```
fprintf('STARTING ANALYSIS OF Track %.0f.wav', ii)
[~,Fs] = audioread(path, [1,2]);
mem = memlength*60*60*Fs;

%This mess breaks up the file automatically into manageable chunks just to
%determine lengthx and not run out of memory

if roundhours<= 10
    [x,Fs] = audioread(path);
        lengthx = length(x);
elseif roundhours<=20
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
        clear x
    [x2,Fs] = audioread(path, [mem+1,inf]);
        lengthx = lengthx + length(x2);
        clear x2
elseif roundhours<=30
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
        clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,Fs] = audioread(path, [2*mem+1,inf]);
        lengthx = lengthx + length(x3);
        clear x3
elseif roundhours<=40
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
            clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,~] = audioread(path, [2*mem+1,3*mem]);
        lengthx = lengthx + length(x3);
        clear x3
    [x4,Fs] = audioread(path, [3*mem+1,inf]);
        lengthx = lengthx + length(x4);
        clear x4
elseif roundhours<=50
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
            clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,~] = audioread(path, [2*mem+1,3*mem]);
        lengthx = lengthx + length(x3);
        clear x3
    [x4,~] = audioread(path, [3*mem+1,4*mem]);
        lengthx = lengthx + length(x4);
        clear x4
    [x5,Fs] = audioread(path, [4*mem+1,inf]);
        lengthx = lengthx + length(x5);
        clear x5
elseif roundhours<=60
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
            clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,~] = audioread(path, [2*mem+1,3*mem]);
        lengthx = lengthx + length(x3);
        clear x3
    [x4,~] = audioread(path, [3*mem+1,4*mem]);
        lengthx = lengthx + length(x4);
        clear x4
    [x5,~] = audioread(path, [4*mem+1,5*mem]);
        lengthx = lengthx + length(x5);
        clear x5
    [x6,Fs] = audioread(path, [5*mem+1,inf]);
        lengthx = lengthx + length(x6);
        clear x6
elseif roundhours<=70
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
            clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,~] = audioread(path, [2*mem+1,3*mem]);
        lengthx = lengthx + length(x3);
        clear x3
    [x4,~] = audioread(path, [3*mem+1,4*mem]);
        lengthx = lengthx + length(x4);
        clear x4
    [x5,~] = audioread(path, [4*mem+1,5*mem]);
        lengthx = lengthx + length(x5);
        clear x5
    [x6,~] = audioread(path, [5*mem+1,6*mem]);
        lengthx = lengthx + length(x6);
        clear x6
    [x7,Fs] = audioread(path, [6*mem+1,inf]);
        lengthx = lengthx + length(x7);
        clear x7
elseif roundhours<=80
    [x,~] = audioread(path, [1, mem]);
        lengthx = length(x);
            clear x
    [x2,~] = audioread(path, [mem+1,2*mem]);
        lengthx = lengthx + length(x2);
        clear x2
    [x3,~] = audioread(path, [2*mem+1,3*mem]);
        lengthx = lengthx + length(x3);
        clear x3
    [x4,~] = audioread(path, [3*mem+1,4*mem]);
        lengthx = lengthx + length(x4);
        clear x4
    [x5,~] = audioread(path, [4*mem+1,5*mem]);
        lengthx = lengthx + length(x5);
        clear x5
    [x6,~] = audioread(path, [5*mem+1,6*mem]);
        lengthx = lengthx + length(x6);
        clear x6
    [x7,~] = audioread(path, [6*mem+1,7*mem]);
        lengthx = lengthx + length(x7);
        clear x7
    [x8,Fs] = audioread(path, [7*mem+1,inf]);
        lengthx = lengthx + length(x8);
        clear x8
end

% This if statement calculates the length of the end of the recording that
% extends beyond the last 30 minute chunk:
if (0<minute) && (minute<30)
    endtime = rem((lengthx-(30-minute)*60*Fs),30*Fs*60);
elseif (30<minute) && (minute<=59)
    endtime = rem((lengthx-(60-minute)*60*Fs),30*Fs*60);
else
    endtime = rem(lengthx,30*Fs*60);
end
%% While loop to evaluate every 30 minutes of recording

datablock = [];
k=0;                    %lead indexing value
q = 1;                  %indexing value needed for plotting U (flight activity per
30 minutes of recording
U = [];                 %set up U to contain number of flight bouts (counter) per 30
minute chunk
allnewz = [];           %set up allnewz to contain all concatenated newz arrays
(which becomes the signal containing only flight)
lastrun = 0;            %set up variable that changes when the loop is on its last
run-through (for triggering the code to state an ending time)
iteration = 1;
while k< lengthx
fprintf('\nTime: %02d:%02d', [hour, minute])   %state the time at the start of each
30 minute chunk
  if k == lengthx - endtime                    %case where there is not a full 30
```

```
minutes left in the recording, only endtime(quantity) samples remain
        sam = endtime;                         %sam is the number of samples the
loop advances by every cycle (here it is set to the exact number of samples
remaining in the signal)
        start = k;                             %the starting point for audioread
command is defined by k, which tracks how far along in x the program is
        lastrun = 1;                           %changes lastrun value to 1, because
it is the final run-through of the loop
  else                                %if the loop is not about to end:
        if 0<minute && minute<30               %if the starting minute of the
recording is between 0 and 30 minutes
            sam = (30-minute)*60*Fs;           %set sam (how much to advance) to the
number of samples (amount of time) between the starting minute and the next 30
minute starting point
            start = 1;                         %because this only happens in the
first iteration, the code will start with the first value in x
        elseif 30<minute && minute<=59         %if starting minute between 30 and
the next hour
            sam = (60-minute)*60*Fs;           %set sam to advance the difference to
the next hour
            start = 1;
            if hour == 23                      %make 24-hr time loop back to 00:00
instead of 24:00
            hour = 0;
            else
            hour = hour+1;                     %advance hour
            end
        else
            sam = 30*60*Fs;                    %if starting minute is already on the
hour or half hour
            if k==0                            %k begins as zero, but must become 1
to be used
            start = 1;
            else
            start = k;                         %sets start point to k. This should
be the case for all but the first and last iteration
            end
        end

  end

    [c,Fsc] = audioread([path2folder filename num2str(0) num2str(8) '.wav'],
[start,start+sam]); %get c, the signal for vial 8, which records only the sound of
the room
    [x,Fs] = audioread(path, [start,start+sam]); %get x, the signal, in a chunk size
defined by the variable sam

    N = length(x);                      % signal length of x
    t = (0:N-1)/Fs;                   % time vector (useful for plotting)
    ratio = median(abs(x))/median(abs(c)); %differences in sensitivity could make x
or c levels of silence uneven. This attempts to equalize the levels of silence in
the two recordings via creating a ratio of median values
    c = c*ratio;                            %c is then multiplied by this ratio

    fracsec = .01;              % small step of time which will be analyzed
    step = fracsec*Fs;          % convert fracsec to no. of samples
    stdevc = std(c);            % standard deviation of track 8 background sound 30
minute recording

    z = x;                      %set up duplicate array to modify

    i = 1;                      %indexing value

        %this loop attempts to distinguish flight from ambient noise by
        %comparing signal standard deviations

    while i<N-step
        if std(x(i:i+step-1))< 6*std(c(i:i+step-1)) %also could use <stdevc   %if
the standard deviation of a step in x is less than 6x the std of a step
            z(i:i+step-1) = 0;                      %then z(i) becomes zero and will
not be counted as flight
        else
            z(i:i+step-1) = 1;                      %otherwise, it will be
        end
        i = i+step;
    end

    newz = x(z>0);                         %newz contains only values of x deemed to be
flight (ones in z)

    tt = N/Fs;                            %total recorded time in chunk
    ft = length(newz)/Fs ;                %time spent in flight during recorded chunk
    percentfly = ft/tt*100;               %calc percent of time flying during recorded
chunk

    fprintf('\n\nThe flies flew for a total of %.2f seconds,', ft)     %these two
display the findings via text
    fprintf('\napproximately %.2f%% of the recorded time, \n', percentfly)

    sf = 4; %sf determines how many 0.01 second intervals, constitute a bout of
flight:
            %This is arbitrary. a value of 4 means that all
            %flight bouts greater than 0.03 seconds will be
            %counted
    i=1;                    %indexing variable
    counter=0;         %placeholder for variable that counts number of flight bouts
    lengths = [];      %placeholder for array of flight bout durations
    kk = 1;                 %indexing variable
    spaces = [];       %placeholder for array of spaces between flight bouts
    while i<(N-sf*step)             %if i is less than the length of x minus the sf
number of steps
        if z(i:i+sf*step-1)==1      %if the next sf steps are flight (as determined
by z)
           nextstart = i;           %saves this start point of a new bout
            if kk ~= 1
                spaces(kk-1) = (nextstart-previousend)/Fs;   %calculate space
between bouts
            else
            end

            j=(i+sf*step);      %set j to the start of the step folowing the already
determined bout
            clear d
            while j<=(N-step)
                if z(j:j+step-1)==0     %if the step does not contain flight
                    d=j;                %save this endpoint of the bout
                    j=N;                %break the loop
                else

                    j = j+step;         %otherwise, advance by one step and run
through checking again
                end
            end

            truth = exist('d');         %check if d exists (it doesn't if a flight
bout never ends, like at the end of a 30minute chunk)
            if truth == 0
                d = N;                  %set d as the last point in the 30 mintue
chunk
            end

            lengths(kk) = d-i;          %d-1 is the length of the bout. add this
value to array lengths()
            kk = kk+1;
            counter = counter+1;        %count that a flight bout has occurred
            i = d;                      %begin i at the end point of the bout
            previousend = i;            %save this for calculating spacing between
bouts
        else
            i = i+step;                 %if no bout detected, advance one step
        end
    end

    avglength = mean(lengths);
    avglengthsec = avglength/Fs;       %average length in seconds
    avgspace = mean(spaces);
    fprintf('with %.0f flight bouts longer than %.2f seconds \n', [counter,
((sf-1)*step)/Fs])
```

```
    fprintf(',an average flight bout duration of %.5f seconds \n', avglengthsec)
    fprintf('and an average time between flight bouts of %.5f seconds \n', avgspace)
    %displays this information

    U(q) = counter;             %plots flight activity per 30 mintues in flight
bouts per half hour

    allnewz = [allnewz; newz];      %concatenation

    if hour == 19 && minute == 0     %to look at what is being counted as flight
graphically, enter time here and run program to see plot
        figure(q)
        plot(t,x)
        hold on
        plot(t,x.*z)
        axis([0 t(end) -1.1*max(abs(x)) 1.1*max(abs(x))])
        hold off
    end

    k = k + sam;
    minute = minute + sam/(Fs*60);          %advance minute and sam for next
iteration
    if minute == 60
        minute = 0;
        if hour == 23
            hour = 0;
        else
            hour = hour + 1;
        end
    end
    q = q + 1;

    datablock(:,iteration) = [filenumber; hour; minute; ft; counter; avglengthsec;
avgspace];

    if lastrun == 1
        fprintf('\nTime: %02d:%02d\n', [hour, round(minute)])    %display final time
at the end of the loop
    end
    iteration = iteration + 1;
end

filename = 'allsoundz.wav';                 %this writes the concatenated allnewz
audiowrite(filename,allnewz,Fs);            %to a wav file that can be analyzed by
freqdetect.m

plot(0:.5:.5*q-1, U);       %plots flight activity in bouts per 30 minutes over the
duration of the recording

freq_detect              %make sure freqdetect.m is in the same folder as
```

flight_detect.m

```
datablock(end+1,1) = loc(1);
datablock(end+1,1) = loc(2);
```