if durs = . then durs = 0; durs = durs / 60; end;
  if AW_all_t_TOT > 0 then do;
    AW_sitting_t_TOTPCT =100*(AW_sitting_t_TOT / AW_all_t_TOT);  AW_standing_t_TOTPCT =100*(AW_standing_t_TOT / AW_all_t_TOT); 
    AW_stepping_t_TOTPCT =100*(AW_stepping_t_TOT / AW_all_t_TOT); 
    AW_active_t_TOTPCT =100*(AW_active_t_TOT / AW_all_t_TOT); 
  end;
array oth SL_steps_n AW_steps_n AW_sitting_t_TOTPCT AW_standing_t_TOTPCT AW_stepping_t_TOTPCT; 
do over oth; if oth = . then oth = 0; end;
  *indicate whether day was invalid wear day;
  validday = 1;
  if AW_all_t_TOT < 60*minhrsvalid then validday = 0; *not valid if <= &minhrsvalid waking wear (e.g. 10h);
  if (AW_standing_t_TOTPCT ge &maxdaypct | AW_sitting_t_TOTPCT ge &maxdaypct | AW_stepping_t_TOTPCT ge &maxdaypct) then 
validday = 0; *not valid if >= &maxdaypct (e.g. 95%) in one activity;
  if AW_steps_n < &minstepsday then validday = 0; *not valid if  < &minstepsday (e.g. 500);
format basicdate dmmmyy10.;
run;

data events; merge events summary; by id basicdate; run;
PART 3: Outputting the data to csv for later use or quality control steps;
%if &tagfiles = 1 %then %do;
*output csv files like the events information and new sleep/nonwear and valid day classifications;
data taggedevents; set events;
APDatetimvar = (sasdatetimenumber/86400) + 21916; APDatacount = Datacount; APIInterval_s = Intervalold_s;
APActivity_code = Activity_code; APCumulativeStepCount = CumulativeStepCount; APActivityScoreMEThr = ActivityScore_METhr;
validday_est = validday; sleepbout_est = sleepboutall; participantID = &id;
keep APDatetimvar APDatacount APIInterval_s APActivity_code APCumulativeStepCount APActivityScoreMEThr validday_est sleepbout_est participantID;
keep APDatetimevar APDatacount APInterval_s APActivity_code APCumulativeStepCount APActivityScoreMEThr validday_est sleepbout_est participantID;
run;
proc export data = taggedevents outfile = "&csvname&id..csv" dbms=csv replace; putnames = yes; run;
%end;
%if &outlist = 1 %then %do;
*outputs a csv list of the sleep-nonwear periods [SLNW bouts] for that id;
data forlist (keep = id slbnum sasdatetimenumber sasdatetimenumber_endbout sleepboutall); set events; by id boutid;
if first.id then do; slbnum = 1; end;
if sleepboutall ne lag(sleepboutall) then do; slbnum + 1; end;
run;
proc means data = forlist noprint; by id slbnum; where sleepboutall = 1; output out=listsl
min(sasdatetimenumber)=slnwperiod_st max(sasdatetimenumber_endbout)=slnwperiod_end; run;
data listsl; set listsl; drop _type_ _freq_; format slnwperiod_st slnwperiod_end datatype. ; drop slbnum; run;
proc export data= listsl outfile= "&listname&id..csv" dbms=csv replace; putnames=yes; run;
%end;
%if &keeplist = 1 %then %do;
*outputs a csv list of all the periods of data to keep as valid waking wear [not SLNW bouts, not invalid days] for that id;
data forkplist (keep = id valnum sasdatetimenumber sasdatetimenumber_endbout incdata sleepboutall validday); set events; by id boutid;
incdata = 1; if sleepboutall = 1 | validday ne 1 then incdata = 0;
if first.id then do; valnum = 1; end;
if incdata ne lag(incdata) then do; valnum + 1; end;
run;
proc means data = forkplist noprint; by id valnum; where incdata = 1; output out=listval
min(sasdatetimenumber)=validperiod_st max(sasdatetimenumber_endbout)=validperiod_end; run;
data listval; set listval; drop _type_ _freq_; format validperiod_st validperiod_end datatype. ; drop valnum; run;
proc export data= listval outfile= "&keepname&id..csv" dbms=csv replace; putnames=yes; run;
%end;
%end;

PART 4: Outputting plots (heatmaps) for quality control;
%if &printheatmaps = 1 %then %do;
*plot heatmap of the maximum activity per timeperiod >=1s (e.g., &hmblock= 60 for per min, or 120 per 2 min or 1 for 1s);

data ev_sr (keep = id boutid Activity_code s_start s_end sleepbout sleepboutall validday); set events;
s_start =  floor(sasdatetimeno) ; s_end =  floor(sasdatetimeno_endbout); /* identifying the second each bout occurred in;
run;

proc sort data = ev_sr; by id boutid; run;
data ev_sr_epoch1 ; set  ev_sr; by id boutid ; do i=s_start to s_end by &hmblock; t_s=i; output; end; run;
proc sort data = ev_sr_epoch1; by id t_s; run;
proc means data = ev_sr_epoch1 noprint; by id t_s; output out = ev_sr_epoch max(Activity_code)=Activity_code max(sleepboutall)=sleepboutall max(validday)=validday; run;
data ev_sr_epoch; set ev_sr_epoch;
if sleepboutall ne 1 & validday = 1 then AW_Activity_code = Activity_code; if (sleepboutall = 1 | validday ne 1) then SLNW_Activity_code = Activity_code; Weardate1= datepart(t_s); timel = timepart(t_s); format Weardatel weekdate15. timel time8.; length wdtext3 $20; wdtext3 = put(Weardatel, WEEKDATE17.);
run;
*heatmap of included data, i.e., waking wear on valid days;
proc template; define style mystyle; parent = styles.statistical; class ThreeColorRamp / endcolor = green neutralcolor = yellow startcolor =red; end;
run;
ods html body='ODS.HTML' style=mystyle GPath="&incmapsout";
proc template; define statgraph heatmapparm; begingraph; layout overlay;
    heatmapparm x=wdtext3 y=timel colorresponse=AW_Activity_code/ name="AW_&id" xbinaxis=false ybinaxis=false yvalues=leftpoints xgap=5 colormodel=ThreeColorRamp includemissingcolor=false;
endlayout; endgraph; end;
run;
ods graphics / imagename="AW_&id";
proc sgrender data=ev_sr_epoch template=heatmapparm; where id = &id; label timel="24 Hour Time" wdtext3="Assessment date"; run;

*heatmap of excluded data -> waking wear on valid days;
ods html body='ODS.HTML' style=mystyle GPath="&excmapsout";
proc template; define statgraph heatmapparm; begingraph; layout overlay;
    heatmapparm x=wdtext3 y=timel colorresponse=SLNW_Activity_code/ name="SL_&id" xbinaxis=false ybinaxis=false xgap=5 yvalues=leftpoints colormodel=ThreeColorRamp;
endlayout; endgraph; end;
run;
ods graphics / imagename="SL_&id";
proc sgrender data=ev_sr_epoch template=heatmapparm; where id = &id; label timel="24 Hour Time" wdtext3="Assessment date"; run;
%end;
*delete datasets and html output before running for next file;
proc datasets lib=work nolist kill; quit; run;
ods html close;
%mend algvb;

*key algorithm decisions and thresholds;
%let vlongbout = 5; %let longestatleast = 2; %let sleepwindow = 15; %let longbout = 2; %let mediumbout = 0.5; %let steptolerance = 21; *re: sleep / non-wear bouts;
%let maxdaypct = 95; %let minstpsday = 500; %let minhrsvalid = 10; *re: valid days;

*indicate desired functions: 1= yes, perform, 0= no do not perform;
%let printheatmaps = 1; %let tagfiles = 1; %let keeplist = 1; %let outlist = 1;
%let hmblock = 120; *heatmap resolution maximum activity per two minute (120 second) time block for optional quality control step;

*nominate folder to find the events files, followed by \, such as;
%let dir = C:\Users\yeahme\Documents\mystudy\activpal\downloaded\;

*nominate folder to save heatmaps of included and excluded data;
%let incmapsout = C:\Users\yeahme\Documents\mystudy\activpal\processed\heatmaps\include ;
%let excmapsout = C:\Users\yeahme\Documents\mystudy\activpal\processed\heatmaps\exclude ;

*nominate folder and beginning of filename for output csv files;
*events like files with the sleep/non-wear (yes/no) and valid day (yes/no) classifications;
%let csvname = C:\Users\yeahme\Documents\mystudy\activpal\processed\taggedfiles\taggedevents_ ;
*lists of continuous periods identified as sleep/non-wear bouts;
%let listname = C:\Users\yeahme\Documents\mystudy\activpal\processed\taggedfiles\slnwlist_ ;
*lists of continuous periods identified as waking wear on valid days;
%let keepname = C:\Users\yeahme\Documents\mystudy\activpal\processed\taggedfiles\validperiods_ ;

***use macro execution statements such as the following to run the program for these files;
%algvb(fname=0001-AP1131854 10Sep11 12-00am  for 13d 11h 7m Events,id=0001);
%algvb(fname=0003-AP1131849 10Sep11 12-00am  for 13d 11h 20m Events,id=0003);