# Supporting information: R scripts


## Acknowledgements

This research is supported by the PALEODEM research project. This project has received funding from the European Research Council (ERC-CoG-2015) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 683018 to J.Fernández-López de Pablo).


## Introduction

This document presents a workflow explaining how the radiocarbon data for the Atlantic facade of Iberia were transformed into various timeseries models for the paper 'Late Glacial and Early Holocene human demographic responses to climatic and environmental change in Atlantic Iberia'. The source code to the custom functions called upon in this analysis is contained in the file `src.r`, which also contains further documentation.

The radiocarbon dates are calibrated using functions taken from McLaughlin 2019 and modeled with R versions of the python scripts published by Fernandez-Lopez de Pablo et al 2019.

For computational expediency, the example analysis that follows uses a default 100 bootstrap iterations. For the analysis presented in the main paper, we used 1000 iterations for the bootstrapping (`SPDboot`) and Monte Carlo simulation (`mixdensity`) processes, and although the results are very similar, interested readers can change the below scripts using the `Nboot` parameter to the bootstrapping functions and the `N` parameter of the Monte Carlo simulations. See the file `src.r` for more details.

```
# load source code
# NB working directory must also contain the source datasets
#       contained in 'datelist.csv' and 'Palaeodiet.csv'
# A connection to INTERNET required for access to IntCal curves

source('src.r')
intcal<-read.csv(url("http://intcal.org/curves/intcal20.14c"),header=FALSE,skip=11)
marine<-read.csv(url("http://intcal.org/curves/intcal20.14c"),header=FALSE,skip=11)
colnames(intcal)<-c("calBP","14Cage","Error","Delta14C","Sigma")
colnames(marine)<-c("calBP","14Cage","Error","Delta14C","Sigma")
```

## Loading data

First we load radiocarbon database and select samples identified as passing the filtering rounds of data screening.

```
datelist<-read.csv('Datelist.csv')[,-1]
datelist<-datelist[which(datelist$Filter==1),]
```

## Defining calibration curves for marine and mixed reservoir samples

The function `addcalcurves` calculates a custom calibration curve for each sample containing marine carbon, using the local reservoir corrections contained in the input data.

```
datelist<-addcalcurves(datelist, c(1,10,11,12))
```

## Binning procedure to compensate for oversampling

The binning process follows Bevan and Crema's `rcarbon` use of hierarchical clustering to compute a dendrogram of the dissimilarities between the dates. We use here a low `h` 'height' of 30 years, so that evidence for continuous occupation at a site between successive human generations is not lost. In subsequent analysis, rather than combine the dates (which is the approach used by Bevan and Crema) we randomly select one date per phase, partly because the KDE and SPD analysis can then proceed using exactly the same input data. Discarding dates also evens out the data from each site from an archaeological-methodological perspective. The low value for `h` we use here produces similar results to larger `h` values under the summing and down-weighting approach of Bevan and Crema.

As an aside, it is possible to replace the call to `findmixmedian` in the below code with a call to `MCmix` to experiment with Monte Carlo draws from the posterior probability density functions of the calibrated dates (rather than the median date) to explore the sensitivity of this analysis to calibration artefacts. In our experiments the median date proved robust.

```
# First find median date
datelist$median<-findmixmedian(datelist[,c('BP','Std','curve')])

# identify generational bins using hierarchical clustering
# (essentially the same process as implemented by Bevan and Crema 2018)
sites<-unique(datelist$ID_Site)
datelist$timebin <- 1
for(N in 1:length(sites)) {
    sitedates<-datelist[datelist$ID_Site==sites[N],'median']
    if(length(sitedates)>1) {
      sitedateindex<-which(datelist$ID_Site==sites[N])
      dendro<-hclust(dist(sitedates))
        sitebins<-cutree(dendro, h=30)
      datelist[sitedateindex,'timebin'] <- sitebins
    }
}

# pick one date per bin
combineddl<-datelist[!duplicated(datelist[,c('ID_Site','timebin')]),]
```

## Summing dates

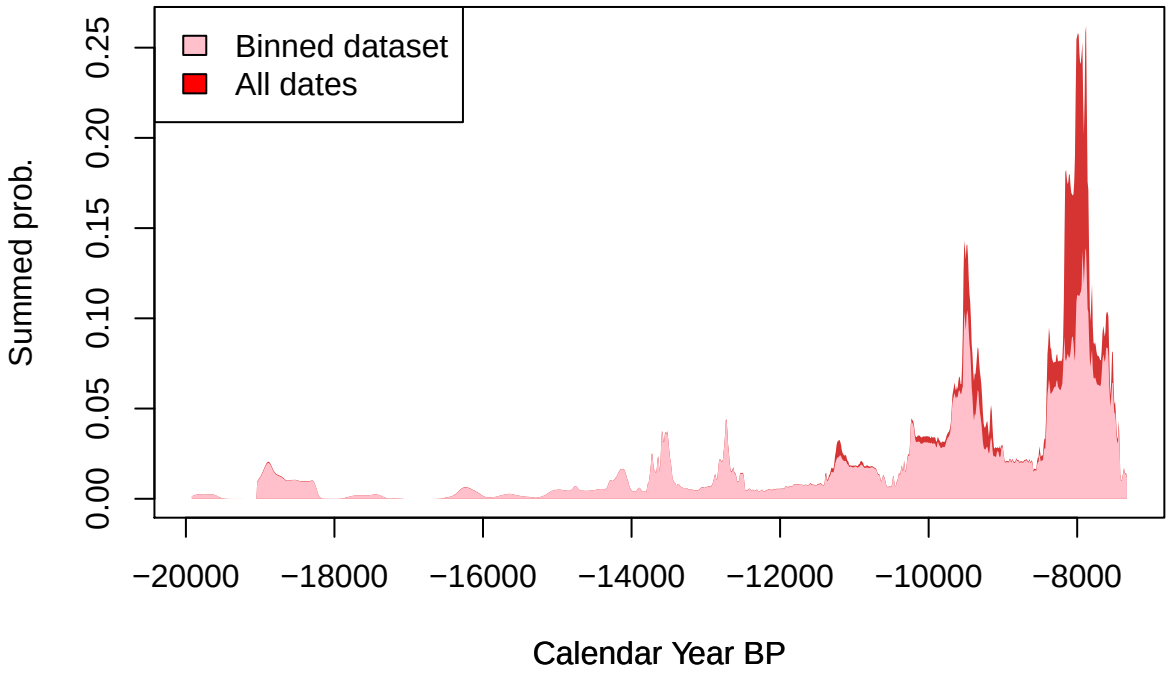
Separate SPDs are calculated for open-air and 'other' sites (i.e., caves and rock shelters), applying a taphonomic correction to those sites identified as 'Open' in the input data.

```
openspd <-rowcalsum(combineddl[
  combineddl$Type.site=='Open',c('BP','Std','curve')], norm=FALSE)
otherspd<-rowcalsum(combineddl[
  combineddl$Type.site!='Open',c('BP','Std','curve')], norm=FALSE)
spd <- taphSPD(openspd) + otherspd
```

```
# for comparison, do same for uncombined datelist
openspd_uc <-rowcalsum(datelist[
  datelist$Type.site=='Open',c('BP','Std','curve')], norm=FALSE)
otherspd_uc<-rowcalsum(datelist[
  datelist$Type.site!='Open',c('BP','Std','curve')], norm=FALSE)
spd_uc <- taphSPD(openspd_uc) + otherspd_uc
```

The binned and unbinned SPDs can be compared:

```
plot(spd_uc)
plot(spd, col='pink', add=T)
legend('topleft',fill=c('pink','red'),legend=c('Binned dataset','All dates'))
```



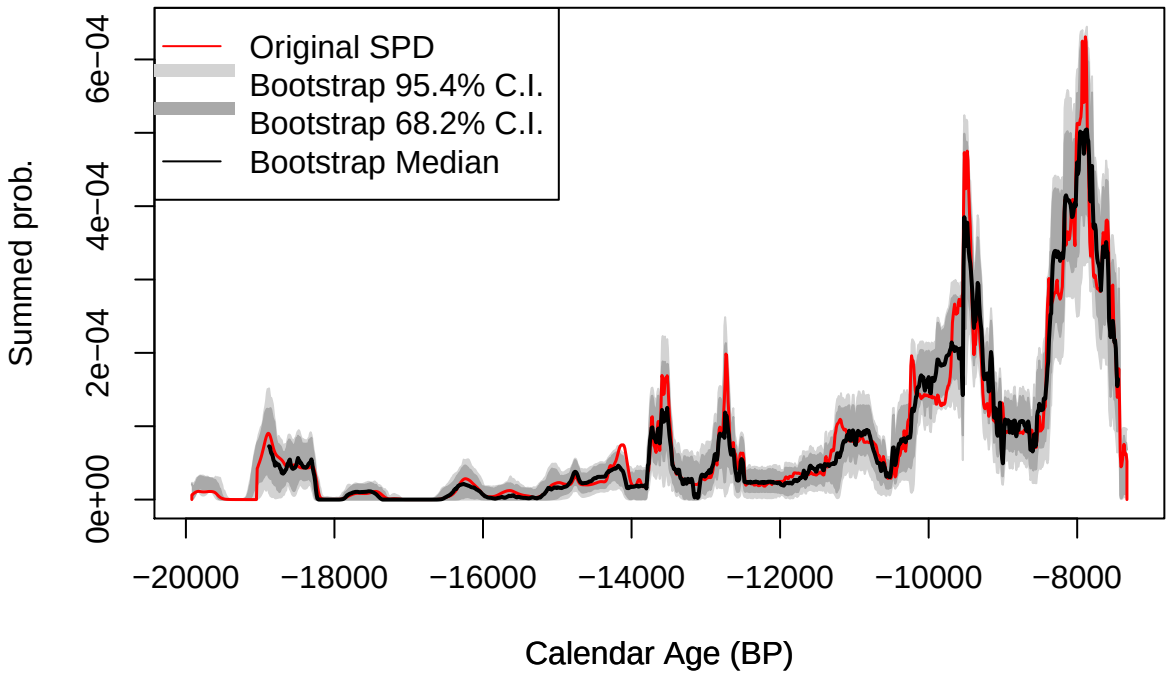

## Bootstrap confidence intervals and model fitting

The function `SPDboot` applies the method of Fernandez-Lopez de Pablo et al 2019 to the SPD.

```
spdb <- SPDboot(spd)
```

```
## ================================================================================
```

```
PDplot(boot=spdb, spd=normSPD(spd))
```

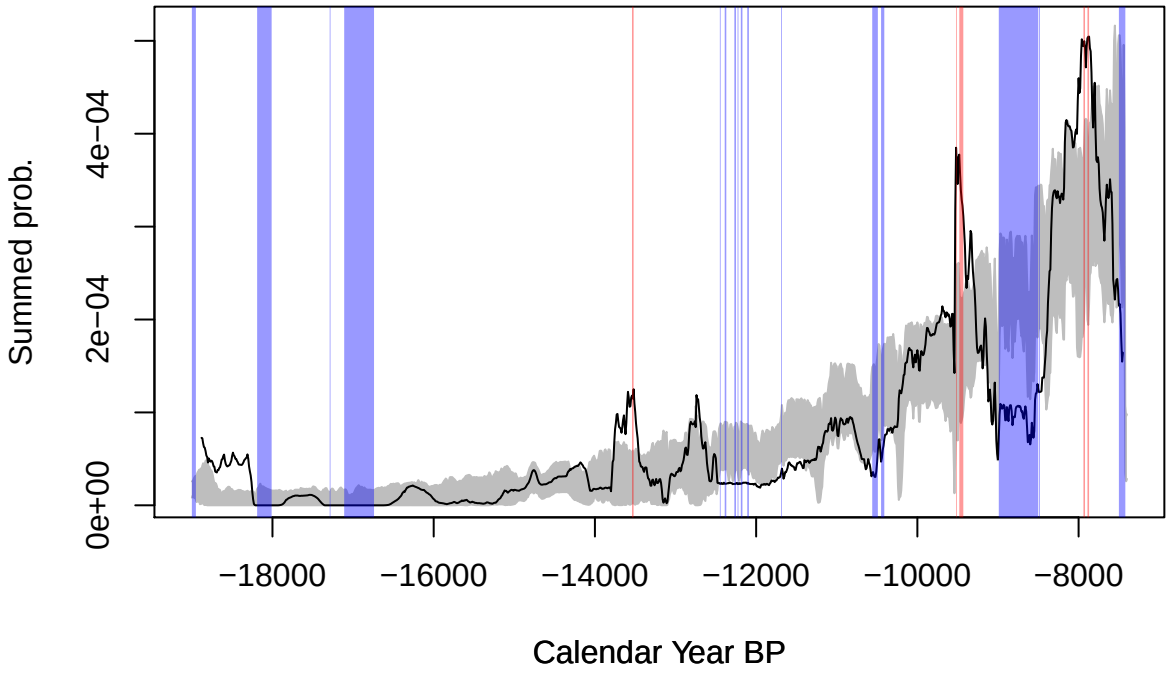



To fit an exponential 'null' model, calculate its CI, and identify significant differences, the following functions are called. More details of how they work is contained in the file `src.r`:

```
nullmod <- SPDfitexp(spdb, start=-19000, end=-7400, ndates=length(spd$dates))
nullmodb<- SPDboot(nullmod)
```

```
## ================================================================================
```

```
plot(nullmodb, col='grey')
lines(median(spdb))
polygon(SPDsignif(spdb / nullmodb))
```



To calculate the probability of the null hypothesis being true, we use the function `SPDsigniftest`. Note that due to the stochastic nature of the bootstrapping process, this will return a different value each time. The example here, for expediency, uses 100 bootstrap iterations; in the main paper we used 1000 bootstrap iterations, which is more conservative.

```
testresult <- SPDsigniftest(spdb, nullmodb)
testresult
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  ctab
## X-squared = 53.858, df = 1, p-value = 2.155e-13
```

## Comparing IntCal20 and IntCal13 curves

Calculating SPDs using the old IntCal13 dataset reveals only minor differences between the SPDs:

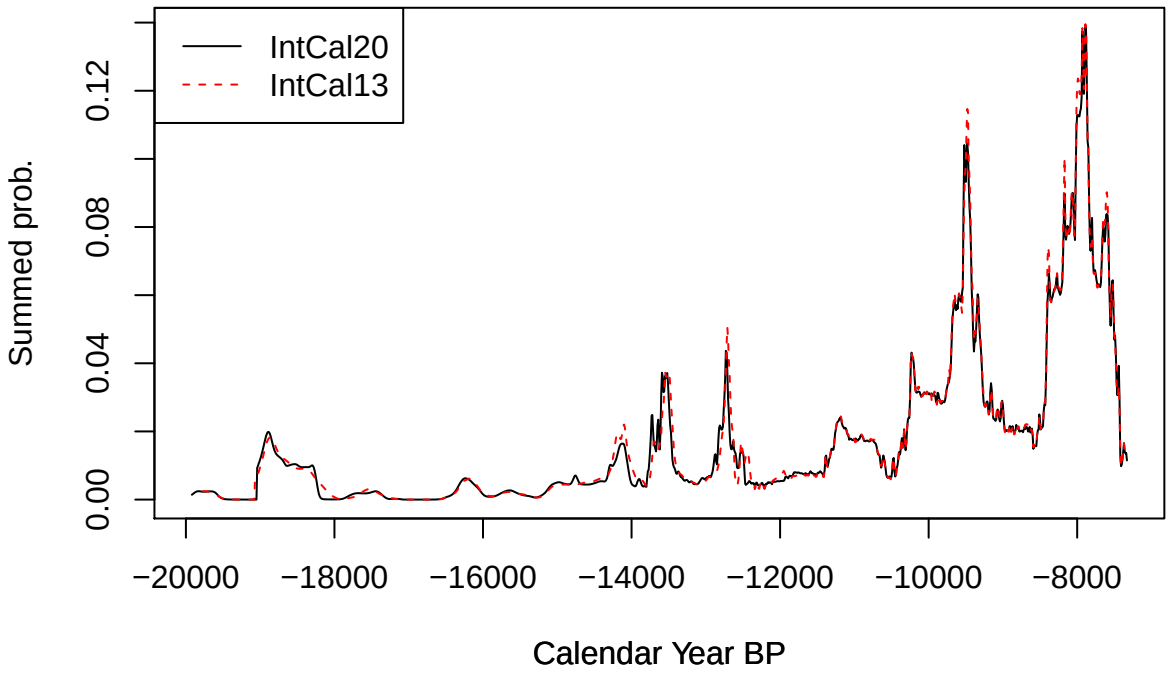
```
intcal20<-intcal
marine20<-marine
intcal<-read.csv(url("http://intcal.org/curves/intcal13.14c"),header=FALSE,skip=11)
marine<-read.csv(url("http://intcal.org/curves/marine13.14c"),header=FALSE,skip=11)
colnames(intcal)<-c("calBP","14Cage","Error","Delta14C","Sigma")
colnames(marine)<-c("calBP","14Cage","Error","Delta14C","Sigma")

openspd13 <-rowcalsum(combineddl[
  combineddl$Type.site=='Open',c('BP','Std','curve')], norm=FALSE)
```

```
## ================================================================================
```

```
otherspd13<-rowcalsum(combineddl[
  combineddl$Type.site!='Open',c('BP','Std','curve')], norm=FALSE)
```

```
## ================================================================================
```

```
spd13 <- taphSPD(openspd13) + otherspd13

plot(spd, col=NA)
lines(spd$yrs, spd$sum)
lines(spd13$yrs, spd13$sum, lty=2, col=2)
legend('topleft',lty=c(1,2),col=c(1,2),legend=c('IntCal20','IntCal13'))
```




```
# Swap back to IntCal20
intcal<-intcal20
marine<-marine20
```

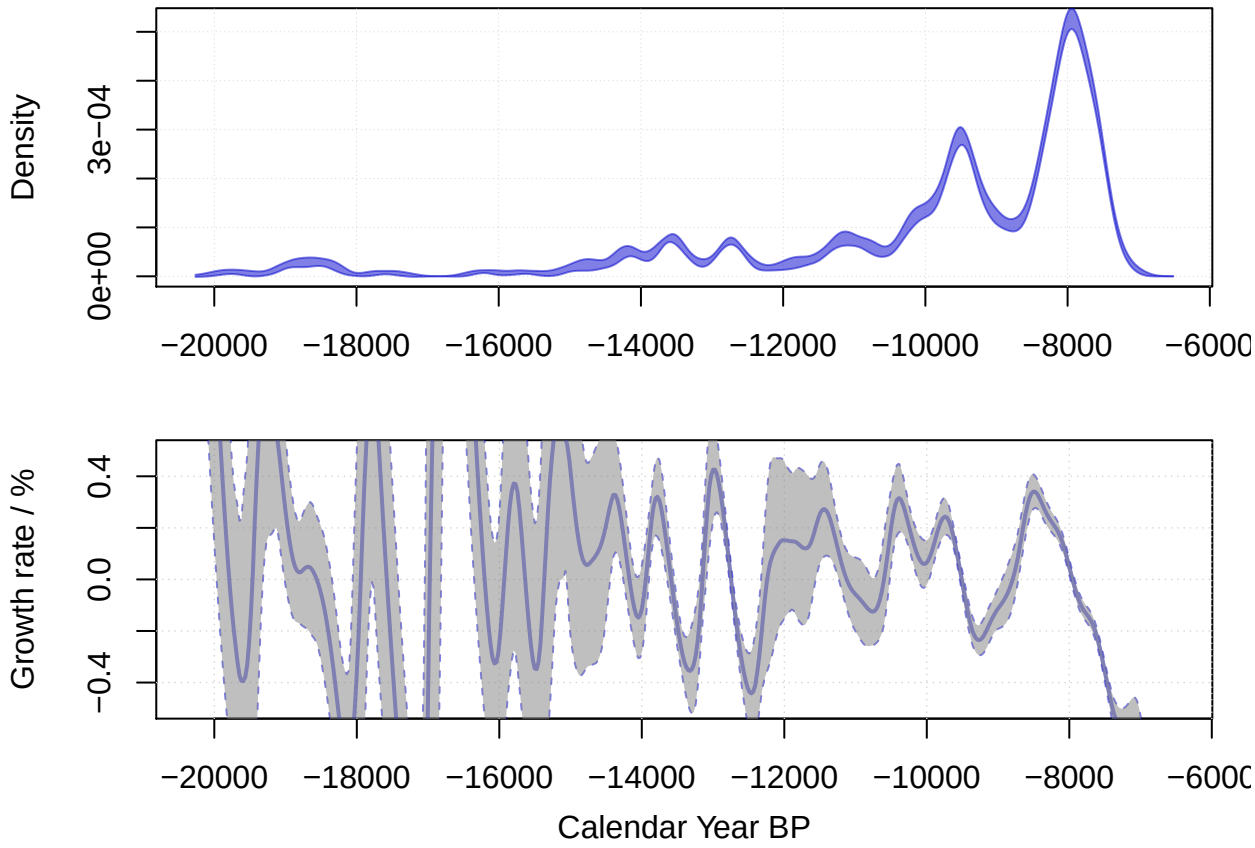
### Dynamic growth rate derived from kernel density estimates

Kernel density analysis is used to produce a timeseries that does not contain high-frequency noise. Its confidence interval is obtained using Monte Carlo simulation, as per McLaughlin 2019. The function `ggr` (geometric growth rate) obtains the first derivative of the KDE model.

```
# Build KDE model using Monte Carlo simulation
kde<-mixdensity(combineddl[,c('BP','Std','curve')], bw=150)
```

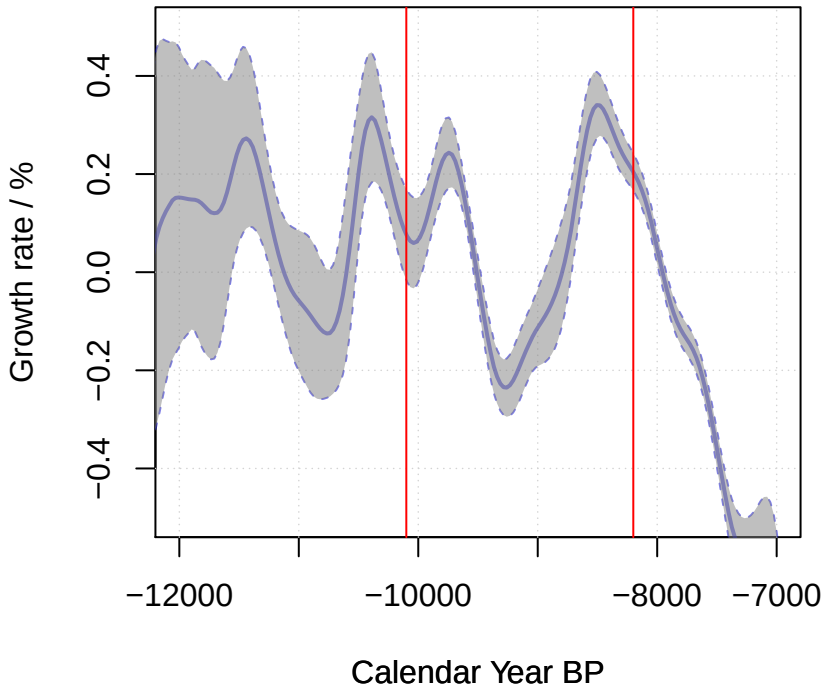
```
## ============================================================================
```

```
# Plot and compare the KDE and its first derivative
par(mfrow=c(2,1))
par(mar=c(3.5,4,0.5,1))
plot(kde, xlab='')
plot(ggr(kde), ylim=c(-.5,.5), xlab='')
mtext(side=1,line=2.3,'Calendar Year BP')
```




The following code demonstrates how to examine the Mesolithic section in detail, and query the model numerically.

```
plot(ggr(kde), xlim=c(-12000,-7000),ylim=c(-0.5,.5))

# Highlight two years and extract the values from the model
abline(v=c(-10100,-8200),col=2)
```




```
summary(year_densities(-10000,ggr(kde)))
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.1429595  0.0006701  0.0581140  0.0621019  0.1337884  0.2681371
```

```
summary(year_densities(-8200,ggr(kde)))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1035  0.1841  0.2051  0.2083  0.2298  0.3294
```

## Investigating mean inter-site distance

The mean distance between sites in a timeslice is calculated using the following function, which depends on `spatstat`.

```
# Function to make a vector containing all the distanes between sites
require(spatstat)
d<-function(coord) {
   # Calculate distances between all sets of sites with dates in the window
   dd<-pairdist(X=coord[,1], Y=coord[,2])
   dd<-dd[upper.tri(dd)]
   # Exclude any cases of two or more dates from the same site
   dd<-dd[dd>0]
   return(dd)
}
```

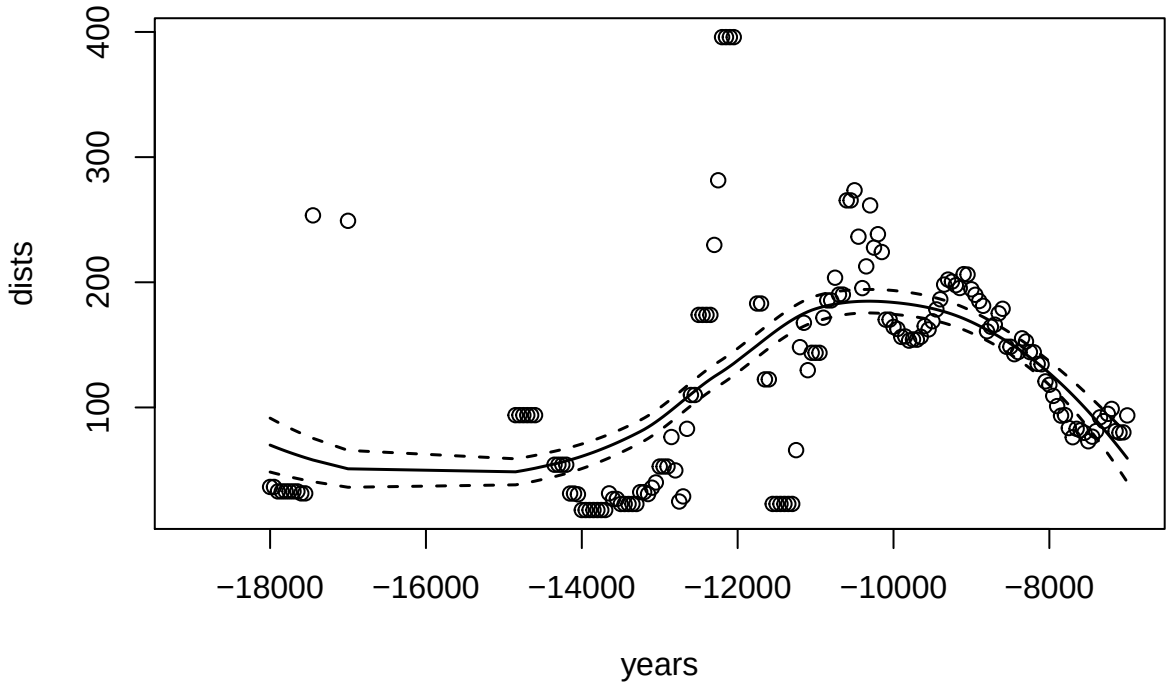
We then use this function to calculate the average distances in a moving window, stepped though the timeseries at 50-year intervals

```
dists <- c()
counter <- 1
from <- -19000 # start date
to <- -7000    # end date
```

```
by <- 50        # step increment (in years)
size <- 500     # size of moving window (in years)
years <- seq(from=from,to=to,by=by)
for(Y in seq(from=from,to=to,by=by)) {
   dists[counter]<-mean(
     d(datelist[datelist$median<Y & datelist$median>=Y-size,c('UTM_E','UTM_N')]) )
   counter<-counter+1
}
# Convert to km
dists<-dists/1000
```

To find a long-term trend, we use a LOESS regression model. We have set the span parameter of this regression heuristically, to produce models that contain a balance of exploratory power and reasonable confidence.

```
l<-loess(dists~years, span=0.8)
lp<-predict(l, se=TRUE)
# plot the output
plot(years, dists)
  lines(l$x,lp$fit, lwd=1.5)
  lines(l$x,lp$fit+lp$se.fit,lty=2, lwd=1.5)
  lines(l$x,lp$fit-lp$se.fit,lty=2, lwd=1.5)
```

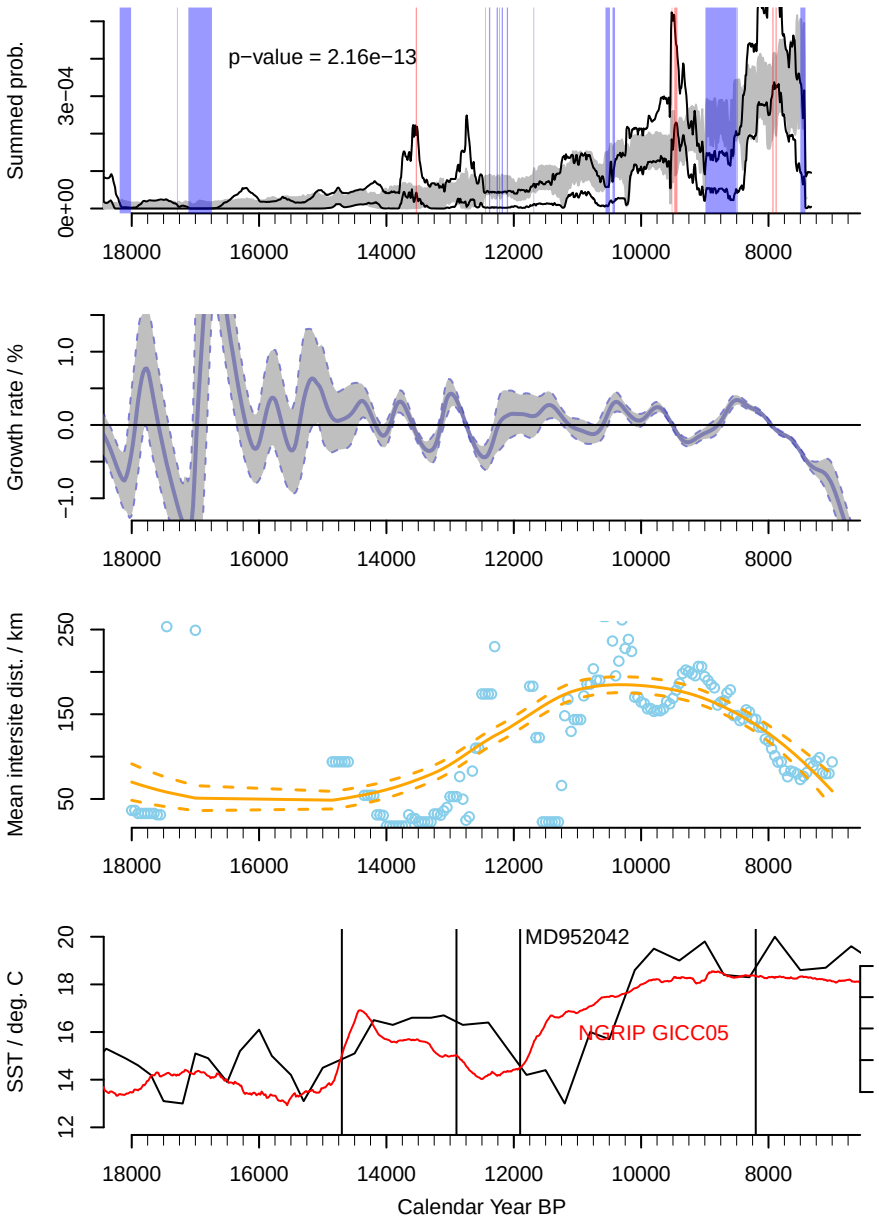



## Code to reproduce Figure 2

To reproduce Figure 2 from the main paper, the following code can be used. This also reads the palaeoclimate data from Pailler and Bard 2002 and the NGRIP GICC05 oxygen isotope series from Rasmussen et al 2006 and Andersen et al 2006.

```
pb42<-read.csv('Pailler_and_Bard_42.csv')
ngrip<-read.csv('ngrip.csv')
```

```
# Load package 'caTools' for calculating 400yr running mean of the NGRIP data
require('caTools')

par(mfrow=c(4,1))
par(mar=c(3.5,4,0.5,1))
xl<-seq(-18000,-6000,2000)

plot(nullmodb, col='grey',xlab='',xlim=c(-18000,-7000),frame.plot=FALSE,xaxt='n')
axis(1,at=xl,lab=-xl)
rug(x=seq(-18000,-6500,250), side=1, ticksize=-0.05)
lines(summary(spdb), col=1)
polygon(SPDsignif(spdb / nullmodb))
text(-15000,4e-04,paste('p-value =',signif(testresult$p.value,3)))
#lines(median(spdb))

plot(ggr(kde), grid=FALSE, xlim=c(-18000,-7000), xlab='',
     ylab='Growth rate / %', frame.plot=FALSE, ylim=c(-1.2,1.4),xaxt='n')
axis(1,at=xl,lab=-xl)
rug(x=seq(-18000,-6500,250), side=1, ticksize=-0.05)
abline(h=0)

plot(years, xlim=c(-18000,-7000),dists, col='skyblue', ylim=c(25,250),
     xlab='', frame.plot=FALSE, ylab='Mean intersite dist. / km',xaxt='n')
  lines(l$x,lp$fit, lwd=1.5, col='orange')
  lines(l$x,lp$fit+lp$se.fit,lty=2, lwd=1.5, col='orange')
  lines(l$x,lp$fit-lp$se.fit,lty=2, lwd=1.5, col='orange')
axis(1,at=xl,lab=-xl)
rug(x=seq(-18000,-6500,250), side=1, ticksize=-0.05)

plot(-1000*pb42$kBP,pb42$SST,type='l',xlab='',ylab='SST / deg. C',
     xlim=c(-18000,-7000), ylim=c(12,20),frame.plot=FALSE,xaxt='n')
axis(1,at=xl,lab=-xl)
rug(x=seq(-18000,-6500,250), side=1, ticksize=-0.05)
abline(v=c(-14700, -12900,-11900,-8200))
lines(runmean(-(ngrip[,2]-50), 1000),runmean((ngrip[,4]+60)^0.9, 20), col=2)
axis(4, at=(seq(-42,-34,2)+60)^0.9, lab=seq(-42,-34,2) )
text(c(-11000,-9800),c(20,16),c('MD952042','NGRIP GICC05'),col=c(1,2))
mtext(side=1,line=2.3,'Calendar Year BP',cex=0.66)
```




## Proxy correlation

Correlation between the SPD and another timeseries, such as palaeotemperature records, is achieved simply by aligning the two series using liner interpolation, and calling R's `cor.test` function.

```
# Align timescales and store data in a useful dataframe
pb42$BP<-pb42$kBP*-1000
popprox<-median(spdb)
yrs <- popprox$x
pop <- as.numeric(popprox$y)
pb42a <- approx(x=pb42$BP, y=pb42$SST, xout=yrs)

prox<-data.frame(yrs=yrs, pop=pop, pb42=pb42a$y)
prox<-prox[!is.na(prox$pop),]
```

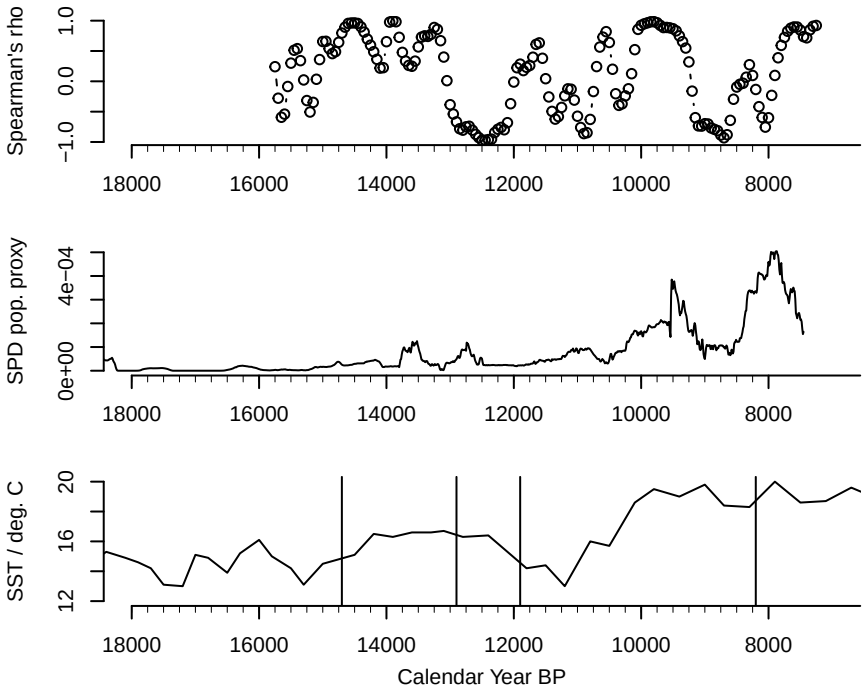
```r
# Overall correlation
cor.test(prox$pop, prox$pb42, method='spearman')$estimate
```

```
## Warning in cor.test.default(prox$pop, prox$pb42, method = "spearman"): Cannot
## compute exact p-value with ties
```

```
##       rho
## 0.7343389
```

A moving window correlation is achieved subsetting the above dataframe into a 500-year window, and running this through the data in 50-year steps, revealing an oscillation between positive and negative correlation.

```r
windows<-seq(-16000,-7500,50)

corpb42<-c()
for(N in 1:length(windows)){
   S<-prox[prox$yrs<windows[N] & prox$yrs>=windows[N]-500,]
   corpb42[N]<-cor.test(S$pop, S$pb42, method='spearman')$estimate
}

# Plot results alongside population and temperature proxies
par(mfrow=c(4,1))
par(mar=c(3.5,4,0.5,1))
plot(windows+250, corpb42, type='b', xlab='',ylab="Spearman's rho",xlim=c(-18000,-7000),
     frame.plot=FALSE,xaxt='n')
  axis(1,at=xl,lab=-xl)
rug(x=seq(-18000,-8000,250), side=1, ticksize=-0.05)
plot(median(spdb),xlab='',xlim=c(-18000,-7000),frame.plot=FALSE,
     xaxt='n', type='l', ylab='SPD pop. proxy')
  axis(1,at=xl,lab=-xl)
  rug(x=seq(-18000,-8000,250), side=1, ticksize=-0.05)
plot(-1000*pb42$kBP,pb42$SST,type='l',xlab='',ylab='SST / deg. C',xlim=c(-18000,-7000),
     ylim=c(12,20),frame.plot=FALSE,xaxt='n')
   axis(1,at=xl,lab=-xl)
   rug(x=seq(-18000,-8000,250), side=1, ticksize=-0.05)
   abline(v=c(-14700, -12900,-11900,-8200))
   mtext(side=1,line=2.3,'Calendar Year BP',cex=0.66)
```




## Palaeodietary analysis

The palaeodietary data differ from the main set of data used in the analysis because we limit our analysis to samples of human bone, but do not discard over-represented site phases. First, we load a spreadsheet containing the palaeodietary data and calculate custom calibration curves as needed.

```
pd<-read.csv('Palaeodiet.csv')
pd$pm<-pd$pm/100 #converts percertage marine to factor
pd$curveid<-rownames(pd)
pd<-addcalcurves(pd,c(11,6,7,8)) # adds calibration curves for marine samples
```

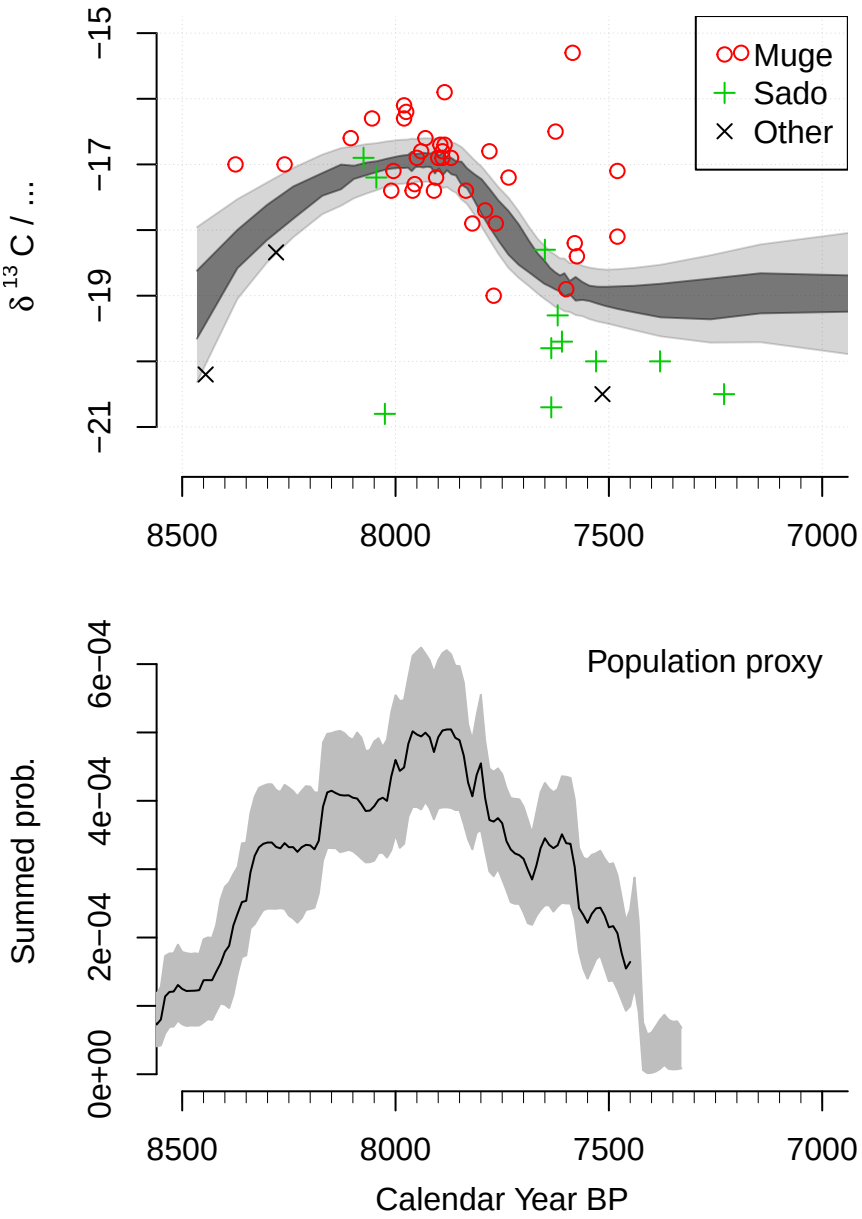
Next, we calculate multiple LOESS models, using Monte Carlo simulation to perpetuate calibration uncertainty into the LOESS models (see function `MCloess` in the `src.r` file).

```
dates<-pd[,c(9,10,12)]
diet<-pd[,4]
dietmodel<-MCloess(dates,diet)
```

The results are then plotted, replicating here Figure 3 in the main paper. For the purposes of plotting the results, we also calculate the median calibrated age for each sample, and highlight different geographic regions (in this example, the delR column in the input data uniquely segregates regions).

```
# Identify points based on region
pd$median<-findmixmedian(pd[,c(9,10,12)])
mugepd<-pd[pd$delR==140,]
sadopd<-pd[pd$delR==-100,]
opd<-pd[!pd$delR %in% c(-100,140),]

# Plot

par(mfrow=c(2,1))
par(mar=c(3.5,4,0.5,1))
```

```
xl2<-seq(-8500,-6500,500)
plot(dietmodel, xlim=c(-8500,-7000), xaxt='n', ylim=c(-21.5,-15),
     ylab = expression(delta~""^{13}~"C / %"), frame.plot=FALSE, xlab='')
axis(1, at=xl2, lab=-xl2)
rug(x=seq(-8500,-7000, 50), side=1, ticksize=-0.025)
points(opd$median, opd$d13C, pch=4)
points(mugepd$median, mugepd$d13C, pch=1, col=2)
points(sadopd$median, sadopd$d13C, pch=3, col=3)
legend('topright',pch=c(1,3,4), col=c(2,3,1), legend=c('Muge','Sado','Other'))

plot(spdb, col='grey',frame.plot = FALSE, xaxt='n',xlab='',xlim=c(-8500,-7000))
axis(1, at=xl2, lab=-xl2)
rug(x=seq(-8500,-7000, 50), side=1, ticksize=-0.025)
lines(median(spdb))
mtext(side=1,line=2.3,'Calendar Year BP')
text(-7275, 0.0006, 'Population proxy')
```




## Notes

Compiled by Rowan McLaughlin, April 2020