```
%This version of the script: (Jan 11 2018)
%Incorporates coral & non-coral (range-type) uncertainties
%Treats replicates as single point data
%Handles both C14 and U/Th dated samples
%No internal screening on d234U error - prescreening required

clear all
close all

NoTimeInstances=10000;
% one RSL generated for each generated age
NoRSLInstances=1;

lowerAge=0;
upperAge=100;

dataPath=('/Location_Of_Sample_Data/');
cd(dataPath);

%Import data extract from db that has a header line
%Data comes in as format: 'coralID',LocationID','uplift','uplift error',
%'Elevation Used','Elevation Used Error','CoralType','CoralErrorType',
%'ErrorRange1','ErrorRange2','Age','AgeError','RepGroup'
%careful of error being pulled in - stats use 1 sigma.
%LocationID and RepGroup aren't used in this version
CoralInPointData = csvread('DBextract2018.csv',1,0);

%Filter it by relevant ages
CoralInPointData=CoralInPointData(CoralInPointData(:,11)>=lowerAge & ...
    CoralInPointData(:,11)<=upperAge,:);

%Generate a list of OXCAL distributions (Accessing pre-processed data)
oxcalSampleList=dir('/OxCalDistributionLocationPath/*.csv');
for x=1:size(oxcalSampleList,1);
    oxcalSamplesCell(x)={oxcalSampleList(x).name(1:end-4)};
    A=oxcalSamplesCell{x};
    B=str2num(A);
    oxcalSamples(x,1)=B;
    clear A B
end

%END SETUP*************************************************************

%Here we treat RANGE TYPE errors   ************************************
NonCoralErrorData=CoralInPointData(CoralInPointData(:,8)==1,:);
NumberNonCoralData=size(NonCoralErrorData,1);

if NumberNonCoralData>0;
   RangeTable=zeros(NumberNonCoralData,10,NoTimeInstances,NoRSLInstances);

for nonCoralLoop=1:NumberNonCoralData
    %every data point has its own error distribution depending
```

```
%on input range handling to find lower of two range entries
minRange=min(NonCoralErrorData(nonCoralLoop,9),...
    NonCoralErrorData(nonCoralLoop,10));
maxRange=max(NonCoralErrorData(nonCoralLoop,9),...
    NonCoralErrorData(nonCoralLoop,10));

rslErrorNumbers=unifrnd(minRange,maxRange,NoTimeInstances,...
    NoRSLInstances);

if ismember(NonCoralErrorData(nonCoralLoop,1),oxcalSamples);
    %i.e. if they are a sample with an oxcal distribution...
    timeError=oxcalDistribute_NOGIA...
        (NonCoralErrorData(nonCoralLoop,1),NoTimeInstances);
else
    %U/Th dated sample
    timeError=normrnd(NonCoralErrorData(nonCoralLoop,11),...
        NonCoralErrorData(nonCoralLoop,12)/2,NoTimeInstances,1);
end

timeInstance=1;
while timeInstance<=NoTimeInstances;
        %each time instance will spawn a simulated RSL instance
        RSLInstance=1;

        %each time instance results in a different uplift and elevation
        %here we perform uplift correction
        [ZedTCP,ZedTCPerror]=upliftCorrectElevation2...
            (NonCoralErrorData,timeError,nonCoralLoop,timeInstance);
        %Form our calculation matrix - 1st coral id
        RangeTable(nonCoralLoop,1,timeInstance,1:NoRSLInstances)...
            =NonCoralErrorData(nonCoralLoop,1);

        %calculated elevation
        RangeTable(nonCoralLoop,3,timeInstance,1:NoRSLInstances)=...
            ZedTCP;
        %Time instanced
        RangeTable(nonCoralLoop,4,timeInstance,1:NoRSLInstances)=...
            timeError(timeInstance,1);

        %time instanced and uplift corrected coral.********************
        RangeTable(nonCoralLoop,8,timeInstance,1:NoRSLInstances)=...
            RangeTable(nonCoralLoop,3,timeInstance,RSLInstance);
        elevationNumbers=normrnd(RangeTable(nonCoralLoop,8,...
            timeInstance,1),ZedTCPerror,NoRSLInstances,1);
        %normalise them
        elevationNumbers=elevationNumbers-RangeTable...
            (nonCoralLoop,8,timeInstance,1);

    %here we are combining the sampled elevation with depth
    %distribution, and converting it into a represetnation of sea level
    %we are combining two distributions, which we assume are gaussian,
    %we cannot assume that the output distribution is a gaussian.
        RangeTable(nonCoralLoop,9,timeInstance,RSLInstance)=...
            -1*(elevationNumbers(RSLInstance,1)+...
```

```
                rslErrorNumbers(timeInstance,RSLInstance));

        %we add the error correction to the elevation to reconstruct
        %sea level
            RangeTable(nonCoralLoop,10,timeInstance,RSLInstance)=...
                (RangeTable(nonCoralLoop,8,timeInstance,RSLInstance)+...
                RangeTable(nonCoralLoop,9,timeInstance,RSLInstance));

            timeInstance=timeInstance+1;
    end

end

TotalRowsA=NumberNonCoralData*NoTimeInstances*NoRSLInstances;
OutputA=zeros(TotalRowsA,2);
i=1;
for a = 1:NumberNonCoralData;
    for y=1:NoTimeInstances
        for x = 1:NoRSLInstances
        OutputA(i,1)=RangeTable(a,1,y,x);
        OutputA(i,2)=RangeTable(a,4,y,x);
        OutputA(i,3)=RangeTable(a,10,y,x);
        i=i+1;
        end
    end
end

else
end

%This section deals only with CORAL TYPE errors   ************************
CoralInPointData=CoralInPointData(CoralInPointData(:,8)==0,:);

if size(CoralInPointData,1)>0;

numberCorals=size(CoralInPointData,1);
uniqueTaxa=unique(CoralInPointData(:,7));
CoralTable=zeros(numberCorals,10,NoTimeInstances,NoRSLInstances);

%START MAIN CALCULATIONS*************************************************

%Loop over taxa type
    CoralCounter = 1;
for taxaLoop=1:size(uniqueTaxa,1)

        similarTaxa=CoralInPointData(CoralInPointData(:,7)==...
            uniqueTaxa(taxaLoop,1),:);
        %how many of the similar taxa
        NoTaxaCorals=size(similarTaxa,1);
        %set up our random samples
        speciesNumbers=coralSimulations(similarTaxa,taxaLoop,...
            uniqueTaxa,NoTimeInstances,NoRSLInstances,1);
        counter = 0;
```

```
for counter=1:NoTaxaCorals
    %each time instance has different uplift error contribution

    if ismember(similarTaxa(counter,1), oxcalSamples)
        timeRandom=oxcalDistribute_NOGIA(similarTaxa(counter,1),...
            NoTimeInstances);
    else
        timeRandom=normrnd(similarTaxa(counter,11),...
            similarTaxa(counter,12)/2,NoTimeInstances,1);
    end

    timeInstance=1;
    while timeInstance<=NoTimeInstances;
        %each time instance will spawn a single simulated RSL instance
        RSLInstance=1;

        %each time instance results in a different uplift and elevation
        [ZedTCP,ZedTCPerror]=upliftCorrectElevation2...
            (similarTaxa,timeRandom,counter,timeInstance);

        %Form our calculation matrix - 1st coral id
        CoralTable(CoralCounter,1,timeInstance,1:NoRSLInstances)=...
            similarTaxa(counter,1);

        %calculated elevation
        CoralTable(CoralCounter,3,timeInstance,1:NoRSLInstances)=ZedTCP;
        %Time instanced
        CoralTable(CoralCounter,4,timeInstance,1:NoRSLInstances)=...
            timeRandom(timeInstance,1);

        %just the time instanced and uplift corrected coral.*****************
        CoralTable(CoralCounter,8,timeInstance,1:NoRSLInstances)=...
            CoralTable(CoralCounter,3,timeInstance,RSLInstance);

        %Now that we have the corrected elevation, we can create a distributio
        elevationNumbers=normrnd(CoralTable(CoralCounter,8,...
            timeInstance,1),ZedTCPerror,NoRSLInstances,1);
        %normalise them
        elevationNumbers=elevationNumbers-CoralTable...
            (CoralCounter,8,timeInstance,1);

        %we combine the sampled elevation with depth distribution
        %and convert it into a representation of sea level
        CoralTable(CoralCounter,9,timeInstance,RSLInstance)=...
            -1*(elevationNumbers(RSLInstance,1)+speciesNumbers...
            (counter,timeInstance,RSLInstance));

        %we add error correction to the elevation to reconstruct sea
        %level
        CoralTable(CoralCounter,10,timeInstance,RSLInstance)=...
            (CoralTable(CoralCounter,8,timeInstance,RSLInstance)+...
            CoralTable(CoralCounter,9,timeInstance,RSLInstance));

        %this ends the time instances
```

```
            timeInstance=timeInstance+1;
        end
        CoralCounter=CoralCounter+1;
        %this ends the loop over corals with similar taxa
    end
    %this ends the loop over the different types of taxa
end

%Save this file down so don't have to re-run
TotalRows=numberCorals*NoTimeInstances*NoRSLInstances;
Output=zeros(TotalRows,2);
i=1;
for a = 1:numberCorals;
    for y=1:NoTimeInstances
        for x = 1:NoRSLInstances
        Output(i,1)=CoralTable(a,1,y,x);
        Output(i,2)=CoralTable(a,4,y,x);
        Output(i,3)=CoralTable(a,10,y,x);
        i=i+1;
        end
    end
end

else
end

if exist('OutputA','var') & exist('Output','var')
Output=[Output;OutputA];
elseif exist('OutputA','var')
    Output=OutputA;
else
end
Output(all(Output==0,2),:)=[];
%************************************************
```