Supplemental Information -- Data Analyses

This document describes the data analyses that underlie the figures and derivations in the paper. Figures 1 and 2 in the main text illustrate (a) the series of GFED4 emissions and the global average (land) temperature and (b) the relationship between them. Figures 3 and 4 are used to describe (a) the relationship between the composite curve of normalized charcoal influx and the $\delta^{13}C$ of CH_4 and CH_4.and (b) the relationship between the composite curve of normalized charcoal influx and the Mann et al. (2009) global temperature data.

The global charcoal data set we use here (GCDv3) is mapped in Section 1 below. The analyses in the paper required the development of four time series of the data with a common time-step (Fig. 3), including:

- a composite curve of normalized anomalies of charcoal influx data (described in Section 2 below),
- a smooth curve of $\delta^{13}C$ of CH_4 data (Section 3),
- a smooth curve of CH_4 data (Section 4), and
- a smooth curve of Mann et al. (2009) temperature data (Section 5).

These time series are then used in regressions of:

- the relationship of normalized charcoal influx to the Mann et al. (2009) temperature data (Section 6); and
- the relationship of normalized charcoal influx of CH_4 and $\delta^{13}C$ of CH_4.(Section 7).

Figures 1 and 2 in the main text illustrate time series of GFED4 emissions and global average (land) temperatures (Fig. 1) and the relationship between them (Fig. 2) (Section 8).

We also compare land and land plus ocean temperatures from the Mann et al. (2009) temperature data (Section 9), and explore further the relationship between charcoal and temperature using an alternative data set (Section 10).

All analysis were done using R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch" (R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. [https://www.R-project.org/]

1) Charcoal-data site locations

Introduction

Here we describe the global distribution of the charcoal data in version 3 of the Global Charcoal Databse as used here. The source of the data is:

Marlon, J. R., Kelly, R., Daniau, A.-L., Vannière B., Power, M. J., Bartlein, P., Higuera, P., Blarquez, O., Brewer, S., Brücher, T., Feurdean, A., Gil Romera, G., Iglesias, V., Maezumi, S. Y., Magi, B., Courtney Mustaphi, C. J., Zhihai, T. (2016). Reconstructions of biomass burning from sediment charcoal records to improve data-model comparisons. *Biogeosciences* 13:3225-3244. [doi:10.5194/bg-13-3225-2016]

The MS-Access database itself can be downloaded from the Global Paleofire Working Group web page at: http://www.paleofire.org

The site-location data plotted here were obtained using one of the built-in queries in GCDv3, and an approach for directly accessing the database can be found at http://geog.uoregon.edu/bartlein/GPWG/GPWGAnalysis/. It is assumed here that the database has been read, and individual .csv files have been created using, for example, the `mdb-to-csv.R` script described on this web page.

Set up

Set path and filenames:

```
# paths for input .csv files -- modify as appropriate
datapath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/"
sitelistpath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_sitelists/"
sitelist <- "v3i_nsa_globe"

# site list file
sitelistfile <- paste(sitelistpath, sitelist, ".csv", sep="")
sitelistfile

## [1] "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_sitelists/v3i_nsa_globe.csv"
```

```
# site .csv  files
csvpath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_sites_csv/"
csvname <- "_data.csv"
```

Read the list of charcoal sites. Note that the site list may contain sites that after transforming and presampling may have no longer have useful data for the interval of interest here, but they are included in the sitelist here for completenesses sake.

```
# read the list of sites
sites <- read.csv(sitelistfile)
head(sites)

##   Site_ID     Lat       Lon Elev depo_context         Site_Name
## 1       1 44.6628 -110.6161 2530         LASE            Cygnet
## 2       2 44.9183 -110.3467 1884         LASE Slough Creek Pond
## 3       3 45.7044 -114.9867 2250         LASE        Burnt Knob
## 4       4 45.8919 -114.2619 2300         LASE             Baker
## 5       5 46.3206 -114.6503 1770         LASE            Hoodoo
## 6       6 45.8406 -113.4403 1921         LASE           Pintlar

ns <- length(sites[,1]) # number of sites in the sitelist
```

Define arrays for the sample-age data for each site, and for the locations of sites. Also define an indicator variable `data_2kyr` for identifying sites with samples that fall withing the last 2 kyr.

```
# arrays for data and fitted values
maxrecs <- 5000 # maximum number of samples at any site
age <- matrix(NA, ncol=ns, nrow=maxrecs) # ages
nsamples <- rep(0, maxrecs) # number of samples at each site
slon <- rep(0, ns) # longitude of sites
slat <- rep(0, ns) # latitude of sites

data_2kyr <- rep(0, ns) # indicator variable -- does the site have data within the last 2 kyr?
```

Read charcoal data

Read and store the ages of each sample. Note that sites without data, but that may be in the site list, are skipped using the `file.exists()` function. Also keep track of the site locations, and whether data are available within the last 2 kyr or not, using the variable `data_2kyr`.

```
# read and store the age of each sample at each site
ii <- 0
for (i in 1:ns) {
  #i <- 6
  sitenum <- sites[i,1]
  print(sitenum)
  siteidchar <- as.character(sitenum)
  if (sitenum >= 1) siteid <- paste("000", siteidchar, sep="")
  if (sitenum >= 10) siteid <- paste("00", siteidchar, sep="")
  if (sitenum >= 100) siteid <- paste("0", siteidchar, sep="")
  if (sitenum >= 1000) siteid <- paste(    siteidchar, sep="")

  inputfile <- paste(csvpath, siteid, csvname, sep="")
  # print(inputfile) # for debugging

  if (file.exists(inputfile)) {
    indata <- read.csv(inputfile)
    nsamp <-  length(indata$est_age)
    if (nsamp > 0) {
        ii <- ii+1
        age[1:nsamp,ii] <- indata$est_age
        nsamples[ii] <- nsamp
        slon[ii] <- sites[i,3]
        slat[ii] <- sites[i,2]
        if (age[1, ii] <= 2000.0) data_2kyr[ii] <- 1
    }
  }
}
nsites <- ii
```

Number of sites with data:

```
# number of sites with data
nsites

## [1] 703
```

Number of sites with data in the last 2 kyr:

```
# number of sites with data, last 2 kyr
sum(data_2kyr)

## [1] 666
```

Map the charcoal sites

Next, plot a map of all sites, and also indicate those sites with data from the last 2 kyr, i.e. sites that will contribute to the subsequent analyses.

Load the appropriate packages for mapping the data:

```
library(sp)
library(maptools)

## Checking rgeos availability: TRUE

library(rgdal)

## rgdal: version: 1.2-4, (SVN revision 643)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Users/bartlein/Documents/R/win-library/3.3/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Users/bartlein/Documents/R/win-library/3.3/rgdal/proj
##  Linking to sp version: 1.2-3
```

Site-data dataframe

Make a dataframe from the site data, including the `data2kyr` indicator variable:

```
gcdv3_pts <- data.frame(cbind(slon[1:nsites],slat[1:nsites],data_2kyr[1:nsites]))
names(gcdv3_pts) <- c("lon","lat","samples2k")
head(gcdv3_pts); tail(gcdv3_pts)

##         lon     lat samples2k
## 1 -110.6161 44.6628         1
## 2 -110.3467 44.9183         1
## 3 -114.9867 45.7044         1
## 4 -114.2619 45.8919         1
## 5 -114.6503 46.3206         1
## 6 -113.4403 45.8406         1

##            lon      lat samples2k
## 698  -70.88694 18.79583         1
## 699  -70.87583 18.79750         1
## 700 -122.66610 45.41111         1
## 701 -123.24280 44.44780         1
## 702 -122.95750 44.24610         1
## 703 -118.92944 48.08194         1
```

Project the site dataframe

Convert the dataframe to a SpatialPointsDataFrame (by assigning coordinates to `gcdv3_pts`):

```
library(sp)
coordinates(gcdv3_pts) <- ~lon+lat
gcdv3_pts_crs <- CRS("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")
proj4string(gcdv3_pts) <- gcdv3_pts_crs
summary(gcdv3_pts)

## Object of class SpatialPointsDataFrame
## Coordinates:
##            min      max
```

```
## lon -179.51000 179.5000
## lat  -54.88333  69.3833
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
## Number of points: 703
## Data attributes:
##    samples2k
##  Min.   :0.0000
##  1st Qu.:1.0000
##  Median :1.0000
##  Mean   :0.9474
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

... and then project the data into the Robinson projection:

```
# project the data
robin.crs <- CRS("+proj=robin +lon_0=0w")
gcdv3_pts.proj <- spTransform(gcdv3_pts, robin.crs)
```

Read basemap shapefiles

Read a set of projected shapefiles (in the Robinson projection) to use as a basemap. The shapefiles were downloaded from the *Natural Earth* web page at http://www.naturalearthdata.com, and were edited and transformed into the Robinson projection, as described here: Nice maps in R.

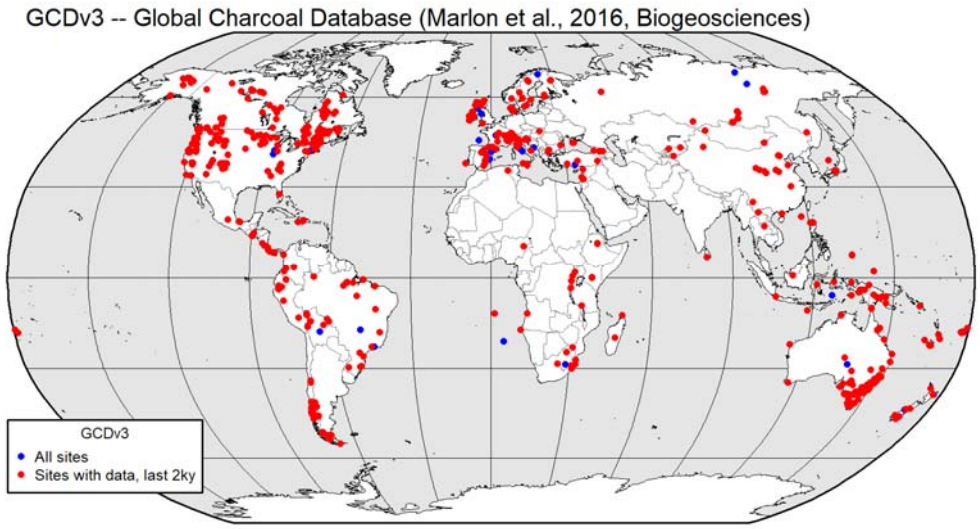
Read the projected basemap shapefiles:

```
# read the projected Robinson shapefiles
shapepath <- "/Projects/GPWG/work/feedback/data/basemaps/glRob_50m/"
coast_lines_proj <- readOGR(paste(shapepath, "glRob_50m_coast_lines.shp", sep=""))
admin0_lines_proj <- readOGR(paste(shapepath, "glRob_50m_admin0_lines.shp", sep=""))
admin0_poly_proj <- readOGR(paste(shapepath, "glRob_50m_admin0_poly.shp", sep=""))
bb_lines_proj <- readOGR(paste(shapepath, "glRob_50m_bb_lines.shp", sep=""))
bb_poly_proj <- readOGR(paste(shapepath, "glRob_50m_bb_poly.shp", sep=""))
grat30_lines_proj <- readOGR(paste(shapepath, "glRob_50m_grat30_lines.shp", sep=""))
lrglakes_poly_proj <- readOGR(paste(shapepath, "glRob_50m_lrglakes_poly.shp", sep=""))
caspian_poly_proj <- readOGR(paste(shapepath, "glRob_50m_caspian_poly.shp", sep=""))
```

Map the site data

Draw the map:

```
plot(bb_poly_proj, col="gray90", bor="black", lwd=0.1)
plot(grat30_lines_proj, col="black", lwd=0.3, add=TRUE)
plot(admin0_poly_proj, col="white", bor="gray50", lwd=0.4, add=TRUE)
plot(lrglakes_poly_proj, col="gray90", bor="black", lwd=0.2, add=TRUE)
plot(caspian_poly_proj, col="gray90", bor="black", lwd=0.2, add=TRUE)
plot(coast_lines_proj, col="black", lwd=0.5, add=TRUE)
plot(bb_lines_proj, col="black", lwd=1.0, add=TRUE)

plot(gcdv3_pts.proj[gcdv3_pts.proj$samples2k == 0,], pch=16, cex=0.5, col="blue", add=TRUE)
plot(gcdv3_pts.proj[gcdv3_pts.proj$samples2k == 1,], pch=16, cex=0.5, col="red", add=TRUE)

text(-17000000, 9100000, pos=4, cex=0.8, "GCDv3 -- Global Charcoal Database (Marlon et al., 2016,
Biogeosciences)")
legend(-17000000, -5000000, legend=c("All sites","Sites with data, last 2ky"), bg="white",
  title="GCDv3", pch=16, cex=0.5, col=c("blue","red"))
```



Discussion

As can be immediately noted, the distribution of charcoal sites is farily non-uniform, with large numbers of sites in North America, Europe, and southeasten Australia, and very few sites in Africa and Siberia. However, as shown by Marlon et al. (2008, *Nature Geoscience* 1:697-702, http://www.nature.com/doifinder/10.1038/ngeo313, the distribution of the sites in climate space is much more even, and provides reasonalbe coverage of the major biomes.

2) Composite curve of GCDv3 charcoal data

Introduction

The code below describes the fitting of a composite curve of transformed and normalized charcoal influx data using the `locfit` package. Here, a curve with a 50-year half-window width (using the default tricube weight function) is fit to the data, and evaluated at 50-year time steps.

A complete illustration of the preliminary steps, consisting of the database query, transformation and normalization and prebinning, can be found at http://geog.uoregon.edu/bartlein/GPWG/GPWGAnalysis/.

Set up

This version implements a fixed window (half) width approach in the `locfit()` function (as opposed to the "span", or a constant-proportion, variable-width window).

The variables `queryname`, `basename` and `binname` are used to compose the path and filenames for the presampled data. The pathnames are local. Note that the data being anlyzed here have already been transformed, converted to normalized anomalies, and pre-binned, following the approach above.

```
# composite curve via the locfit package
# bootstrap-by-site confidence intervals

# names
queryname <- "v3i" #
basename <- "nt2kb"

# paths for input and output .csv files -- modify as appropriate
datapath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/"
sitelistpath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_sitelists/"
sitelist <- "v3i_nsa_globe"
outpath <- "/Projects/GPWG/work/feedback/data/curves/"

# prebinning bin width
pbw <- 10

pbw_char <- as.character(pbw)
```

```
if (pbw < 10) pbw_char <- paste("0", pbw_char, sep="")

# prebinning file name
binname <- paste("bw", pbw_char, sep="")

# presampled/binned files
csvpath <- "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_presamp_csv/"
csvname <- paste("_presamp_influx_",basename,"_",binname,".csv", sep="")
```

Load the `locfit` library and set parameters. The window width `hw` is, following convention, the half-window width (of the tricube weight function). The number of bootstrap resampling replications is given by `nreps`. In this SI, `nreps` is set to 200 for efficiency. There is no difference the fitted curve (because it is fit to all data), and little difference in the bootstrap confidence intervals between analysis with `nreps` equal to 200 and `nreps` equal to 1000.

```
library(locfit)

## locfit 1.5-9.1    2013-03-22

# locfit (half) window-width parameter
# note:  hw should be an integer multiple of pbw
hw <- 50 # bandwidth (smoothing parameter)
hw_char <- as.character(hw)
if (hw < 100) hw_char <- paste("0", hw_char, sep="")

# number of bootstrap samples/replications
nreps <- 200
```

Set the target ages for fitted values.

```
# target ages for fitted values
targstep <- 50
targbeg <- -50
targend <- 2200
```

Set array sizes for saving bootstrap results.

```
# array sizes
maxrecs <- 2000
maxreps <- 1000
```

The composite curve and individual bootstrap curves can take a long time to plot. The output can be redirected to a .pdf file by setting `plotout` to `pdf`. (Set to `screen` to plot the usual way.)

```
# plotting set up
xmin <- 0; xmax <- 2020; ymin <- -0.5; ymax <- 0.5
xlim=c(xmin,xmax); ylim=c(ymin,ymax)
xlab="Year CE"; xminortick <- 50
ylab="Normalized Anomalies of Transformed Influx"

# plot output
plotout <- "screen" # "pdf" #
```

Calculation preliminaries

Initial steps

Set various output paths and filenames.

```
# no changes below here
# prebinning file name
binname <- paste("bw", pbw_char, sep="")

# site list file
sitelistfile <- paste(sitelistpath, sitelist, ".csv", sep="")
sitelistfile

## [1] "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_sitelists/v3i_nsa_globe.csv"

# curve (output) path and file
curvecsvpath <- paste(datapath,queryname,"_curves/",sep="")
```

```
# if output folder does not exist, create it
dir.create(file.path(datapath, paste(queryname,"_curves/",sep="")), showWarnings=FALSE)
curvename <- paste(sitelist,"_locfit_",basename,"_",binname,"_hw",hw_char,"_",
  as.character(nreps), sep="")
curvefile <- paste(curvename, ".csv", sep="")
print(curvecsvpath)

## [1] "/Projects/GPWG/GPWGv3/GCDv3Data/v3i/v3i_curves/"

print(curvename)

## [1] "v3i_nsa_globe_locfit_nt2kb_bw10_hw050_200"

print(curvefile)

## [1] "v3i_nsa_globe_locfit_nt2kb_bw10_hw050_200.csv"
```

Code block to implement writing a .pdf to a file:

```
# .pdf plot of bootstrap iterations
if (plotout == "pdf") {
pdffile <- paste(curvename, "_final.pdf", sep="")
print(pdffile)
}
```

Read the list of sites to be processed. Note that this is the site list may contain sites that after transforming and presampling may have no useful data. Those sites are ignored when the data are read in.

```
# read the list of sites
sites <- read.csv(sitelistfile)
head(sites)

##   Site_ID     Lat       Lon Elev depo_context         Site_Name
## 1       1 44.6628 -110.6161 2530         LASE            Cygnet
## 2       2 44.9183 -110.3467 1884         LASE Slough Creek Pond
## 3       3 45.7044 -114.9867 2250         LASE        Burnt Knob
## 4       4 45.8919 -114.2619 2300         LASE             Baker
## 5       5 46.3206 -114.6503 1770         LASE            Hoodoo
## 6       6 45.8406 -113.4403 1921         LASE           Pintlar

ns <- length(sites[,1])
ns

## [1] 703
```

Define arrays for data, fitted values and statistics.

```
# arrays for data and fitted values
age <- matrix(NA, ncol=ns, nrow=maxrecs)
influx <- matrix(NA, ncol=ns, nrow=maxrecs)
nsamples <- rep(0, maxrecs)
targage <- seq(targbeg,targend,targstep)
targage.df <- data.frame(x=targage)
lowage <- targage - hw; highage <- targage + hw
ntarg <- length(targage)
yfit <- matrix(NA, nrow=length(targage.df$x), ncol=maxreps)

# arrays for sample number and effective window span tracking
ndec <- matrix(0, ncol=ntarg, nrow=ns)
ndec_tot <- rep(0, ntarg)
xspan <- rep(0, ntarg)
ninwin <- matrix(0, ncol=ntarg, nrow=ns)
ninwin_tot <- rep(0, ntarg)
```

Read data

Read and store the data. Note that sites without data, even if in the site list, are skipped using the `file.exists()` function.

```
# read and store the presample (binned) files as matrices of ages and influx values
ii <- 0
for (i in 1:ns) {
  #i <- 1
```

```
  sitenum <- sites[i,1] # sites$ID_SITE[i]
  print(sitenum)
  siteidchar <- as.character(sitenum)
  if (sitenum >= 1) siteid <- paste("000", siteidchar, sep="")
  if (sitenum >= 10) siteid <- paste("00", siteidchar, sep="")
  if (sitenum >= 100) siteid <- paste("0", siteidchar, sep="")
  if (sitenum >= 1000) siteid <- paste(    siteidchar, sep="")

  inputfile <- paste(csvpath, siteid, csvname, sep="")
  print(inputfile)

  if (file.exists(inputfile)) {
    indata <- read.csv(inputfile)
    nsamp <-  length(indata$age) #
    if (nsamp > 0) {
        ii <- ii+1
        age[1:nsamp,ii] <- indata$age #
        influx[1:nsamp,ii] <- indata$norman # indata$zt #
        nsamples[ii] <- nsamp
    }
  }
}
nsites <- ii
```

Number of sites with data.

```
# number of sites with data
nsites

## [1] 633
```

As the presample files are read, individual files will be listed, e.g.:

```
##  [1] 1
##  [1] "/Projects/GPWG/GPWGv3/GCDv3Data/v3plus/v3plus_presamp_csv/0001_presamp_influx_zt-lme_bw10.csv"
##  ...
```

Trim the input data to an appropriate range given the target ages, and censor (set to `NA`) any transformed influx values greater than 10. (Such values typically arise in records that have few samples within the base period used in a specific analysis, and are generally downweighted or excluded by the robustness step in the `locfit()` smoothing. Here, 0.6298491 percent of the sample fall into that class, and were censored.)

```
# trim samples to age range
influx[age >= targend+hw] <- NA
age[age >= targend+hw] <- NA

# censor abs(influx) values > 10
influx[abs(influx) >= 10] <- NA
age[abs(influx) >= 10] <- NA
```

Find number of sites and samples contributing to fitted values

The number of sites with samples (`ndec_tot`) and the number of samples (`ninwin_tot`) that contribute to each fitted value are calculated, along with the effective window width or "span". This code is clunky, but parallels that in the Fortran versions.

```
# count number of sites that contributed to each fitted value
ptm <- proc.time()
for (i in 1:ntarg) {
  agemax <- -1e32; agemin <- 1e32
  for (j in 1:nsites) {
    for (k in 1:nsamples[j]) {
      if (!is.na(age[k,j])) {
        ii <- as.integer(ceiling((age[k,j]-targbeg)/targstep))+1
        #print (c(i,j,k,ii))
        if (ii > 0 && ii <= ntarg) {ndec[j,ii] = 1}
        if (age[k,j] >= targage[i]-hw && age[k,j] <= targage[i]+hw) {
          ninwin[j,i] = 1
          if (agemax <= age[k,j]) {agemax <- age[k,j]}
          if (agemin >= age[k,j]) {agemin <- age[k,j]}
        }
      }
```

```
    }
  }
  ndec_tot[i] <- sum(ndec[,i])
  ninwin_tot[i] <- sum(ninwin[,i])
  xspan[i] <- agemax - agemin
}
proc.time() - ptm

##    user  system elapsed
##   10.66    0.00   10.65

head(cbind(targage,ndec_tot,xspan,ninwin_tot))

##      targage ndec_tot xspan ninwin_tot
## [1,]     -50      125    60        347
## [2,]       0      311   100        424
## [3,]      50      303   100        413
## [4,]     100      309   100        405
## [5,]     150      310   100        421
## [6,]     200      311   100        397

tail(cbind(targage,ndec_tot,xspan,ninwin_tot))

##       targage ndec_tot xspan ninwin_tot
## [41,]    1950      249   100        352
## [42,]    2000      247   100        352
## [43,]    2050      243   100        346
## [44,]    2100      237   100        337
## [45,]    2150      224   100        336
## [46,]    2200      239    90        324
```

Curve-fitting and bootstrapping

First, the overall composite curve, i.e. without bootstrapping, is determined and saved for plotting over the individual bootstrap curves later. Second, the `nreps` individual bootstrap samples are selected, and a composite curve fitted to each and saved.

Composite curve

The steps in getting this curve include:

1. Reshaping the data matrices (`age` and `influx`) into vectors
2. Running `locfit()`, and
3. Getting fitted values at the target ages

The composite curve using all data is calculated as follows:

```
ptm <- proc.time()
# 1. reshape matrices into vectors
x <- as.vector(age)
y <- as.vector(influx)
lfdata <- data.frame(x,y)
lfdata <- na.omit(lfdata)
x <- lfdata$x; y <- lfdata$y

# 2. locfit
# initial fit, unresampled (i.e. all) data
loc01 <- locfit(y ~ lp(x, deg=1, h=hw), maxk=800, maxit=20, family="qrgauss")
summary(loc01)

## Estimation type: Local Regression
##
## Call:
## locfit(formula = y ~ lp(x, deg = 1, h = hw), maxk = 800, maxit = 20,
##     family = "qrgauss")
##
## Number of data points:  24993
## Independent variables:  x
## Evaluation structure: Rectangular Tree
## Number of evaluation points:  65
## Degree of fit:  1
## Fitted Degrees of Freedom:  4.985
```

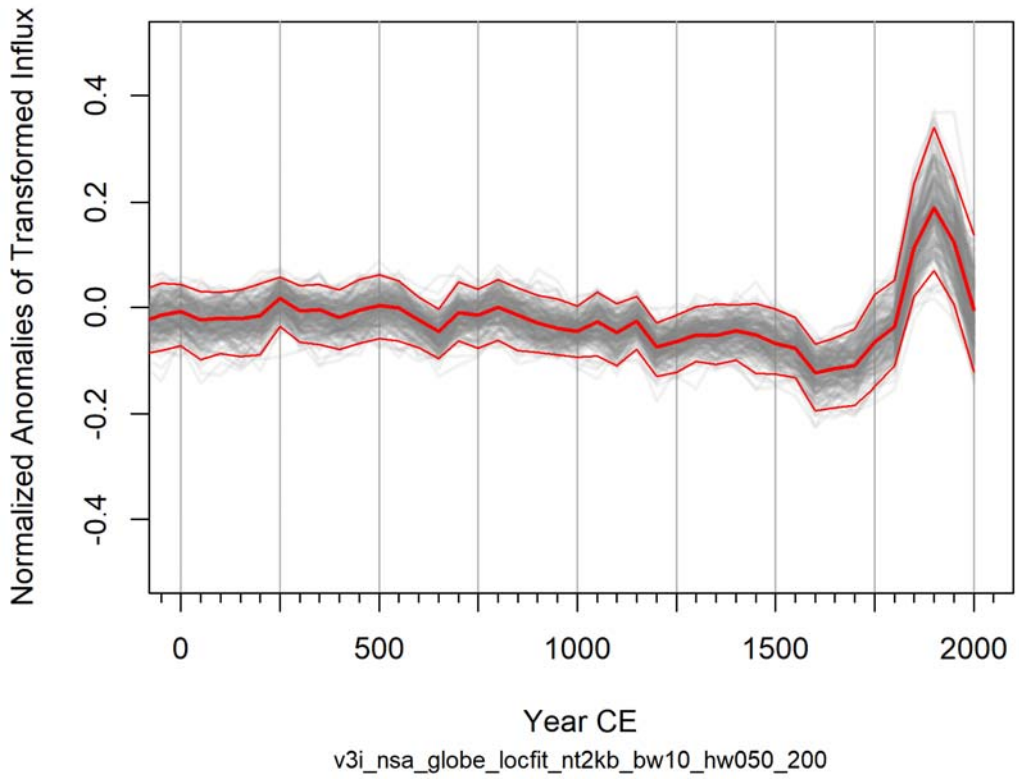
```
# 3. get  fitted values
pred01 <- predict(loc01, newdata=targage.df, se.fit=TRUE)
loc01_fit <- data.frame(targage.df$x, pred01$fit)
fitname <- paste("locfit_",as.character(hw), sep="")
colnames(loc01_fit) <- c("age", fitname)
head(loc01_fit)

##   age    locfit_50
## 1 -50 -0.004281473
## 2   0  0.124228293
## 3  50  0.190135776
## 4 100  0.114240126
## 5 150 -0.036255569
## 6 200 -0.065222342

proc.time() - ptm

##    user  system elapsed
##    2.15    0.00    2.16

ptm <- proc.time()
```

Bootstrap-by-site confidence intervals

The bootstrap confidence intervals are obtained by sampling with replacement sites (as opposed to individual samples), fitting a composite curve using `locfit()`, and saving these. The 95-percent confidence intervals are then determined for each target age using the `quantile()` function.

```
# Bootstrap samples

# Step 1 -- Set up to plot individual replications
if (plotout == "pdf") {pdf(file=paste(outpath,pdffile,sep=""))}
if (plotout == "png") {png(file=paste(outpath,pngfile,sep=""), res=150)}
plot(NULL, NULL, ylim=ylim, xlim=xlim, ylab=ylab, xlab=xlab, cex.sub=0.8,
     sub=curvename, type="n")
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tck=1.0,
     fg="gray70")
axis(side = 1, at = seq(xmin-xminortick, xmax+xminortick, by = xminortick),
     labels = FALSE, tcl = -.25)
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tcl = -.5)

# Step 2 -- Do the bootstrap iterations, and plot each composite curve
set.seed(10) # do this to get the same sequence of random samples for each run
nerr <- 0
for (i in 1:nreps) {
  print(i)
  randsitenum <- sample(seq(1:nsites), nsites, replace=TRUE)
  # print(head(randsitenum))
  x <- as.vector(age[,randsitenum])
  y <- as.vector(influx[,randsitenum])
  lfdata <- data.frame(x,y)
  lfdata <- na.omit(lfdata)
  x <- lfdata$x; y <- lfdata$y
  locboot <- locfit(y ~ lp(x, deg=1, h=hw), maxk=400, maxit=20, debug=0, family="qrgauss")
  result <- try(predict(locboot, newdata=targage.df, se.fit=TRUE), silent=TRUE)
  if (class(result) != "try-error") {
    predboot <- predict(locboot, newdata=targage.df, se.fit=TRUE)
    yfit[,i] <- predboot$fit
    lines(1950 - targage.df$x, yfit[,i], lwd=2, col=rgb(0.5,0.5,0.5,0.10))
  } else {
    print ("NA's produced")
    i <- i-1
    nerr <- nerr + 1
  }
  if (i %% 10 == 0) {print(i)}
}
print(nerr)

# Step 3 -- Plot the unresampled (initial) fit
fitname <- paste("locfit_",as.character(hw), sep="")
colnames(loc01_fit) <- c("age", fitname)
```

```
lines(1950 - loc01_fit[,1], loc01_fit[,2], lwd=2, col="red")

# Step 4 -- Find and add bootstrap CIs
yfit975 <- apply(yfit, 1, function(x) quantile(x,prob=0.975, na.rm=T)) # upper 95-percent C.I.
yfit025 <- apply(yfit, 1, function(x) quantile(x,prob=0.025, na.rm=T)) # lower 95-percent C.I.
lines(1950 - targage.df$x, yfit975, lwd=1, col="red")
lines(1950 - targage.df$x, yfit025, lwd=1, col="red")
```




```
if (plotout == "pdf") {dev.off()}
if (plotout == "png") {dev.off()}
```

Output

The fitted curves are written out in the usual way. The columns include the target age (`age`), the fitted value (`locfit`), the upper and lower 95-percent confidence limits (`yfit975` and `yfit025` respectively), the number of sites with sample that contributred to each fitted value (`nsites`), the effective window size (which is smaller than `2*h2` at the beginning and end of the record, `window`), and the number of samples (as opposed to sites) that fell in each window (`ninwin`).

```
curveout <- data.frame(cbind(targage.df$x, pred01$fit, yfit975, yfit025, ndec_tot, xspan, ninwin_tot))
colnames(curveout) <- c("age", "locfit", "cu975", "cl025", "nsites", "window", "ninwin")
outputfile <- paste(outpath, curvefile, sep="")
write.table(curveout, outputfile, col.names=TRUE, row.names=FALSE, sep=",")
```

3) Smooth curve of δ13C data

Introduction

The code below describes the fitting of a smooth curve of the δ13C data using the `locfit` package. Because the data become quite sparse before 1000 CE, the fixed window-width (for the tricube weight function) does not work well, and so the variable window-width "span" approach for local regression is used.

Set up

This version implements a variable window-width weight function approach in the `locfit()` function, with the span parameter (`nn`) equal to 0.1.

Set local pathnames.

```
# lwr (via locfit) of d13C of CH4 data
# paths for input and output .csv files -- modify as appropriate
datapath <- "/Projects/GPWG/work/feedback/data/csv/"
outpath <- "/Projects/GPWG/work/feedback/data/curves/"
```

Input data set filename:

```
dataname <- "delta_C13_adjust"
```

Response (δ13C), and predictor (Age (yr BP 1950)) variable names (in input file):

```
in_x <- "Age_yrBP1950"
in_y <- "d13C"
```

Set parameters for `locfit`, and the number of bootstrap resampling replications (`nreps`).

```
# locfit span parameter
nn <- 0.1  # bandwidth (smoothing parameter)
nn_char <- as.character(nn)

# number of bootstrap samples/replications
nreps <- 200
nreps_char <- as.character(nreps)
```

Set the target ages for fitted values.

```
# target ages for fitted values
targstep <- 50
targbeg <- -50
targend <- 2200
```

Plotting set up:

```
# plot output device
plotout <- "screen" # "pdf" # "png" #

# plotting set up
xmin <- 0; xmax <- 2020; ymin <- -50.0; ymax <- -45.0
xlim=c(xmin,xmax); ylim=c(ymin,ymax)
xlab="Year CE"; xminortick <- 50
ylab="d13C"
```

Calculation preliminaries

Initial steps

Generate target ages for fitted values.

```
# usually, no changes below here are required
# target values
targage <- seq(targbeg,targend,targstep)
targage

##  [1]  -50    0   50  100  150  200  250  300  350  400  450  500  550  600  650  700
## [17]  750  800  850  900  950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500
## [33] 1550 1600 1650 1700 1750 1800 1850 1900 1950 2000 2050 2100 2150 2200

YearCE <- 1950 - targage
YearCE

##  [1] 2000 1950 1900 1850 1800 1750 1700 1650 1600 1550 1500 1450 1400 1350 1300 1250
## [17] 1200 1150 1100 1050 1000  950  900  850  800  750  700  650  600  550  500  450
## [33]  400  350  300  250  200  150  100   50    0  -50 -100 -150 -200 -250

# generate a variable pltYearCE with a 1/2 time-step (abw) offset
# so step-plot type "s" registers correctly
pltYearCE <- YearCE  +  (targstep/2)
```

Generate file names for input and output files.

```
inputname <- paste(dataname, ".csv", sep="")
inputname

## [1] "delta_C13_adjust.csv"

# curve (output) file
curvename <- paste(dataname,"_lwrboot_span_", nn_char, "_", nreps_char, sep="")
curvefile <- paste(curvename, ".csv", sep="")
print(curvefile)

## [1] "delta_C13_adjust_lwrboot_span_0.1_200.csv"
```

Define arrays for fitted values.

```
# matrices for data and fitted values
maxreps <- 1000
min_age <- targbeg-(targstep/2); max_age <- targend + (targstep/2)

# array for bootstrap results
targage <- seq(targbeg,targend,targstep)
targage.df <- data.frame(x=targage)
lowage <- targage ; highage <- targage
ntarg <- length(targage)
yfit <- matrix(NA, nrow=length(targage.df$x), ncol=maxreps)
```

Code block to implement writing plot to file:

```
# .pdf plot of bootstrap iterations
if (plotout == "pdf") {
  pdffile <- paste(dataname,"_lwrboot_span_",nn_char,"_",nreps_char,".pdf", sep="")
  print(pdffile)
}
# .png plot of bootstrap iterations
if (plotout == "png") {
  pngfile <- paste(dataname,"_lwrboot_span_",nn_char,"_",nreps_char,".png", sep="")
  print(pngfile)
}
```

Read the data

Read the data.

```
# read data
inputfile <- paste(datapath, inputname, sep="")
print(inputfile)

## [1] "/Projects/GPWG/work/feedback/data/csv/delta_C13_adjust.csv"

indata <- read.csv(inputfile)
head(indata); tail(indata)

##   Age_yrBP1950   d13C   Source Gas.Age..AD.
## 1        -56.0 -47.00 Mischler       2006.0
## 2        -54.4 -47.00 Mischler       2004.4
## 3        -53.0 -47.06 Ferretti       2003.0
## 4        -52.3 -46.90 Mischler       2002.3
## 5        -52.0 -47.09 Ferretti       2002.0
## 6        -51.0 -47.11 Ferretti       2001.0

##     Age_yrBP1950      d13C Source Gas.Age..AD.
## 214         1921 -46.87628 Sapart           29
## 215         2004 -47.67034 Sapart          -54
## 216         2044 -47.11113 Sapart          -94
## 217         2445 -48.51876 Sapart         -495
## 218         2445 -47.31317 Sapart         -495
## 219         2639 -46.94700 Sapart         -689

age <- indata[in_x] # Age_yrBP1950
depvar <- indata[in_y]

# trim samples to target age range
depvar <- depvar[age < targend+targstep]
```

```
age <- age[age < targend+targstep]
nrecs <- length(age)
```

Curve-fitting and bootstrapping

First, the overall composite curve, i.e. without bootstrapping, is determined and saved for plotting over the individual bootstrap curves later. Second, the `nreps` individual bootstrap samples are selected, and a composite curve fitted to each and saved.

Load the `locfit` package.

```
library(locfit)

## locfit 1.5-9.1    2013-03-22
```

Composite curve

The steps in getting this curve include:

1. Reshaping the data
2. Running `locfit()` using the "original" data, and
3. Getting fitted values at the target ages

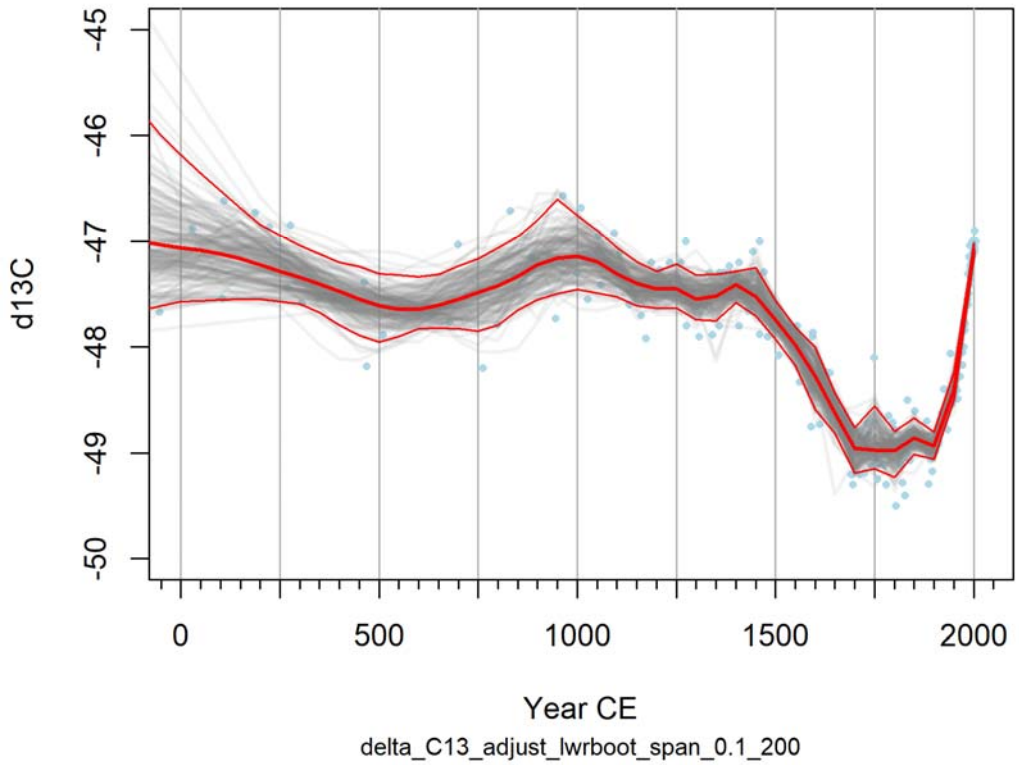
The composite curve using all data is calculated as follows:

```
# 1. reshape input data into vectors and create a data frame
x <- as.vector(age)
y <- as.vector(depvar)
lfdata <- data.frame(x,y)
lfdata <- na.omit(lfdata)
x <- lfdata$x; y <- lfdata$y

# 2. locfit
# initial fit, unresampled (i.e. all) data
loc01 <- locfit(y ~ lp(x, deg=1, nn=nn), maxk=800, maxit=20, family="qrgauss")
summary(loc01)

## Estimation type: Local Regression
##
## Call:
## locfit(formula = y ~ lp(x, deg = 1, nn = nn), maxk = 800, maxit = 20,
##     family = "qrgauss")
##
## Number of data points:  216
## Independent variables:  x
## Evaluation structure: Rectangular Tree
## Number of evaluation points:  47
## Degree of fit:  1
## Fitted Degrees of Freedom:  0.879

# 3. get  fitted values
pred01 <- predict(loc01, newdata=targage.df, se.fit=TRUE)
loc01_fit <- data.frame(targage.df$x, pred01$fit)
fitname <- paste("locfit_",as.character(nn), sep="")
colnames(loc01_fit) <- c("age", fitname)
head(loc01_fit)

##   age locfit_0.1
## 1 -50  -47.08483
## 2   0  -48.40371
## 3  50  -48.93529
## 4 100  -48.85804
## 5 150  -48.97794
## 6 200  -48.97285
```

Bootstrap confidence intervals

The bootstrap confidence intervals are obtained by sampling with replacement the individual samples, fitting a smooth curve using `locfit()`, and saving these. The 95-percent confidence intervals are then determined for each target age using the `quantile()` function.

```
# Bootstrap samples

# Step 1 -- Set up to plot individual replications
if (plotout == "pdf") {pdf(file=paste(outpath,pdffile,sep=""))}
if (plotout == "png") {png(file=paste(outpath,pngfile,sep=""), res=150)}
plot(NULL, NULL, ylim=ylim, xlim=xlim, ylab=ylab, xlab=xlab, cex.sub=0.8,
     sub=curvename, type="n")
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tck=1.0,
     fg="gray70")
axis(side = 1, at = seq(xmin-xminortick, xmax+xminortick, by = xminortick),
     labels = FALSE, tcl = -.25)
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tcl = -.5)

# plot individual data points
points(cbind((1950-indata[in_x]), indata[in_y]), pch=16, cex=0.6, col="lightblue")

# Step 2 -- Do the bootstrap iterations, and plot each composite curve
set.seed(10) # do this to get the same sequence of random samples for each run
nerr <- 0
for (i in 1:nreps) {
  print(i)
  randrecnum <- sample(seq(1:nrecs), nrecs, replace=TRUE)
  x <- as.vector(age[randrecnum])
  y <- as.vector(depvar[randrecnum])
  lfdata <- data.frame(x,y)
  lfdata <- na.omit(lfdata)
  lfdata <- lfdata[lfdata$x >= min_age & lfdata$x < max_age, ]
  lfdata_boot <- lfdata[randrecnum, ]

  x <- lfdata_boot$x; y <- lfdata_boot$y
  locboot <- locfit(y ~ lp(x, deg=1, nn=nn), maxk=400, maxit=20, debug=0, family="qrgauss")
  result <- try(predict(locboot, newdata=targage.df, se.fit=TRUE), silent=TRUE)
  if (class(result) != "try-error") {
    predboot <- predict(locboot, newdata=targage.df, se.fit=TRUE)
    yfit[,i] <- predboot$fit
    lines(1950 - targage.df$x, yfit[,i], lwd=2, col=rgb(0.5,0.5,0.5,0.10))
  } else {
    print ("NA's produced")
    i <- i-1
    nerr <- nerr + 1
  }
  if (i %% 10 == 0) {print(i)}
}
print(nerr)

# Step 3 -- Plot the unresampled (initial) fit
fitname <- paste("locfit_",as.character(nn), sep="")
colnames(loc01_fit) <- c("age", fitname)
lines(1950 - loc01_fit[,1], loc01_fit[,2], lwd=2, col="red")

# Step 4 -- Find and add bootstrap CIs
yfit975 <- apply(yfit, 1, function(x) quantile(x,prob=0.975, na.rm=T))
yfit025 <- apply(yfit, 1, function(x) quantile(x,prob=0.025, na.rm=T))
lines(1950 - targage.df$x, yfit975, lwd=1, col="red")
lines(1950 - targage.df$x, yfit025, lwd=1, col="red")
```




```
if (plotout == "pdf") {dev.off()}
if (plotout == "png") {dev.off()}
```

Output

The fitted curves are written out in the usual way.

```
curveout <- data.frame(cbind(targage, 1950-targage, loc01_fit[,2], yfit975, yfit025))
colnames(curveout) <- c("age", "YearCE", "locfit", "cu975", "cl025")
outputfile <- paste(outpath, curvefile, sep="")
write.table(curveout, outputfile, col.names=TRUE, row.names=FALSE, sep=",")
```

4) Smooth curve of CH4 data

Introduction

The code below describes the fitting of a smooth curve of the CH4 data using the `locfit` package. Because the data are generally sparse throughout the record, the fixed window-width (for the tricube weight function) does not work well, and so instead, the variable window-width "span" approach for local regression is used.

Set up

This version implements a variable window width weight function approach in the `locfit()` function, with span parameter (`nn`) equal to 0.1.

Set local pathnames

```
# lwr (via locfit) of methane data
# paths for input and output .csv files -- modify as appropriate
datapath <- "/Projects/GPWG/work/feedback/data/csv/"
outpath <- "/Projects/GPWG/work/feedback/data/curves/"
```

Input data set filename:

```
dataname <- "law2006_CH4"
```

Response (CH4), and predictor (Age (yr BP 1950)) variable names (in input file):

```
in_x <- "Age_yrBP1950"
in_y <- "CH4_ppb_NOAA04_Scale"
```

Set parameters for `locfit`, and the number of bootstrap resampling replications (`nreps`).

```
# locfit span parameter
nn <- 0.1  # bandwidth (smoothing parameter)
nn_char <- as.character(nn)

# number of bootstrap samples/replications
nreps <- 200
nreps_char <- as.character(nreps)
```

Set the target ages for fitted values.

```
# target ages for fitted values
targstep <- 50
targbeg <- -50
targend <- 2200
```

Plotting set up:

```
# plot output device
plotout <- "screen" # "pdf" # "png" #

# plotting set up
xmin <- 0; xmax <- 2020; ymin <- 0.0; ymax <- 1800.0
xlim=c(xmin,xmax); ylim=c(ymin,ymax)
xlab="Year CE"; xminortick <- 50
ylab="CH4"
```

Calculation preliminaries

Initial steps

Generate target ages for fitted values.

```
# usually, no changes below here are required
# target values
targage <- seq(targbeg,targend,targstep)
targage

##  [1]  -50    0   50  100  150  200  250  300  350  400  450  500  550  600  650  700
## [17]  750  800  850  900  950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500
## [33] 1550 1600 1650 1700 1750 1800 1850 1900 1950 2000 2050 2100 2150 2200

YearCE <- 1950 - targage
YearCE

##  [1] 2000 1950 1900 1850 1800 1750 1700 1650 1600 1550 1500 1450 1400 1350 1300 1250
## [17] 1200 1150 1100 1050 1000  950  900  850  800  750  700  650  600  550  500  450
## [33]  400  350  300  250  200  150  100   50    0  -50 -100 -150 -200 -250

# generate a variable pltYearCE with a 1/2 time-step (abw) offset
# so step-plot type "s" registers correctly
pltYearCE <- YearCE  +  (targstep/2)
```

Generate file names for input and output files.

```
inputname <- paste(dataname, ".csv", sep="")
inputname

## [1] "law2006_CH4.csv"

# curve (output) file
curvename <- paste(dataname,"_lwrboot_span_", nn_char, "_", nreps_char, sep="")
curvefile <- paste(curvename, ".csv", sep="")
print(curvefile)

## [1] "law2006_CH4_lwrboot_span_0.1_200.csv"
```

Define arrays for fitted values.

```
# matrices for data and fitted values
maxreps <- 1000
min_age <- targbeg-(targstep/2); max_age <- targend + (targstep/2)

# array for bootstrap results
targage <- seq(targbeg,targend,targstep)
targage.df <- data.frame(x=targage)
lowage <- targage ; highage <- targage
ntarg <- length(targage)
yfit <- matrix(NA, nrow=length(targage.df$x), ncol=maxreps)
```

Code block to implement writing plot to file:

```
# .pdf plot of bootstrap iterations
if (plotout == "pdf") {
  pdffile <- paste(dataname,"_lwrboot_span_",nn_char,"_",nreps_char,".pdf", sep="")
  print(pdffile)
}
# .png plot of bootstrap iterations
if (plotout == "png") {
  pngfile <- paste(dataname,"_lwrboot_span_",nn_char,"_",nreps_char,".png", sep="")
  print(pngfile)
}
```

Read the data

Read the data.

```
# read data
inputfile <- paste(datapath, inputname, sep="")
print(inputfile)

## [1] "/Projects/GPWG/work/feedback/data/csv/law2006_CH4.csv"

indata <- read.csv(inputfile)
head(indata); tail(indata)

##   Age_yrBP1950 CH4_ppb_NOAA04_Scale CH4_ppb Sample_Type publication_status  X
## 1       1935.7                655.5   647.6         DSS    MacFM 2004/2006 NA
## 2       1919.5                641.6   633.9         DSS    MacFM 2004/2006 NA
## 3       1893.0                636.1   628.4         DSS    MacFM 2004/2006 NA
## 4       1844.5                639.7   632.0         DSS    MacFM 2004/2006 NA
## 5       1813.0                646.6   638.8         DSS    MacFM 2004/2006 NA
## 6       1780.8                648.7   640.9         DSS    MacFM 2004/2006 NA
##   CH4.gas_age_years_AD
## 1                 14.3
## 2                 30.5
## 3                 57.0
## 4                105.5
## 5                137.0
## 6                169.2

##     Age_yrBP1950 CH4_ppb_NOAA04_Scale CH4_ppb Sample_Type publication_status  X
## 326        -53.1                1727.5  1706.7   CAPE GRIM                    NA
## 327        -53.2                1727.0  1706.2   CAPE GRIM                    NA
## 328        -53.2                1728.1  1707.2   CAPE GRIM                    NA
## 329        -53.4                1730.5  1709.7   CAPE GRIM                    NA
## 330        -54.1                1728.1  1707.3   CAPE GRIM                    NA
## 331        -54.5                1729.7  1708.8   CAPE GRIM                    NA
##     CH4.gas_age_years_AD
## 326               2003.1
## 327               2003.2
## 328               2003.2
## 329               2003.4
## 330               2004.1
## 331               2004.5

age <- indata[in_x]
depvar <- indata[in_y]

# trim samples to age range
depvar <- depvar[age < targend+targstep]
```

```
age <- age[age < targend+targstep]
nrecs <- length(age)
```

Curve-fitting and bootstrapping

First, the overall composite curve, i.e. without bootstrapping, determined and saved for plotting over the individual bootstrap curves later. Second, the `nreps` individual bootstrap samples are selected, and a composite curve fitted to each and saved.

Load the `locfit` package.

```
library(locfit)

## locfit 1.5-9.1    2013-03-22
```

Composite curve

The steps in getting this curve include:

1. Reshaping the data
2. Running `locfit()` using the "original" data, and
3. Getting fitted values at the target ages

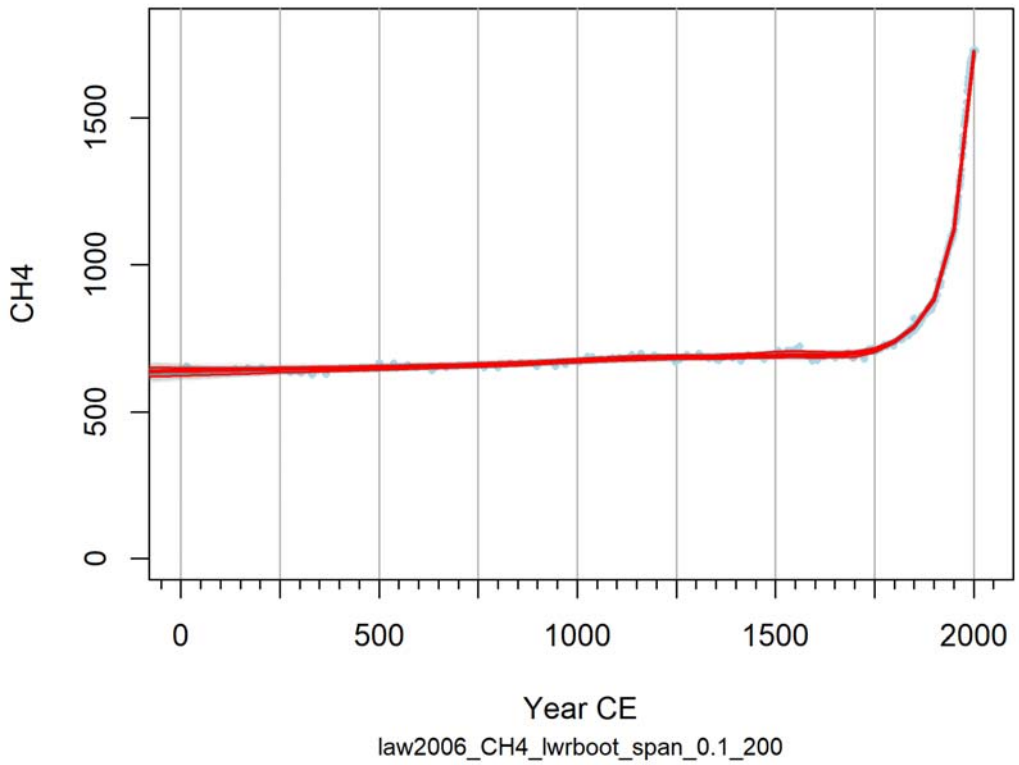
The composite curve using all data is calculated as follows:

```
# 1. reshape matrices into vectors
x <- as.vector(age)
y <- as.vector(depvar)
lfdata <- data.frame(x,y)
lfdata <- na.omit(lfdata)
x <- lfdata$x; y <- lfdata$y

# 2. locfit
# initial fit, unresampled (i.e. all) data
ptm <- proc.time()
loc01 <- locfit(y ~ lp(x, deg=1, nn=nn), maxk=800, maxit=20, family="qrgauss")
summary(loc01)

## Estimation type: Local Regression
##
## Call:
## locfit(formula = y ~ lp(x, deg = 1, nn = nn), maxk = 800, maxit = 20,
##     family = "qrgauss")
##
## Number of data points:  331
## Independent variables:  x
## Evaluation structure: Rectangular Tree
## Number of evaluation points:  46
## Degree of fit:  1
## Fitted Degrees of Freedom:  104

# 3. get  fitted values
pred01 <- predict(loc01, newdata=targage.df, se.fit=TRUE)
loc01_fit <- data.frame(targage.df$x, pred01$fit)
fitname <- paste("locfit_",as.character(nn), sep="")
colnames(loc01_fit) <- c("age", fitname)
head(loc01_fit)

##   age locfit_0.1
## 1 -50  1724.6061
## 2   0  1119.9040
## 3  50   880.9026
## 4 100   788.1916
## 5 150   738.2398
## 6 200   708.1790
```

Bootstrap confidence intervals

The bootstrap confidence intervals are obtained by sampling with replacement the individual samples, fitting a smooth curve using `locfit()`, and saving these. The 95-percent confidence intervals are then determined for each target age using the `quantile()` function.

```
# Bootstrap samples

# Step 1 -- Set up to plot individual replications
if (plotout == "pdf") {pdf(file=paste(outpath,pdffile,sep=""))}
if (plotout == "png") {png(file=paste(outpath,pngfile,sep=""), res=150)}
plot(NULL, NULL, ylim=ylim, xlim=xlim, ylab=ylab, xlab=xlab, cex.sub=0.8,
     sub=curvename, type="n")
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tck=1.0,
     fg="gray70")
axis(side = 1, at = seq(xmin-xminortick, xmax+xminortick, by = xminortick),
     labels = FALSE, tcl = -.25)
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tcl = -.5)

# plot individual data points
points(cbind((1950-indata[in_x]), indata[in_y]), pch=16, cex=0.6, col="lightblue")

# Step 2 -- Do the bootstrap iterations, and plot each composite curve
set.seed(10) # do this to get the same sequence of random samples for each run
nerr <- 0
for (i in 1:nreps) {
  print(i)
  randrecnum <- sample(seq(1:nrecs), nrecs, replace=TRUE)
  # print(head(randrecnum))
  x <- as.vector(age[randrecnum])
  y <- as.vector(depvar[randrecnum])
  lfdata <- data.frame(x,y)
  lfdata <- na.omit(lfdata)
  lfdata <- lfdata[lfdata$x >= min_age & lfdata$x < max_age, ]
  lfdata_boot <- lfdata[randrecnum, ]

  x <- lfdata_boot$x; y <- lfdata_boot$y
  locboot <- locfit(y ~ lp(x, deg=1, nn=nn), maxk=400, maxit=20, debug=0, family="qrgauss")
  result <- try(predict(locboot, newdata=targage.df, se.fit=TRUE), silent=TRUE)
  if (class(result) != "try-error") {
    predboot <- predict(locboot, newdata=targage.df, se.fit=TRUE)
    yfit[,i] <- predboot$fit
    lines(1950 - targage.df$x, yfit[,i], lwd=2, col=rgb(0.5,0.5,0.5,0.10))
  } else {
    print ("NA's produced")
    i <- i-1
    nerr <- nerr + 1
  }
  if (i %% 10 == 0) {print(i)}
}
print(nerr)

# Step 3 -- Plot the unresampled (initial) fit
fitname <- paste("locfit_",as.character(nn), sep="")
colnames(loc01_fit) <- c("age", fitname)
lines(1950 - loc01_fit[,1], loc01_fit[,2], lwd=2, col="red")
#points(1950 - loc01_fit[,1], loc01_fit[,2], pch=16, cex=0.5, col="blue")

# Step 4 -- Find and add bootstrap CIs
yfit975 <- apply(yfit, 1, function(x) quantile(x,prob=0.975, na.rm=T))
yfit025 <- apply(yfit, 1, function(x) quantile(x,prob=0.025, na.rm=T))
lines(1950 - targage.df$x, yfit975, lwd=1, col="red")
lines(1950 - targage.df$x, yfit025, lwd=1, col="red")
```




```
print(proc.time() - ptm)

if (plotout == "pdf") {dev.off()}
if (plotout == "png") {dev.off()}
```

Output

The fitted curves are written out in the usual way.

```
curveout <- data.frame(cbind(targage, 1950-targage, loc01_fit[,2], yfit975, yfit025))
colnames(curveout) <- c("age", "YearCE", "locfit", "cu975", "cl025")
outputfile <- paste(outpath, curvefile, sep="")
write.table(curveout, outputfile, col.names=TRUE, row.names=FALSE, sep=",")
```

5) Smooth curve of Mann et al. (2009) temperature data

Introduction

The code below describes the fitting of a a smooth curve to the Mann et al. (2009) area-weighted global (land) temperature reconstructions using the `locfit` package. Here, a curve with a 50-year half-window width (using the default tricube weight function) is fitted to the data.

Set up

This version implements a fixed window (half) width approach in the 'locfit()` function (as opposed to the "span", or a constant-proportion, variable-width window).

Set local pathnames.

```
# lwr (via locfit) of Mann et al. temperature data
# paths for input and output .csv files -- modify as appropriate
datapath <- "/Projects/GPWG/work/feedback/data/csv/"
outpath <- "/Projects/GPWG/work/feedback/data/curves/"
```

Input data set filename:

```
dataname <- "aaves_tmp_land"
```

Response (global average land temperature), and predictor (Age (yr BP 1950)) variable names (in input file):

```
in_x <- "Age_yrBP1950"
in_y <- "GLOBE"
```

Set parameters for `locfit()` and the number of bootstrap sampling replications (`nreps`). The window width `hw` is, following convention, the half-window width (of the tricube weight function). The number of bootstrap resampling replications is given by `nreps`.

```
# locfit (half) window-width parameter
hw <- 50 # bandwidth (smoothing parameter)
hw_char <- as.character(hw)
if (hw < 100) hw_char <- paste("0", hw_char, sep="")

# number of bootstrap samples/replications
nreps <- 200
nreps_char <- as.character(nreps)
```

Set the target ages for fitted values.

```
# target ages for fitted values
targstep <- 50
targbeg <- -50
targend <- 1450
```

Plotting set up:

```
# plot output
plotout <- "screen" #  "pdf" # "png" #

# plotting set up
xmin <- 0; xmax <- 2020; ymin <- -1.0; ymax <- 1.0
xlim=c(xmin,xmax); ylim=c(ymin,ymax)
xlab="Year CE"; xminortick <- 50
ylab="Mann et al. temperature"
```

Calculation preliminaries

Initial steps

Generate target points for fitted values.

```
# no changes below here
# target values
targage <- seq(targbeg,targend,targstep)
targage

##  [1]  -50    0   50  100  150  200  250  300  350  400  450  500  550  600  650  700
## [17]  750  800  850  900  950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450

YearCE <- 1950 - targage
YearCE

##  [1] 2000 1950 1900 1850 1800 1750 1700 1650 1600 1550 1500 1450 1400 1350 1300 1250
## [17] 1200 1150 1100 1050 1000  950  900  850  800  750  700  650  600  550  500

# generate a variable pltYearCE with a 1/2 time-step (abw) offset
# so step-plot type "s" registers correctly
pltYearCE <- YearCE  +  (targstep/2)
```

Generate file names for input and output files.

```
inputname <- paste(dataname, ".csv", sep="")
inputname

## [1] "aaves_tmp_land.csv"

# curve (output) file
curvename <- paste(dataname,"_lwrboot_hw",hw_char,"_", nreps_char, sep="")
curvefile <- paste(curvename, ".csv", sep="")
print(curvefile)
```

```
## [1] "aaves_tmp_land_lwrboot_hw050_200.csv"
```

Define arrays for fitted values.

```
# matrices for data and fitted values
maxreps <- 1000
min_age <- targbeg-(targstep/2); max_age <- targend + (targstep/2)

# array for bootstrap results
targage <- seq(targbeg,targend,targstep)
targage.df <- data.frame(x=targage)
lowage <- targage - hw; highage <- targage + hw
ntarg <- length(targage)
yfit <- matrix(NA, nrow=length(targage.df$x), ncol=maxreps)
```

Code block to implement writing .pdf to file:

```
# .pdf plot of bootstrap iterations
if (plotout == "pdf") {
  pdffile <- paste(dataname,"_lwrboot_",hw_char,"_",nreps_char,".pdf", sep="")
  print(pdffile)
}
# .png plot of bootstrap iterations
if (plotout == "png") {
  pngfile <- paste(dataname,"_lwrboot_",hw_char,"_",nreps_char,".png", sep="")
  print(pngfile)
}
```

Read the data

Read the data, and reshape

```
# read data
inputfile <- paste(datapath, inputname, sep="")
print(inputfile)

## [1] "/Projects/GPWG/work/feedback/data/csv/aaves_tmp_land.csv"

indata <- read.csv(inputfile)
head(indata); tail(indata)

##   Age_yrBP1950     GLOBE    Nhemis    Shemis  Americas NonAmericas Year
## 1         1450 -0.079763 -0.072276 -0.099327 -0.097123   -0.072423  500
## 2         1449 -0.082286 -0.074604 -0.102358 -0.105380   -0.072522  501
## 3         1448 -0.085331 -0.077275 -0.106381 -0.116327   -0.072225  502
## 4         1447 -0.087564 -0.079037 -0.109842 -0.126770   -0.070987  503
## 5         1446 -0.087372 -0.078352 -0.110938 -0.133218   -0.067987  504
## 6         1445 -0.083136 -0.073647 -0.107932 -0.132505   -0.062263  505

##      Age_yrBP1950    GLOBE   Nhemis   Shemis Americas NonAmericas Year
## 1502          -51 0.634493 0.720904 0.408713 0.482653    0.698693 2001
## 1503          -52 0.659177 0.746519 0.430961 0.508005    0.723094 2002
## 1504          -53 0.678379 0.763815 0.455146 0.549697    0.732788 2003
## 1505          -54 0.695755 0.778418 0.479769 0.608625    0.732595 2004
## 1506          -55 0.712816 0.793885 0.500992 0.678728    0.727228 2005
## 1507          -56 0.728328 0.810616 0.513319 0.747950    0.720030 2006

age <- indata[in_x] # Age_yrBP1950
depvar <- indata[in_y]

# trim samples to age range
depvar <- depvar[age < targend+targstep]
age <- age[age < targend+targstep]
nrecs <- length(age)
```

Curve-fitting and bootstrapping

First, the overall composite curve, i.e. without bootstrapping, determined and saved for plotting over the individual bootstrap curves later. Second, the `nreps` individual bootstrap samples are selected, and a composite curve fitted to each and saved.

Load the `locfit` package.

```
library(locfit)

## locfit 1.5-9.1    2013-03-22
```

Composite curve

The steps in getting this curve include:

1. Reshaping the data
2. Running `locfit()` using the "original" data, and
3. Getting fitted values at the target ages

The composite curve using all data is calculated as follows:

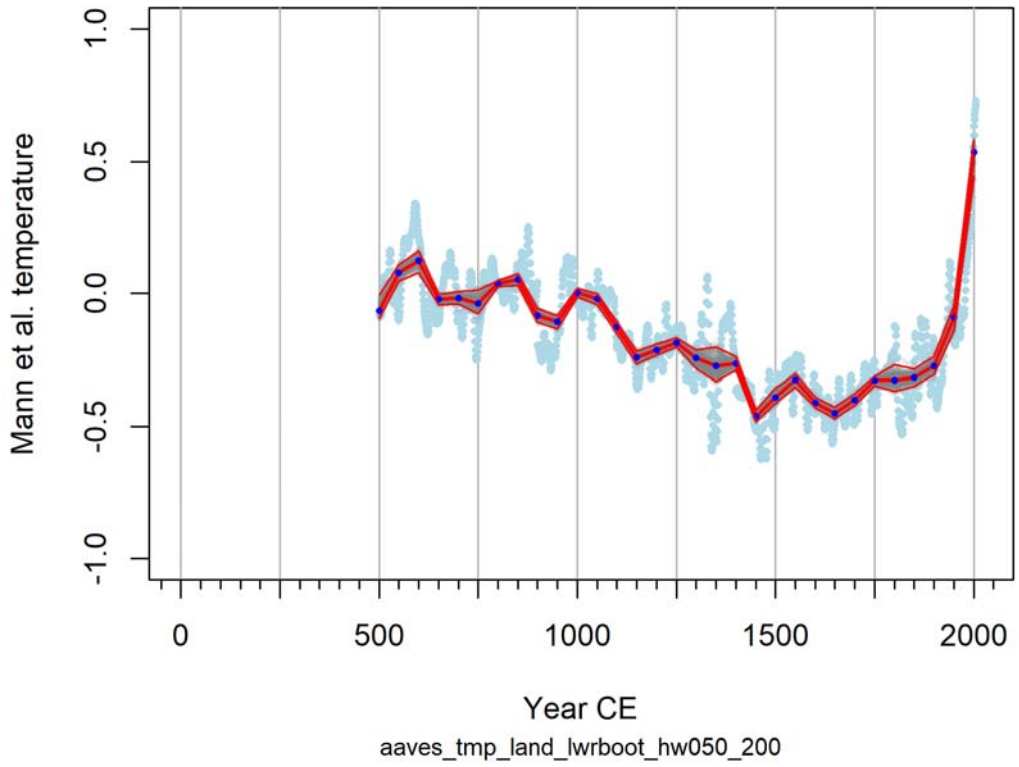
```
# 1. reshape matrices into vectors
x <- as.vector(age)
y <- as.vector(depvar)
lfdata <- data.frame(x,y)
lfdata <- na.omit(lfdata)
x <- lfdata$x; y <- lfdata$y

# 2. locfit
# initial fit, unresampled (i.e. all) data
loc01 <- locfit(y ~ lp(x, deg=1, h=hw), maxk=800, maxit=20, family="qrgauss")
summary(loc01)

## Estimation type: Local Regression
##
## Call:
## locfit(formula = y ~ lp(x, deg = 1, h = hw), maxk = 800, maxit = 20,
##     family = "qrgauss")
##
## Number of data points:  1507
## Independent variables:  x
## Evaluation structure: Rectangular Tree
## Number of evaluation points:  65
## Degree of fit:  1
## Fitted Degrees of Freedom:  0.125

# 3. get  fitted values
pred01 <- predict(loc01, newdata=targage.df, se.fit=TRUE)
loc01_fit <- data.frame(targage.df$x, pred01$fit)
fitname <- paste("locfit_",as.character(hw), sep="")
colnames(loc01_fit) <- c("age", fitname)
head(loc01_fit)

##   age   locfit_50
## 1 -50  0.53664547
## 2   0 -0.08910605
## 3  50 -0.27303344
## 4 100 -0.31614782
## 5 150 -0.32650917
## 6 200 -0.32775265
```

Bootstrap confidence intervals

The bootstrap confidence intervals are obtained by sampling with replacement the individual samples, fitting a smooth curve using `locfit()`, and saving these. The 95-percent confidence intervals are then determined for each target age using the `quantile()` function.

```
# Bootstrap samples

# Step 1 -- Set up to plot individual replications
if (plotout == "pdf") {pdf(file=paste(outpath,pdffile,sep=""))}
if (plotout == "png") {png(file=paste(outpath,pngfile,sep=""), res=150)}
plot(NULL, NULL, ylim=ylim, xlim=xlim, ylab=ylab, xlab=xlab, cex.sub=0.8,
     sub=curvename, type="n")
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tck=1.0,
     fg="gray70")
axis(side = 1, at = seq(xmin-xminortick, xmax+xminortick, by = xminortick),
     labels = FALSE, tcl = -.25)
```

```
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tcl = -.5)

# plot individual data points
points(cbind((1950-indata[in_x]), indata[in_y]), pch=16, cex=0.6, col="lightblue")

# Step 2 -- Do the bootstrap iterations, and plot each composite curve
set.seed(10) # do this to get the same sequence of random samples for each run
nerr <- 0
for (i in 1:nreps) {
  print(i)
  randrecnum <- sample(seq(1:nrecs), nrecs, replace=TRUE)
  # print(head(randrecnum))
  x <- as.vector(age[randrecnum])
  y <- as.vector(depvar[randrecnum])
  lfdata <- data.frame(x,y)
  lfdata <- na.omit(lfdata)
  lfdata <- lfdata[lfdata$x >= min_age & lfdata$x < max_age, ]
  lfdata_boot <- lfdata[randrecnum, ]

  x <- lfdata_boot$x; y <- lfdata_boot$y
  locboot <- locfit(y ~ lp(x, deg=1, h=hw), maxk=400, maxit=20, debug=0, family="qrgauss")
  result <- try(predict(locboot, newdata=targage.df, se.fit=TRUE), silent=TRUE)
  if (class(result) != "try-error") {
    predboot <- predict(locboot, newdata=targage.df, se.fit=TRUE)
    yfit[,i] <- predboot$fit
    lines(1950 - targage.df$x, yfit[,i], lwd=2, col=rgb(0.5,0.5,0.5,0.10))
  } else {
    print ("NA's produced")
    i <- i-1
    nerr <- nerr + 1
  }
  if (i %% 10 == 0) {print(i)}
}
print(nerr)

# Step 3 -- Plot the unresampled (initial) fit
fitname <- paste("locfit_",as.character(hw), sep="")
colnames(loc01_fit) <- c("age", fitname)
lines(1950 - loc01_fit[,1], loc01_fit[,2], lwd=2, col="red")
points(1950 - loc01_fit[,1], loc01_fit[,2], pch=16, cex=0.5, col="blue")

# Step 4 -- Find and add bootstrap CIs
yfit975 <- apply(yfit, 1, function(x) quantile(x,prob=0.975, na.rm=T))
yfit025 <- apply(yfit, 1, function(x) quantile(x,prob=0.025, na.rm=T))
lines(1950 - targage.df$x, yfit975, lwd=1, col="red")
lines(1950 - targage.df$x, yfit025, lwd=1, col="red")
```




```
if (plotout == "pdf") {dev.off()}
if (plotout == "png") {dev.off()}
```

Output

The fitted curves are written out in the usual way.

```
curveout <- data.frame(cbind(targage, 1950-targage, loc01_fit[,2], yfit975, yfit025))
colnames(curveout) <- c("age", "YearCE", "locfit", "cu975", "cl025")
outputfile <- paste(outpath, curvefile, sep="")
write.table(curveout, outputfile, col.names=TRUE, row.names=FALSE, sep=",")
```

6) Normalized anomalies of charcoal vs. Mann et al. temperature

Introduction

This is a charcoal vs. temperature regression, using data from locally weighted regression (`locfit()`) fits to "normans" (normalized anomalies) of charcoal from GCDv3 (version v3i, the public release) and the Mann et al. (2009) temperature data, expressed as anomalies from a 1961-1990 base period, land areas. The specific data are the fitted values for each data set, at 50-yr intervals, spanning the interval 500 through 2000 CE, the range of the Mann et al. data.

Read data

Read the appropriate data.

```
# filenames
datapath <- "/Projects/GPWG/work/feedback/data/curves/"
char_filename <- "v3i_nsa_globe_locfit_nt2kb_bw10_hw050_200.csv"
char_file <- paste(datapath, char_filename , sep="")
char_label <- "Normans (lwr, hw=50)"
tmp_filename <- "aaves_tmp_land_lwrboot_hw050_200.csv"
tmp_file <- paste(datapath, tmp_filename, sep="")
tmp_label <- "Mann et al. temp. (land) (lwr, hw=50)"

# read data
char_data <- read.csv(char_file); names(char_data)
tmp_data <- read.csv(tmp_file); names(tmp_data)
```

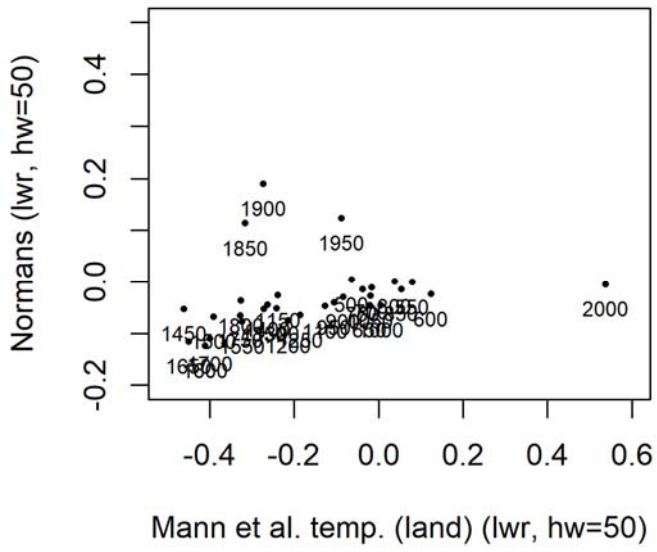
Merge the two data sets. The data span the interval from 500 CE to 2000 CE, at the 50-year intervals of the `locfit()` fitted values for each series.

```
# merge data
data_all <- as.data.frame(matrix(NA, nrow=46, ncol=4))
names(data_all) <- c("age", "YearCE", "char", "tmp")
data_all[1] <- char_data[1]
data_all[2] <- 1950-data_all[1]
data_all[3] <- char_data[2]
data_all$tmp[1:dim(tmp_data)[1]] <- tmp_data[1:dim(tmp_data)[1], 3]
data_all <- data_all[1:41, ]

# remove observations with missing temperature data
data_all <- na.exclude(data_all)
```

Scatter plots

Get a basic scatter plot.

```
attach(data_all)
plot(tmp, char, pch=16, cex=0.5, xlim=c(-.5, .6), ylim=c(-.2,.5), xlab=tmp_label, ylab=char_label)
text(tmp, char, lab=YearCE, cex=0.7, pos=1)
```




```
detach(data_all)
```

Note that 1850 to 2000 are outliers, and together, these four points have a negative slope of normalized anomalies with respect to temperature.

Pre-industrial era data

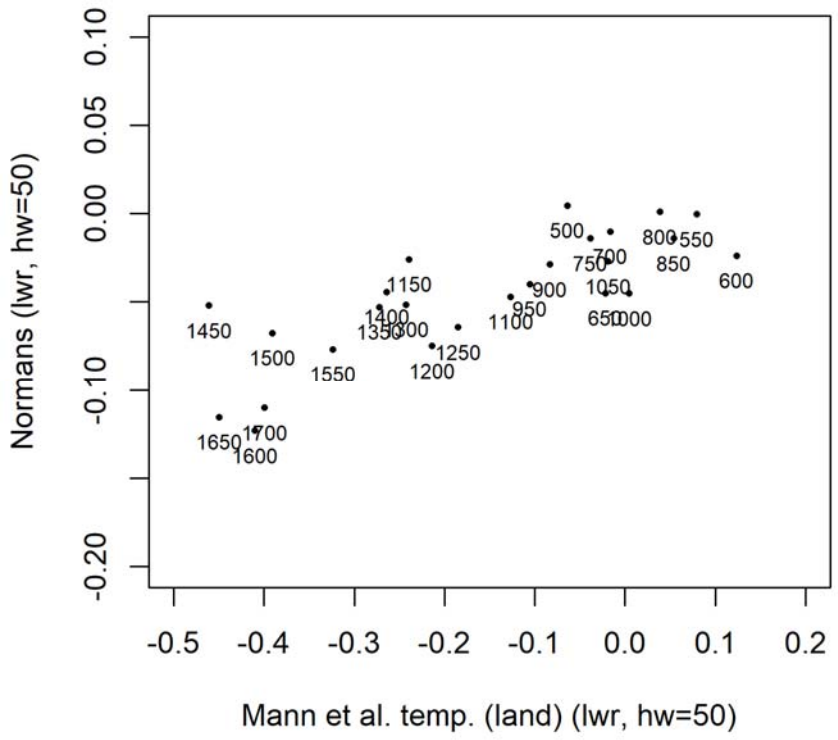
Next, restrict the data to the interval 500 to 1700 CE.

```
data_preind <- data_all[data_all$YearCE < 1750, ]
print(data_preind)
```

```
##     age YearCE         char          tmp
## 7   250   1700 -0.110073596 -0.399386316
## 8   300   1650 -0.115291532 -0.450079584
## 9   350   1600 -0.123100471 -0.410691003
## 10  400   1550 -0.076735113 -0.323993514
## 11  450   1500 -0.067580606 -0.391139017
## 12  500   1450 -0.051962066 -0.461479859
## 13  550   1400 -0.044253313 -0.264108023
## 14  600   1350 -0.052750717 -0.272250454
## 15  650   1300 -0.051476194 -0.242686168
## 16  700   1250 -0.064230215 -0.185461164
## 17  750   1200 -0.074888346 -0.214029291
## 18  800   1150 -0.025895789 -0.239769158
```

```
## 19  850   1100 -0.047103091 -0.126768932
## 20  900   1050 -0.026978098 -0.019089251
## 21  950   1000 -0.045118739  0.004320395
## 22 1000    950 -0.039846700 -0.105567878
## 23 1050    900 -0.028690003 -0.083501007
## 24 1100    850 -0.014028394  0.053885656
## 25 1150    800  0.001058576  0.038401115
## 26 1200    750 -0.014103626 -0.038636563
## 27 1250    700 -0.010158801 -0.016262921
## 28 1300    650 -0.045011700 -0.021635988
## 29 1350    600 -0.023765679  0.123927698
## 30 1400    550 -0.000424504  0.079327252
## 31 1450    500  0.004540854 -0.063826678
```

```
attach(data_preind)
plot(tmp, char, pch=16, cex=0.5, xlim=c(-.5, .2), ylim=c(-.2, .1), xlab=tmp_label, ylab=char_label)
text(tmp, char, lab=YearCE, cex=0.7, pos=1)
```




```
detach(data_preind)
```

Outliers are less apparent in the preindustrial data

Regression

Fit a linear regression with

- Normans (lwr, hw=50) as the response (`char`), and
- Mann et al. temp. (land) (lwr, hw=50) as the predictor (`tmp`).

```
char_lm01 <- lm(char ~ tmp, data=data_preind)
summary(char_lm01)
```

```
##
## Call:
## lm(formula = char ~ tmp, data = data_preind)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.037786 -0.020633 -0.002086  0.012862  0.041373
##
## Coefficients:
```

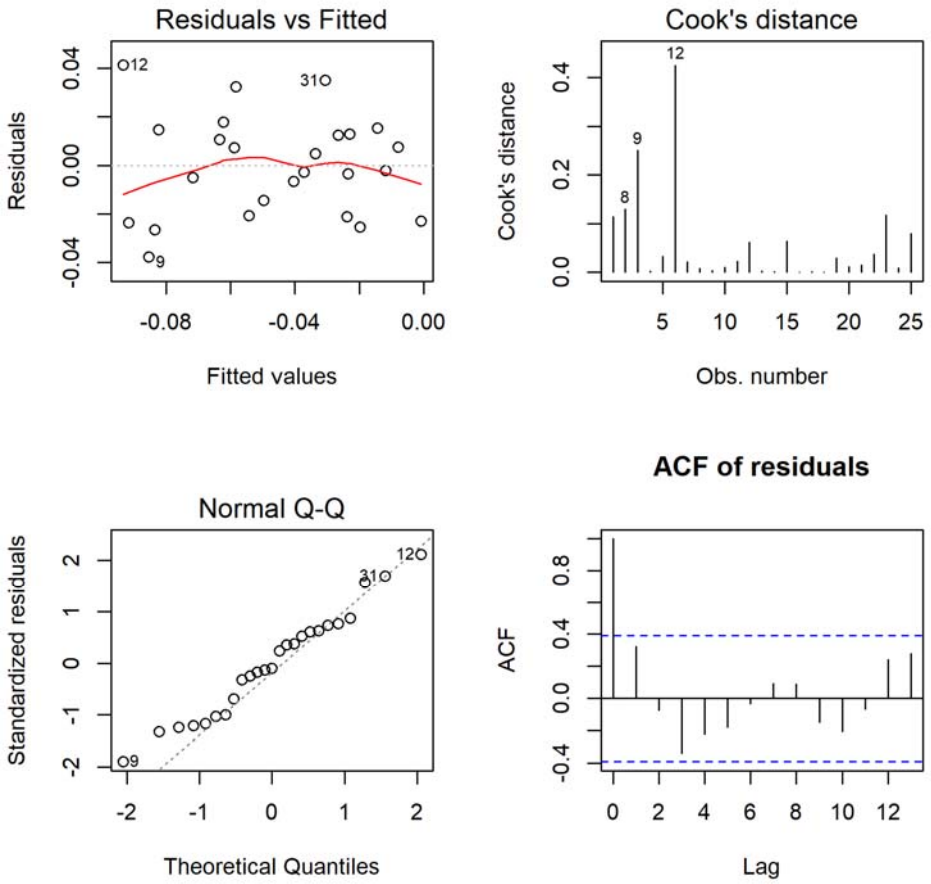
```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.020453   0.005787  -3.534  0.00177 **
## tmp          0.157932   0.024376   6.479  1.3e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02124 on 23 degrees of freedom
## Multiple R-squared:  0.646,  Adjusted R-squared:  0.6306
## F-statistic: 41.98 on 1 and 23 DF,  p-value: 1.304e-06
```

There are no issues apparent in the summary output.

Diagnostic analysis

Get the basic diagnostic plots.

```
oldpar <- par(mfcol=c(2,2))
plot(char_lm01, which=c(1,2,4))
acf(char_lm01$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar
```

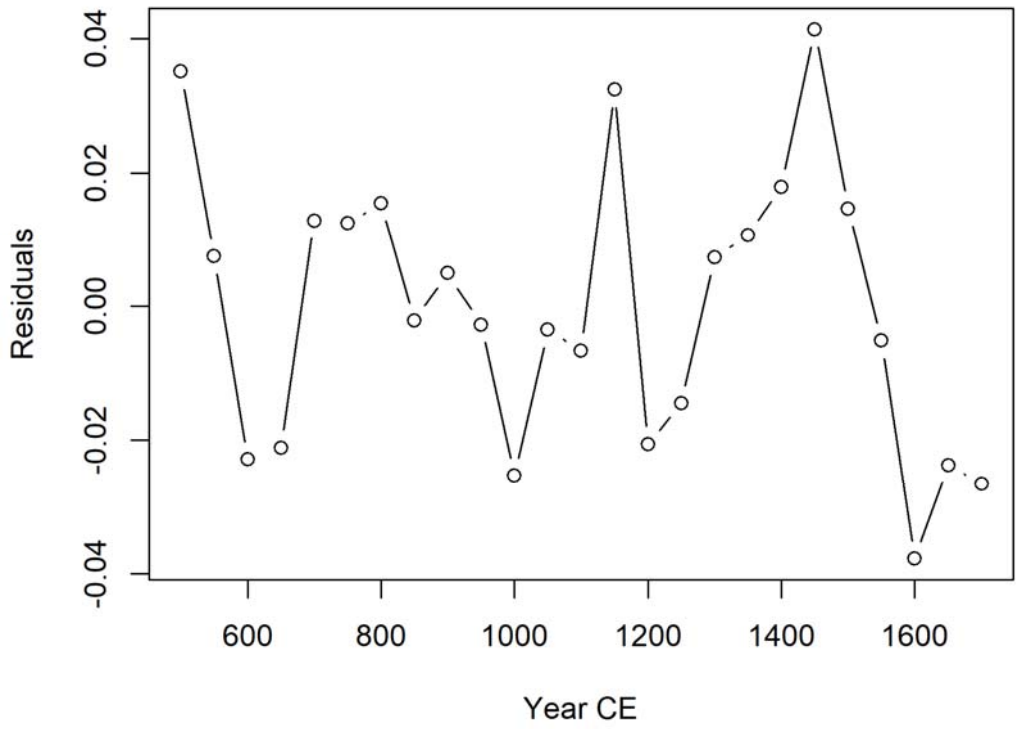
There is a little bit of an arch in the residual scatter plot, but probably not enough to want to change the basic model. Cook's distance indicate that most of the influential points are from 1450 CE and later (with 1450 having the most influence), The normal probability plot suggests the residual are normally distributed, and they also do not seem to be autocorrelated. This is supported by the Ljung-Box test of independence in a time series.

```
Box.test(char_lm01$residuals, lag=12, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  char_lm01$residuals
## X-squared = 16.154, df = 12, p-value = 0.1843
```

Plot the time series of the residuals:

```
plot(data_preind$YearCE, char_lm01$residuals, type="b", xlab="Year CE", ylab="Residuals")
```

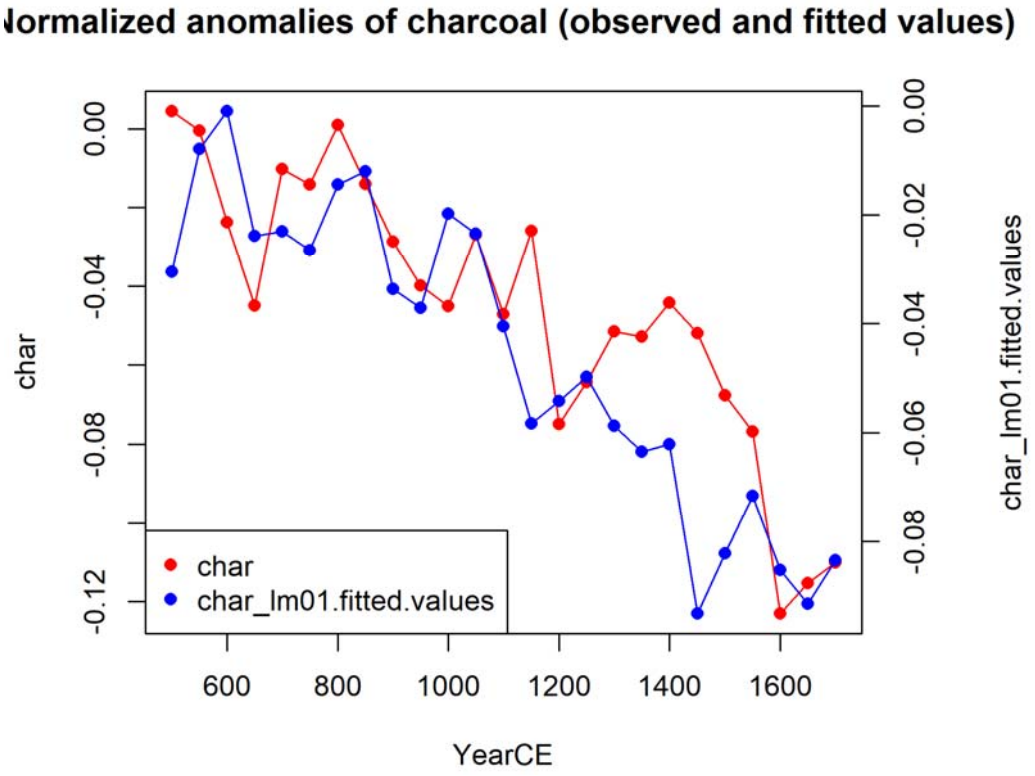


Function for plotting two series:

```
# function for plotting two series at a time
plotseries <- function(title, dataset, xvarin, yvar1in, yvar2in) {
  xvar <- dataset[,xvarin]
  yvar1 <- dataset[,yvar1in]; yvar2 <- dataset[,yvar2in]
  opar <- par(mar=c(5,4,4,5)+0.1) # space for second axis
  plot(yvar1 ~ xvar, pch=16, xlab=xvarin, ylab=yvar1in, type="o", col="red", data=dataset,
    main=title)
  par(new=T)          # second plot is going to get added to first
  plot(yvar2 ~ xvar, xlab="", pch=16, axes=F, ylab="", type="o", col="blue", data=dataset)
  axis(side=4); mtext(side=4,line=3.8,yvar2in)
  legend("bottomleft", c(yvar1in,yvar2in), pch=16, col=c("red","blue"))
  par(opar)  # restore plot parameters
}
```

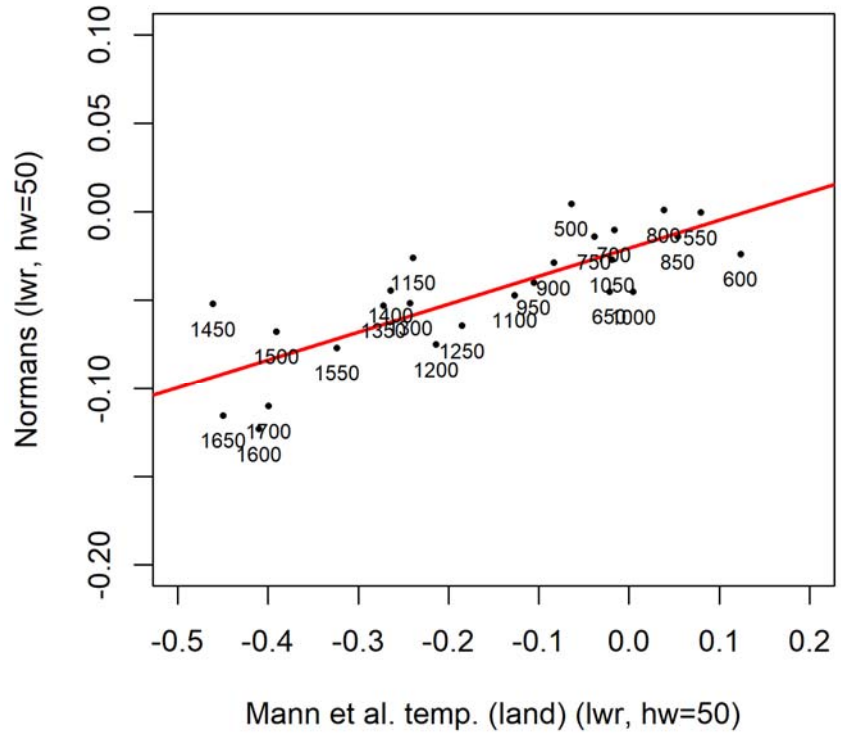
Observed and fitted values of the normalized anomalies of charcoal:

```
fitted <- data.frame(cbind(data_preind, char_lm01$residuals, char_lm01$fitted.values))
plotseries("Normalized anomalies of charcoal (observed and fitted values)", fitted, "YearCE", "char",
"char_lm01.fitted.values")
```



The basic scatter plot with the regression curve added:

```
attach(data_preind)
plot(tmp, char, pch=16, cex=0.5, xlim=c(-.5, .2), ylim=c(-.2, .1), xlab=tmp_label, ylab=char_label)
abline(char_lm01, col="red", lwd=2)
text(tmp, char, lab=YearCE, cex=0.7, pos=1)
```




```
detach(data_preind)
```

Output

The observed and fitted values are written out in the usual way.

```
curveout <- data.frame(data_preind, char_lm01$residuals, char_lm01$fitted.values)
colnames(curveout) <- c("age", "YearCE", "char", "tmp", "char_lm01$residuals", "char_lm01$fitted.values")
outpath <- "/Projects/GPWG/work/feedback/data/Mann_etal_tmp/"
outputfile <- paste(outpath, "fig02_data.csv", sep="")
write.table(curveout, outputfile, col.names=TRUE, row.names=FALSE, sep=",")
```

7) Charcoal vs. $\delta^{13}C$ and CH_4

Introduction

Regression of normalized anomalies of charcoal against $\delta^{13}C$ and CH_4 as predictors. This is the regression of F_b against M and δM (see Methods).

Read data

```
# filenames
datapath <- "/Projects/GPWG/work/feedback/data/curves/"
char_filename <- "v3i_nsa_globe_locfit_nt2kb_bw10_hw050_200.csv"
char_file <- paste(datapath, char_filename , sep="")
char_label <- "Normans (lwr, hw=50)"
d13C_filename <- "delta_C13_adjust_lwrboot_span_0.1_200.csv"
d13C_file <- paste(datapath, d13C_filename, sep="")
d13C_label <- "d13C (lwr, span=0.1)"
CH4_filename <- "law2006_CH4_lwrboot_span_0.1_200.csv"
CH4_file <- paste(datapath, CH4_filename, sep="")
CH4_label <- "CH4 (lwr, span-0.1)"

# read data
char_data <- read.csv(char_file); names(char_data)
d13C_data <- read.csv(d13C_file); names(d13C_data)
CH4_data <- read.csv(CH4_file); names(CH4_data)
```

Merge the individual data sets into a dataframe.

```
# merge data
data_all <- as.data.frame(matrix(NA, nrow=46, ncol=5))
names(data_all) <- c("age", "YearCE", "char", "d13C", "CH4")
data_all[1] <- char_data[1]
data_all[2] <- 1950-data_all[1]
data_all[3] <- char_data[2]
data_all$d13C[1:dim(d13C_data)[1]] <- d13C_data[1:dim(d13C_data)[1], 3]
data_all$CH4[1:dim(CH4_data)[1]] <- CH4_data[1:dim(CH4_data)[1], 3]
data_all <- data_all[1:41, ]

# remove observations with missing temperature data
data_all <- na.exclude(data_all)
```

Pre-industrial era data

Use the same "preindustrical" definition as for the charcoal vs. temperature regressions.

```
preind <- data_all[data_all$YearCE <= 1700, ]
print(preind)

##     age YearCE         char      d13C      CH4
## 7   250   1700 -0.110073596 -48.95273 698.1907
## 8   300   1650 -0.115291532 -48.61730 692.7086
## 9   350   1600 -0.123100471 -48.27533 693.0741
## 10  400   1550 -0.076735113 -47.98270 692.7039
## 11  450   1500 -0.067580606 -47.74417 691.6821
## 12  500   1450 -0.051962066 -47.52550 690.4336
## 13  550   1400 -0.044253313 -47.41089 688.5142
## 14  600   1350 -0.052750717 -47.51929 686.0854
## 15  650   1300 -0.051476194 -47.54650 685.3370
```

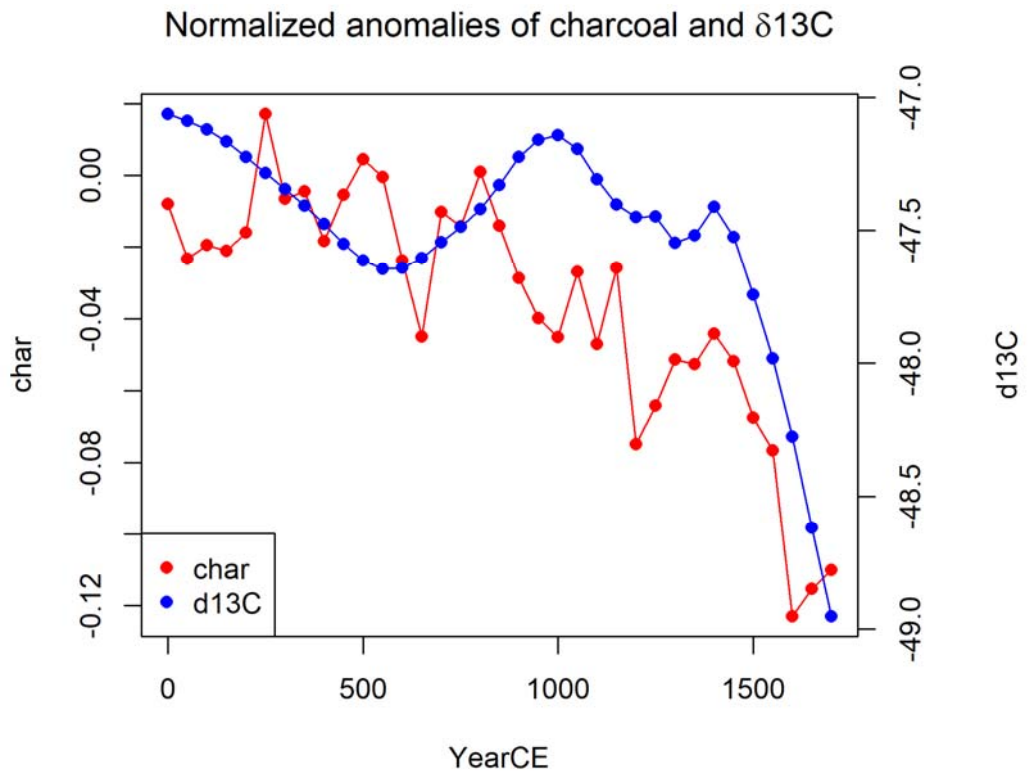
```
## 16  700   1250 -0.064230215 -47.44917 685.3902
## 17  750   1200 -0.074888346 -47.45136 684.7066
## 18  800   1150 -0.025895789 -47.40237 682.9422
## 19  850   1100 -0.047103091 -47.30916 680.2965
## 20  900   1050 -0.026978098 -47.19461 676.9689
## 21  950   1000 -0.045118739 -47.14221 673.1757
## 22 1000    950 -0.039846700 -47.16045 669.6527
## 23 1050    900 -0.028690003 -47.22347 666.6186
## 24 1100    850 -0.014028394 -47.32919 663.9463
## 25 1150    800  0.001058576 -47.42033 661.5087
## 26 1200    750 -0.014103626 -47.48631 659.1744
## 27 1250    700 -0.010158801 -47.54601 656.7536
## 28 1300    650 -0.045011700 -47.60483 654.2879
## 29 1350    600 -0.023765679 -47.64245 651.9029
## 30 1400    550 -0.000424504 -47.64740 649.7243
## 31 1450    500  0.004540854 -47.61321 647.8867
## 32 1500    450 -0.005319649 -47.55065 646.6389
## 33 1550    400 -0.018265692 -47.47639 645.8531
## 34 1600    350 -0.004501493 -47.40786 645.2696
## 35 1650    300 -0.006485827 -47.34446 644.6285
## 36 1700    250  0.017220883 -47.28414 643.6920
## 37 1750    200 -0.015875796 -47.22489 642.7013
## 38 1800    150 -0.021083701 -47.16622 641.7758
## 39 1850    100 -0.019441544 -47.12140 640.8818
## 40 1900     50 -0.023219157 -47.08878 639.9859
## 41 1950      0 -0.007912565 -47.06187 639.0570
```

Time series and scatter plots

Plot time series of the variables over the "preindustrial" period (i.e through 1700 CE).

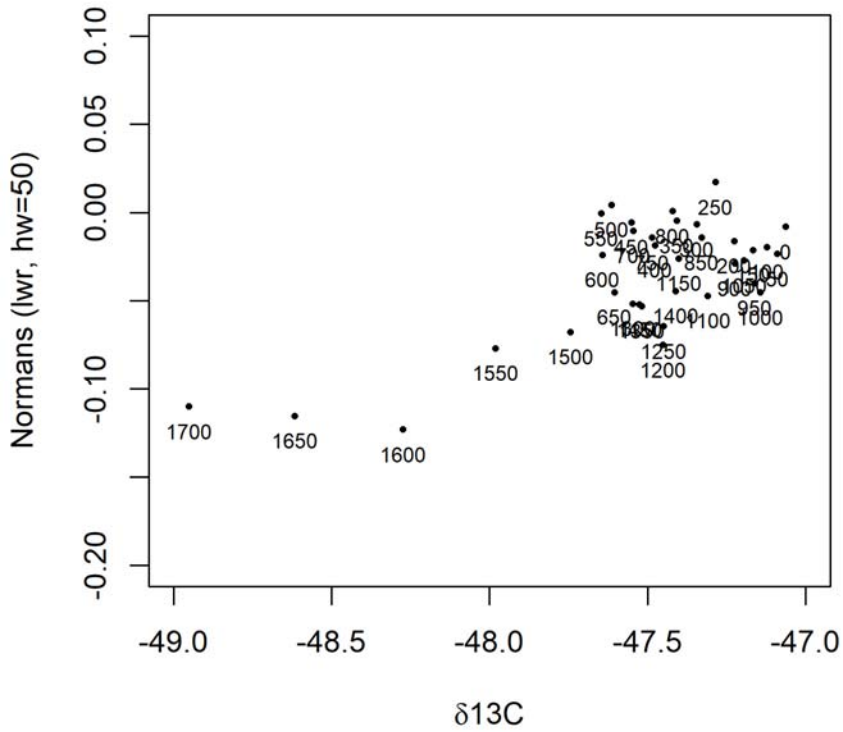
Function for plotting two series:

```
# function for plotting two series at a time
plotseries <- function(title, dataset, xvarin, yvar1in, yvar2in) {
  xvar <- dataset[,xvarin]
  yvar1 <- dataset[,yvar1in]; yvar2 <- dataset[,yvar2in]
  opar <- par(mar=c(5,4,4,5)+0.1) # space for second axis
  plot(yvar1 ~ xvar, pch=16, xlab=xvarin, ylab=yvar1in, type="o", col="red", data=dataset,
    main=title)
  par(new=T)          # second plot is going to get added to first
  plot(yvar2 ~ xvar, xlab="", pch=16, axes=F, ylab="", type="o", col="blue", data=dataset)
  axis(side=4); mtext(side=4,line=3.8,yvar2in)
  legend("bottomleft", c(yvar1in,yvar2in), pch=16, col=c("red","blue"))
  par(opar)  # restore plot parameters
}

title=expression(paste("Normalized anomalies of charcoal and ", delta, "13C", sep=""))
plotseries(title, preind, "YearCE", "char", "d13C")
```



Scatter plot of normalized transformed charcoal influx ("Normans") and δ13C, labeled by Year.

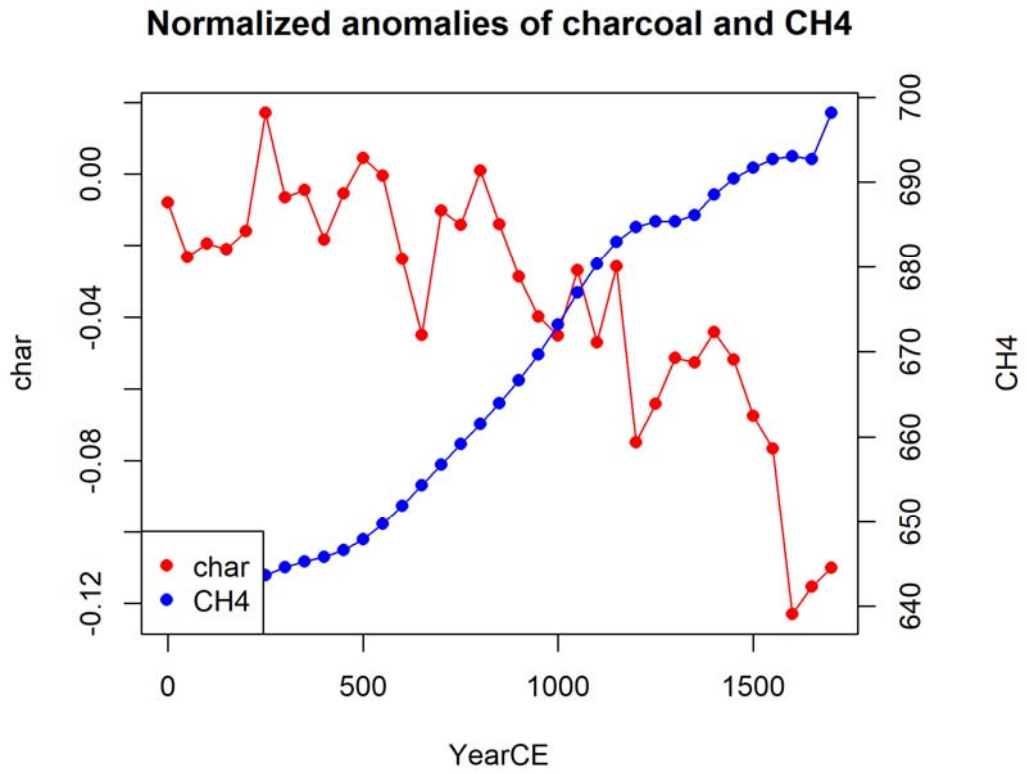
```
d13C_text <- expression(paste(delta,"13C", sep=""))
plot(preind$d13C, preind$char, pch=16, cex=0.5, xlim=c(-49, -47), ylim=c(-.2,.1), xlab=d13C_text,
ylab=char_label)
text(preind$d13C, preind$char, lab=preind$YearCE, cex=0.7, pos=1)
```



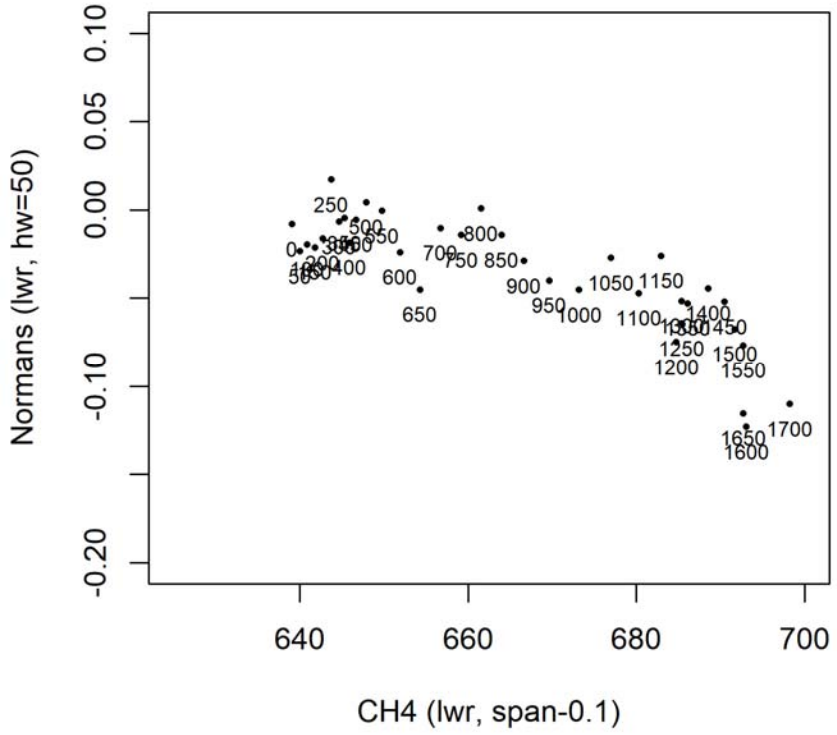
Note the detachment of the points for 1600, 1650 and 1700 CE (and to some extent those for 1500 and 1550 CE). These points all plot on the same general trend line, however, and are likely more consistent with Little Ice Age (LIA) climate trends than with the effects of early industrialization on the relationship between charcoal and δ13C.

Construct similar plots for CH4<>:

```
plotseries("Normalized anomalies of charcoal and CH4", preind, "YearCE", "char", "CH4")
```




```
plot(preind$CH4, preind$char, pch=16, cex=0.5, xlim=c(625, 700), ylim=c(-0.2, 0.1), xlab=CH4_label,
ylab=char_label)
text(preind$CH4, preind$char, lab=preind$YearCE, cex=0.7, pos=1)
```



Regression

Fit a linear regression with

- Normans (lwr, hw=50) as the response (`char`), and
- CH4 (lwr, span-0.1) and CH4 (lwr, span-0.1) x d13C (lwr, span=0.1) as the predictors.

```
check_lm01 <- lm(char ~ I(d13C * CH4) + CH4, data=preind)
summary(check_lm01)

##
## Call:
## lm(formula = char ~ I(d13C * CH4) + CH4, data = preind)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.03464 -0.01358  0.00184  0.01128  0.02900
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.593e-01  1.147e-01   5.747 2.26e-06 ***
## I(d13C * CH4) 4.679e-05  1.237e-05   3.783 0.000642 ***
## CH4           1.180e-03  6.967e-04   1.694 0.100036
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01681 on 32 degrees of freedom
## Multiple R-squared:  0.7716, Adjusted R-squared:  0.7573
## F-statistic: 54.04 on 2 and 32 DF,  p-value: 5.501e-11
```

The regression superficially looks ok, except for the large p-value from the t-test for significance of the `CH4` term. However, the regression suffers from severe collinearity owing the inclusion of the cross product of δ13C and CH4 as a predictor. The condition number of the predictor covariance matrix is quite high:

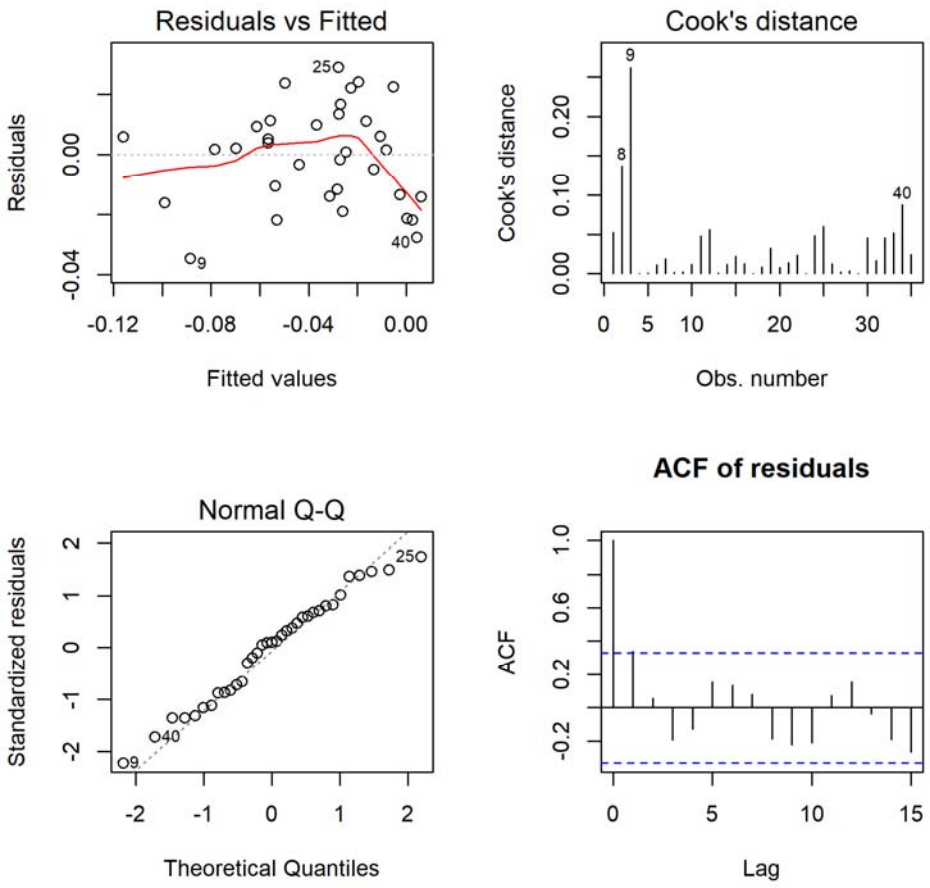
```
kappa(check_lm01)

## [1] 923586.7
```

Because of the collinearity, the significance test should be discounted.

Diagnostic analysis

Standard regression diagnostic plots:

```
oldpar <- par(mfcol=c(2,2))
plot(check_lm01, which=c(1,2,4))
acf(check_lm01$residuals, main="ACF of residuals", font.main=1)
```



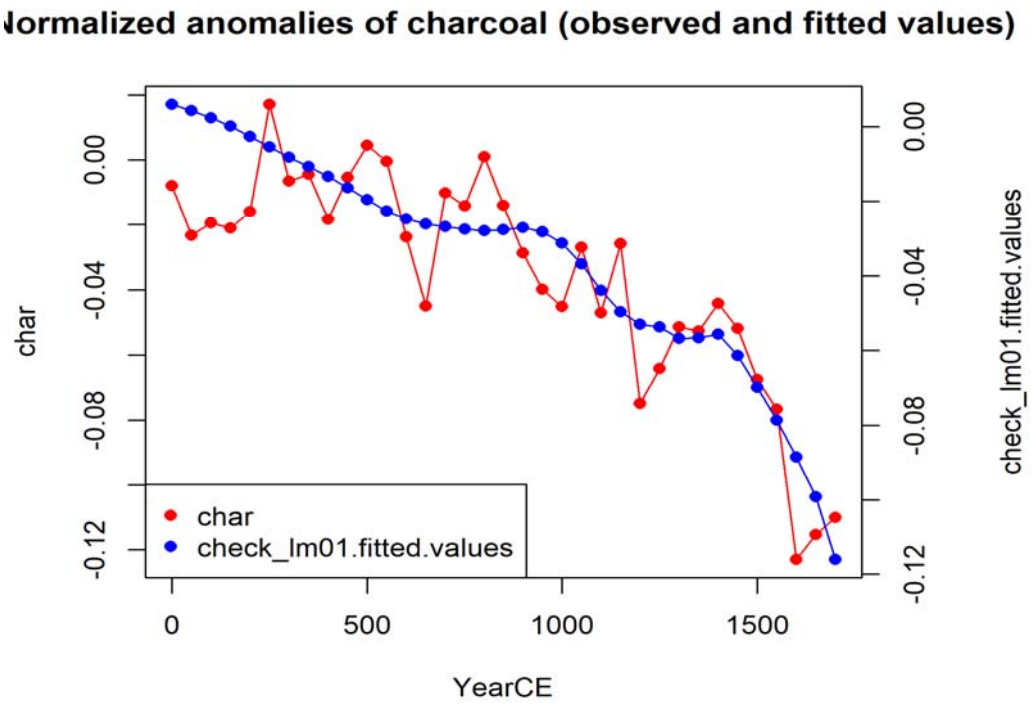

```
par <- oldpar
```

There's a bit of an arch to the residual scatter plot (upper left), but it is not large. The first order autocorrelation coefficient of the residuals appears to be barely significant (lower right). The Ljung-Box test for serial dependence of the residuals suggests, however, that overall the autocorrelations are not significant.

```
Box.test(check_lm01$residuals, lag=12, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  check_lm01$residuals
## X-squared = 16.9, df = 12, p-value = 0.1534
```

Observed and fitted values of the normalized anomalies of charcoal:

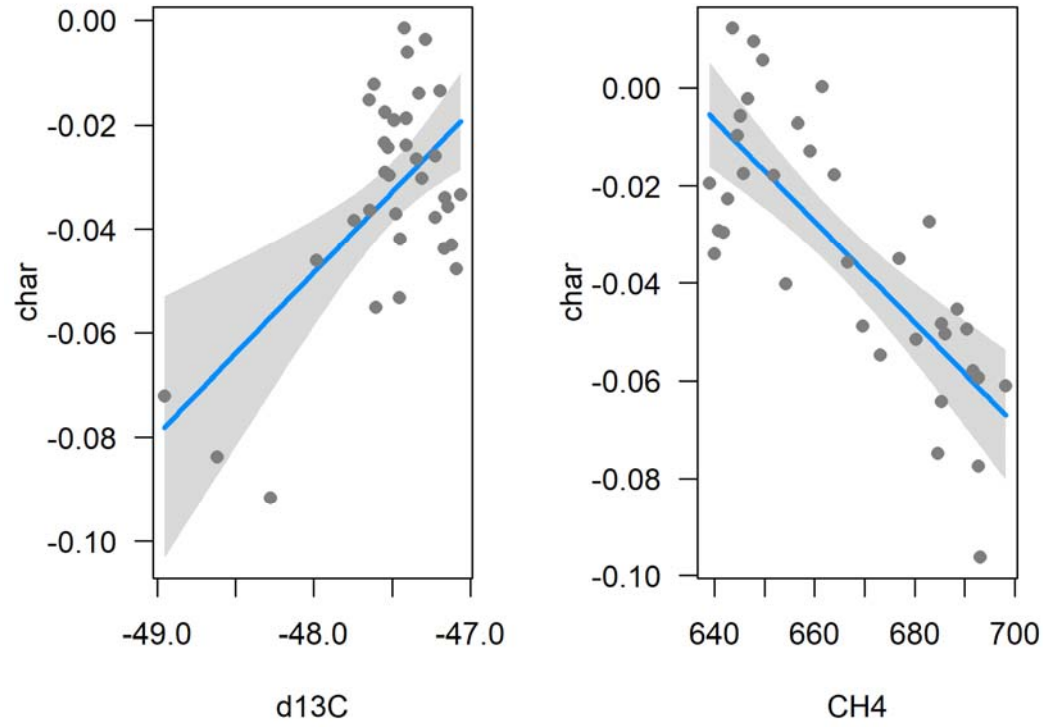
```
fitted <- data.frame(cbind(preind, check_lm01$residuals, check_lm01$fitted.values))
plotseries("Normalized anomalies of charcoal (observed and fitted values)", fitted, "YearCE", "char",
"check_lm01.fitted.values")
```




```
library(visreg)
```

Partial residual plots:

```
oldpar <- par(mfcol=c(1,2))
visreg(check_lm01, points=list(pch=16, cex=1))
```




```
par <- oldpar
```

The partial residual plots are consistent with the relationship among the individual variables.

8) Global and natural emissions vs. temperature (GFED4)

Introduction

The regressions documented here illustrate the relationship between GFED4 emissions and global- and regional-average land temperature anomalies, over the interval 2000 through 2014 CE. Although the GFED4 emissions data area available back to 1997, the emissions prior to 2000 are thought to have larger uncertainties. Carbon emissions in the GFED4 are partitioned into those from savannah, grassland and shrubland fires; boreal forest fires; temperate forest fires; deforestation fires; peatland fires, agricultural fires). The estimates of total fire emissions we used include all of these sectors except agricultural fires ("global" fires); we also estimate the total emissions from natural sources excluding deforestation and peatland fires ("natural" fires). The temperature data come from the NOAA GHCNMv2 data base, "reanomalized" to a 2000 - 2014 CE base period.

Because the data set is small, and there are some obvious points that could be regarded as outliers, we fit several models to explore that possibility.

Read and plot data

Set path and file names.

```
datapath <- "/Projects/GPWG/work/feedback/data/csv/"
dataname <- "gfed4_NOAAtmp_anm.csv"
filename <- paste(datapath, dataname , sep="")
```

Read the data.

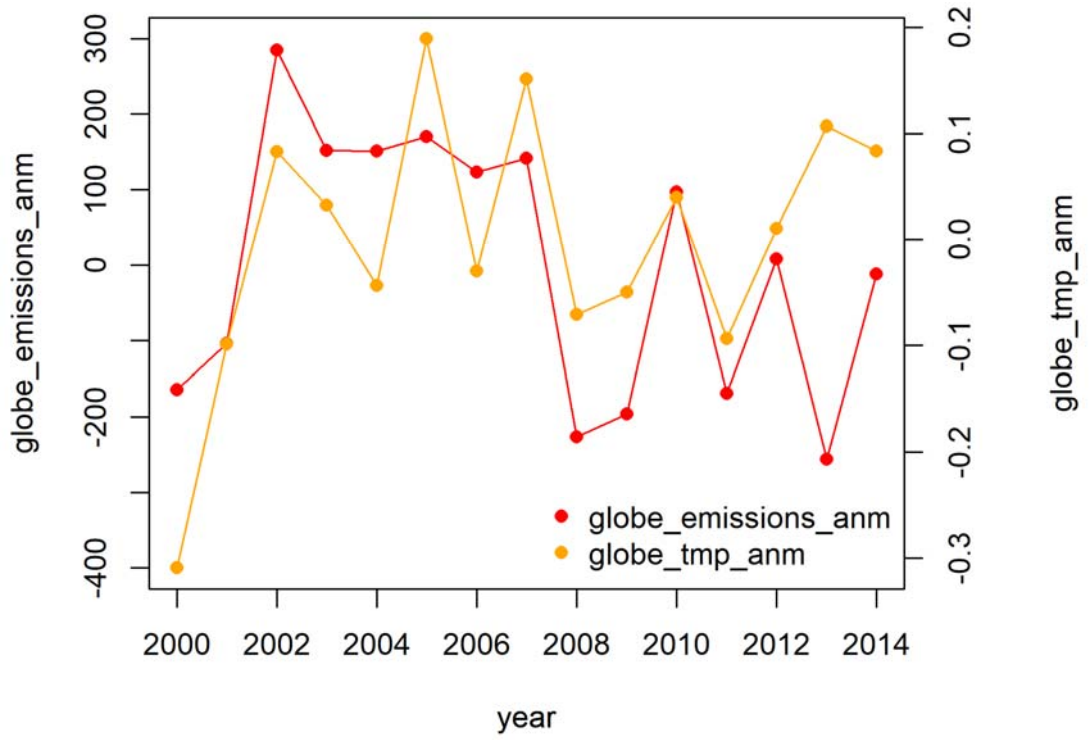
```
GFED4_tmp <- read.csv(filename)
names(GFED4_tmp)

## [1] "year"                  "globe_tmp_anm"         "globe_emissions_anm"
## [4] "natural_tmp_anm"       "natural_emissions_anm"
```

In the variable names above `anm` indicates that the values are anomalies, or differences from a long-term average.

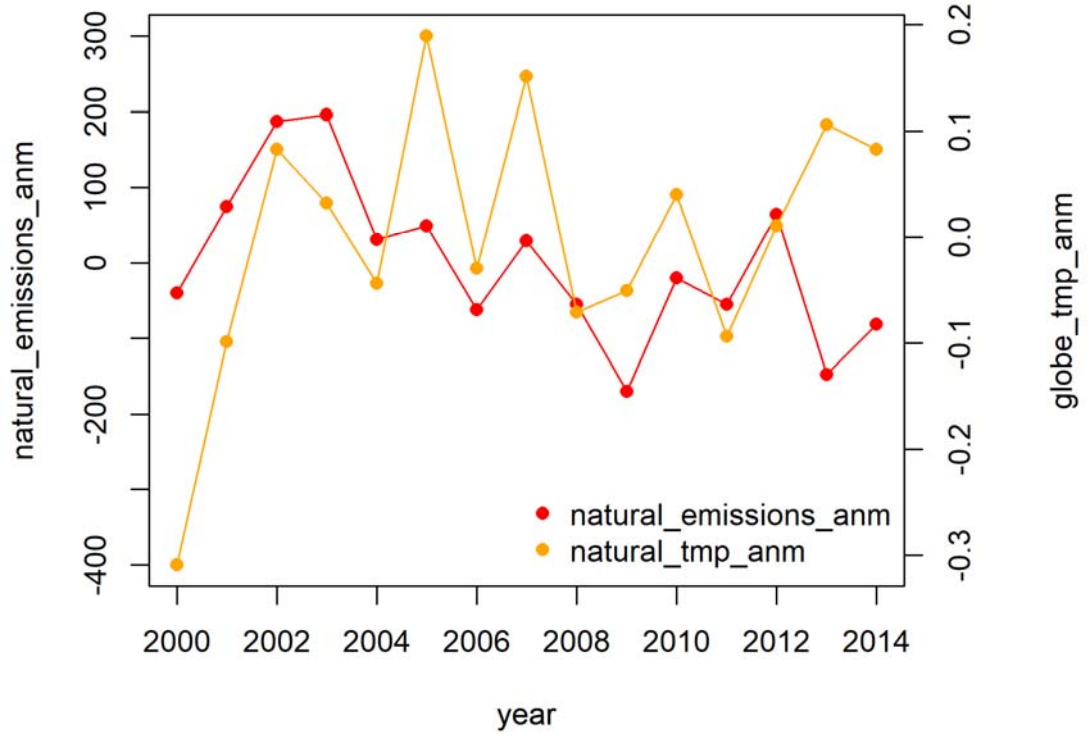
```
GFED4_tmp

##    year globe_tmp_anm globe_emissions_anm natural_tmp_anm natural_emissions_anm
## 1  2000   -0.30934667         -164.335499     -0.30946667              -40.32799
## 2  2001   -0.09904667         -104.084119     -0.09866667               75.12581
## 3  2002    0.08285333          284.764981      0.08283333              186.91011
## 4  2003    0.03245333          152.188781      0.03233333              195.90861
## 5  2004   -0.04364667          151.222381     -0.04376667               31.28921
## 6  2005    0.18965333          170.303481      0.18963333               49.66521
## 7  2006   -0.02994667          123.823181     -0.02946667              -62.27169
## 8  2007    0.15155333          141.977351      0.15143333               29.14821
## 9  2008   -0.07074667         -226.883339     -0.07116667              -55.14739
## 10 2009   -0.05014667         -196.218119     -0.05036667             -170.07599
## 11 2010    0.03975333           96.844201      0.04023333              -20.48859
## 12 2011   -0.09384667         -168.908819     -0.09386667              -55.51129
## 13 2012    0.01015333            7.713681      0.01063333               64.44711
## 14 2013    0.10685333         -256.065719      0.10663333             -147.29509
## 15 2014    0.08345333          -12.342419      0.08303333              -81.37629
```

Plot the data.

```
oldpar <- par(mar=c(5,4,4,5)+0.1)
plot(globe_emissions_anm ~ year, data=GFED4_tmp, type="o", pch=16, ylim=c(-400,300), col="red")
par(new=T)          # second plot is going to get added to first
plot(globe_tmp_anm ~ year, data=GFED4_tmp, axes=F, ylab="", type="o", pch=16, col="orange")
axis(side=4)   # add axis
mtext(side=4,line=3.8,"globe_tmp_anm")
legend("bottomright", bty="n", legend=c("globe_emissions_anm","globe_tmp_anm"), pch=c(16,16),
       col=c("red","orange"))
```

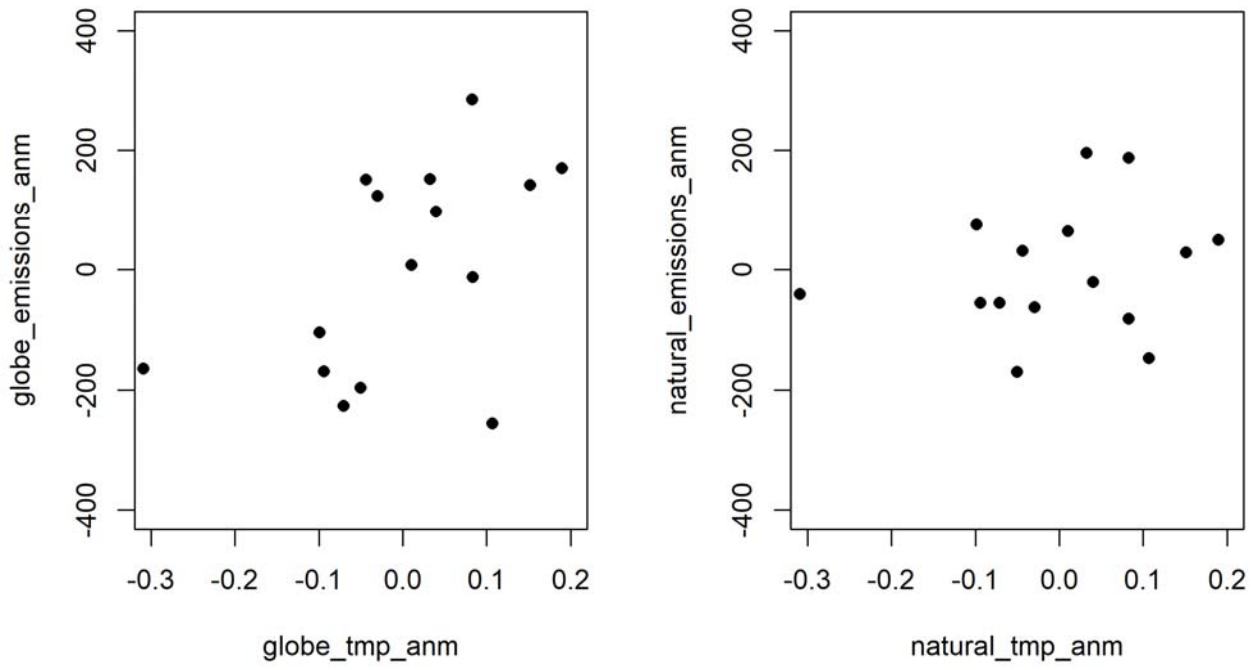



```r
plot(natural_emissions_anm ~ year, data=GFED4_tmp, type="o", pch=16, ylim=c(-400,300), col="red")
par(new=T)          # second plot is going to get added to first
plot(natural_tmp_anm ~ year, data=GFED4_tmp, axes=F, ylab="", type="o", pch=16, col="orange")
axis(side=4)   # add axis
mtext(side=4,line=3.8,"globe_tmp_anm")
legend("bottomright", bty="n", legend=c("natural_emissions_anm","natural_tmp_anm"), pch=c(16,16),
       col=c("red","orange"))
```




```r
par(oldpar)

oldpar <- par(mfrow=c(1,2))
plot(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
```

```
plot(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
```




```
par <- oldpar
```

There is an obvious outlier on both scatter plots, corresponding to the year 2000, which has the lowest value of the land-temperture anomaly. However, it turns out that this is not the most influential observation in the regressions, as will be indicated by Cook's-distance values. That most-influential observation turns out to be 2013.

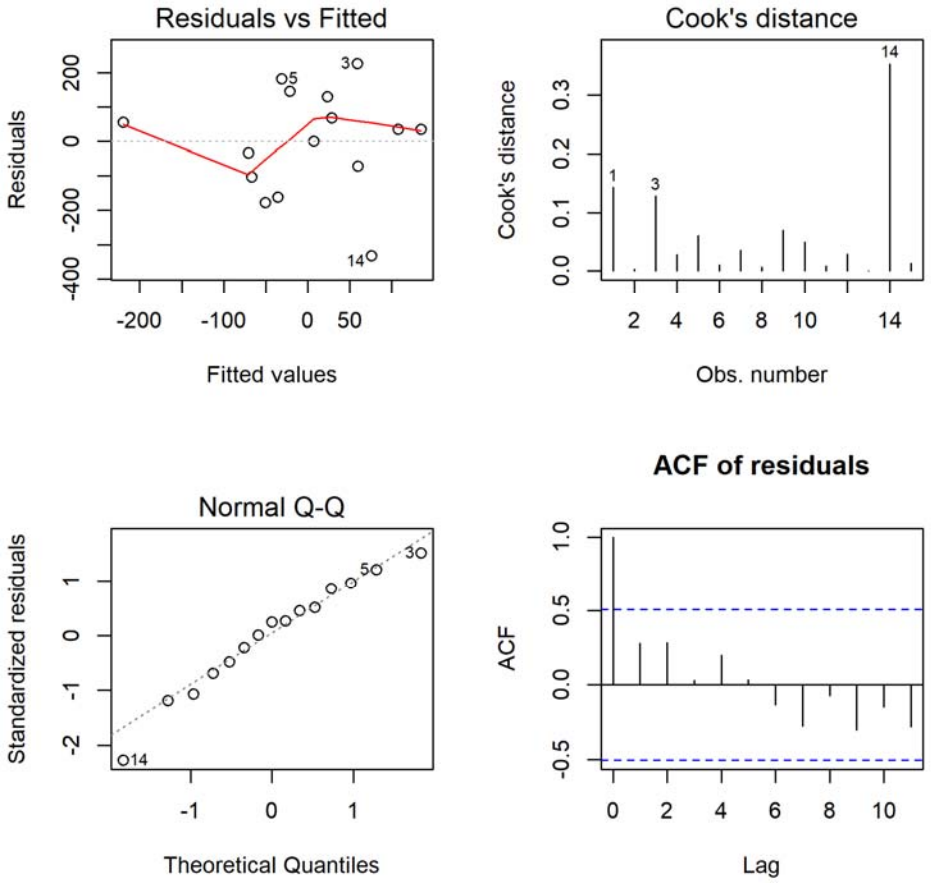
Because a reviewer expressed concern about the potential role of that point in overly influencing we explored a robust regression approach, as well as some alternative regressions in which points were explicitly removed.

Regressions

"All fires"

Regression and diagnostic plots for all fires (except agricultural fires, i.e. `global_emissions_anm`):

```
globe_lm01 <- lm(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp)
summary(globe_lm01)

##
## Call:
## lm(formula = globe_emissions_anm ~ globe_tmp_anm, data = GFED4_tmp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -332.06  -86.93   34.19   98.84  225.84
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.642e-07  4.026e+01   0.000    1.000
## globe_tmp_anm 7.112e+02  3.390e+02   2.098    0.056 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 155.9 on 13 degrees of freedom
## Multiple R-squared:  0.2529, Adjusted R-squared:  0.1955
## F-statistic: 4.401 on 1 and 13 DF,  p-value: 0.05603

oldpar <- par(mfcol=c(2,2))
plot(globe_lm01, which=c(1,2,4))
acf(globe_lm01$residuals, main="ACF of residuals", font.main=1)
```



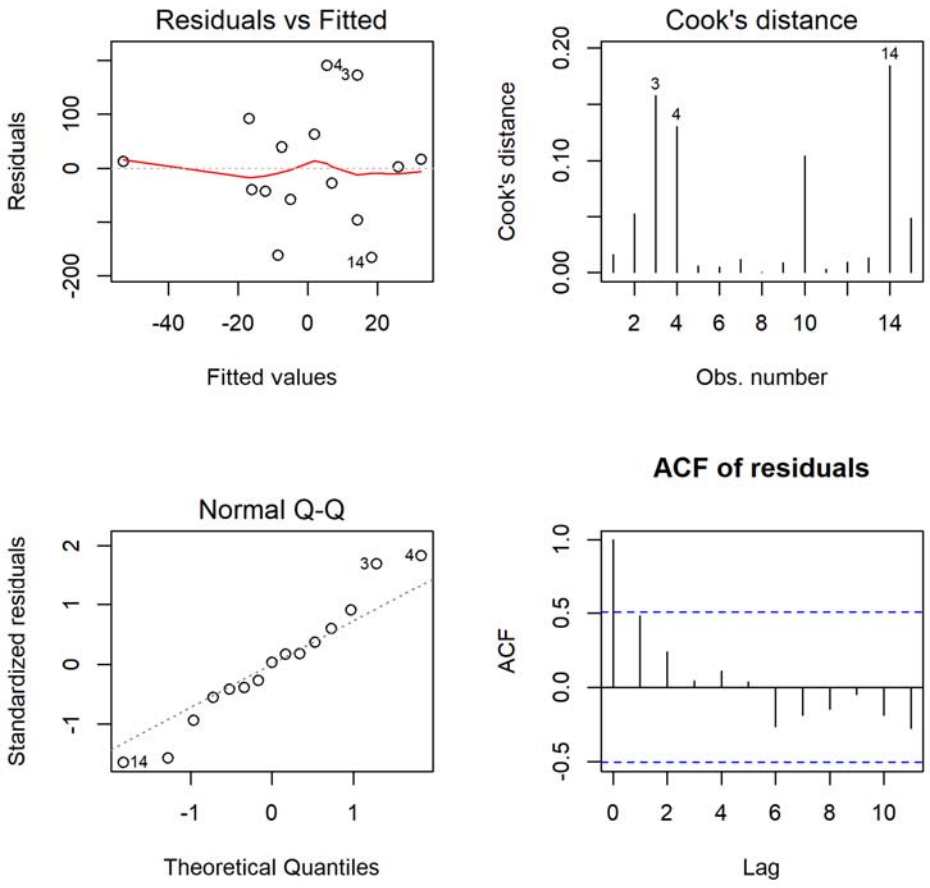

```
par <- oldpar
```

The diagnostic plots indicate that if there is an overly influential observation, it is observation number 14 (2003), as opposed to observation 1 (2000)

"Natural fires"

Regression and diagnostic plots for "natural" fires (excluding deforestation and peatland fires, `natural_emissions_anm`):

```
natural_lm01 <- lm(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp)
summary(natural_lm01)

##
## Call:
## lm(formula = natural_emissions_anm ~ natural_tmp_anm, data = GFED4_tmp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -165.587  -50.078    3.171   50.710  190.362
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.585e-08  2.778e+01   0.000    1.000
## natural_tmp_anm 1.715e+02  2.339e+02   0.733    0.476
##
## Residual standard error: 107.6 on 13 degrees of freedom
## Multiple R-squared:  0.03972,    Adjusted R-squared:  -0.03415
## F-statistic: 0.5377 on 1 and 13 DF,  p-value: 0.4764

oldpar <- par(mfcol=c(2,2))
plot(natural_lm01, which=c(1,2,4))
acf(natural_lm01$residuals, main="ACF of residuals", font.main=1)
```



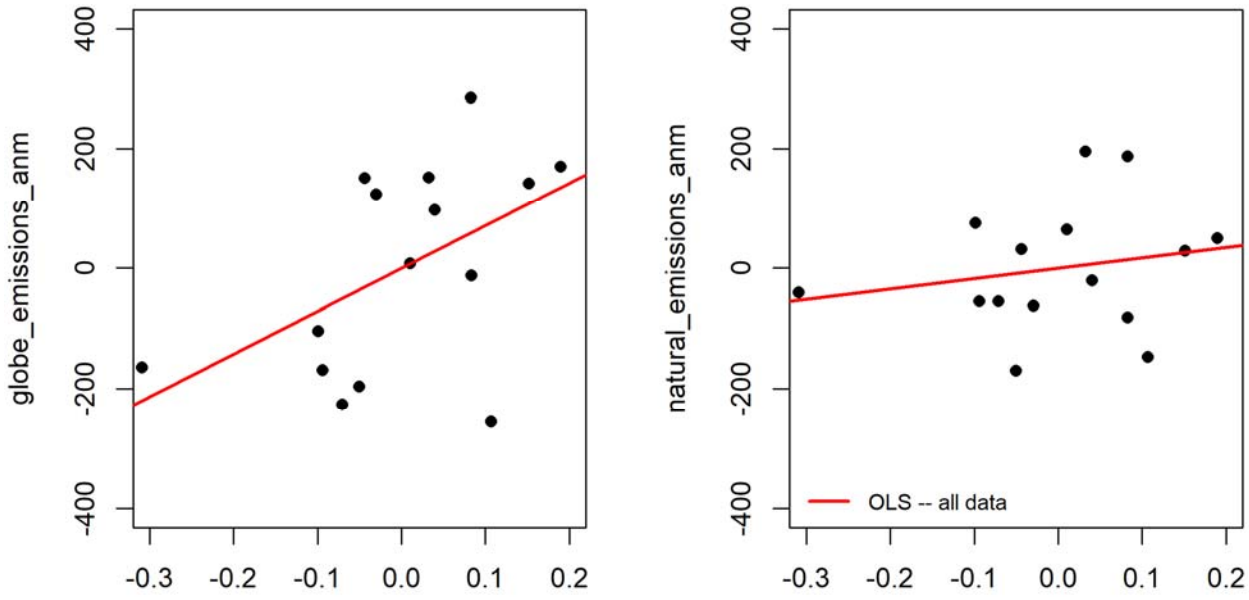

```
par <- oldpar
```

The diagnostic plots again indicate that observation 14 (2013) is influential, along with observations 3 and 4 (2002 and 2003).

Plot results

Plot the scatter plots and regression lines:

```
oldpar <- par(mfrow=c(1,2))
plot(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(globe_lm01, lwd=2, col="red")
plot(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(natural_lm01, lwd=2, col="red")
legend("bottomleft", bty="n", legend=c("OLS -- all data"),
       cex=0.8, lwd=2, col=c("red"))
```




```
par <- oldpar
```

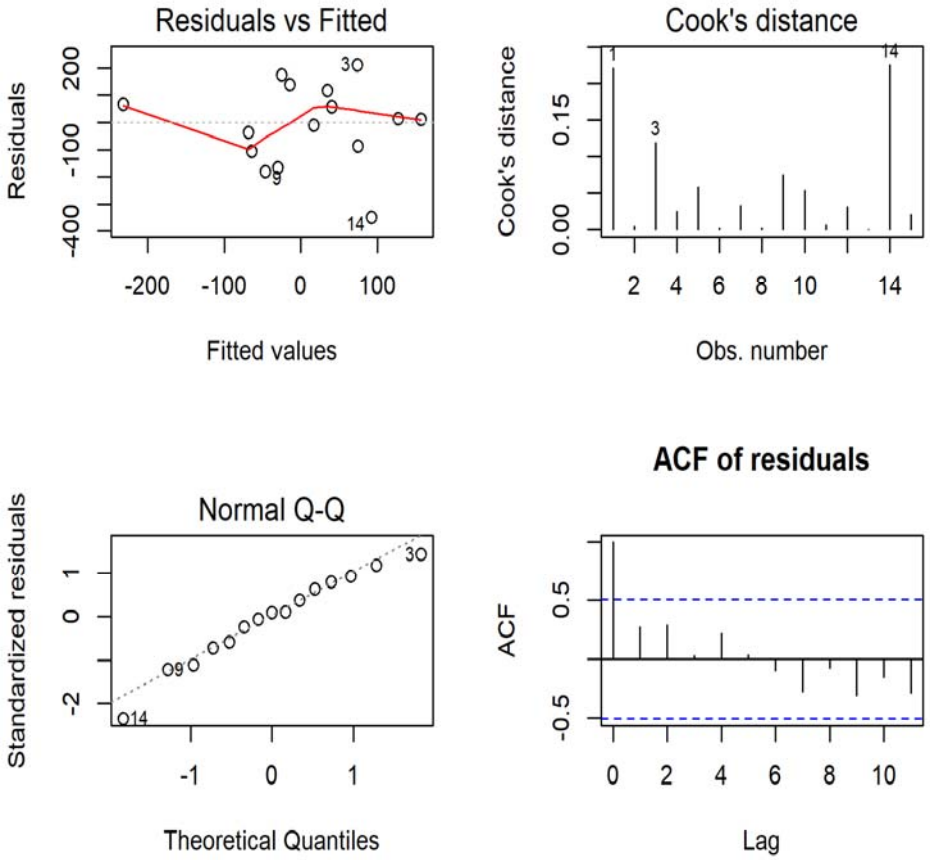
Robust regressions

There are two alternative approachs for addressing the possibility that some points are overly influential: 1) robust regression, in which the influential points are downweighed, and 2) ordinary regressions in which the influential points are explicitly removed. Robust regression is implemented by the `rlm()` function in the `MASS` library.

"All fires"

```
library(MASS)
globe_rlm01 <- rlm(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp, init="lts")

## Warning in lqs.default(structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, : only 105
## sets, so all sets will be tried

summary(globe_rlm01)

##
## Call: rlm(formula = globe_emissions_anm ~ globe_tmp_anm, data = GFED4_tmp,
##     init = "lts")
## Residuals:
##     Min      1Q  Median      3Q     Max
## -348.72  -95.76   12.98   92.87  210.86
##
## Coefficients:
##               Value    Std. Error t value
## (Intercept)     9.1901  41.3414     0.2223
## globe_tmp_anm 781.0763 348.1128     2.2437
##
## Residual standard error: 155.4 on 13 degrees of freedom
```

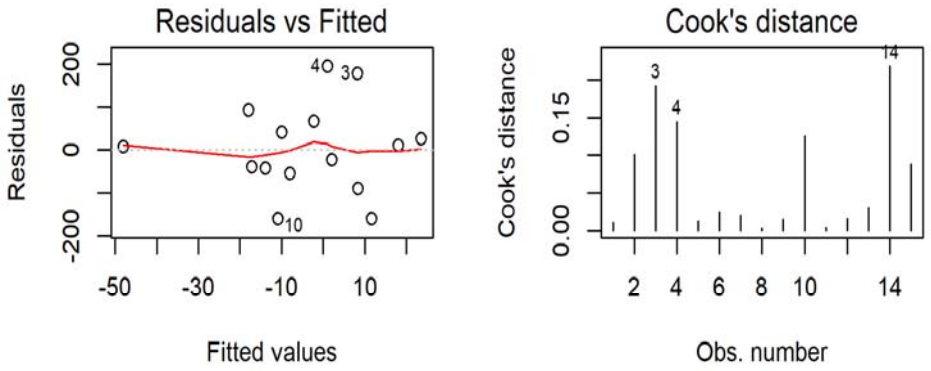
```
oldpar <- par(mfcol=c(2,2))
plot(globe_rlm01, which=c(1,2,4))
acf(globe_rlm01$residuals, main="ACF of residuals", font.main=1)
```

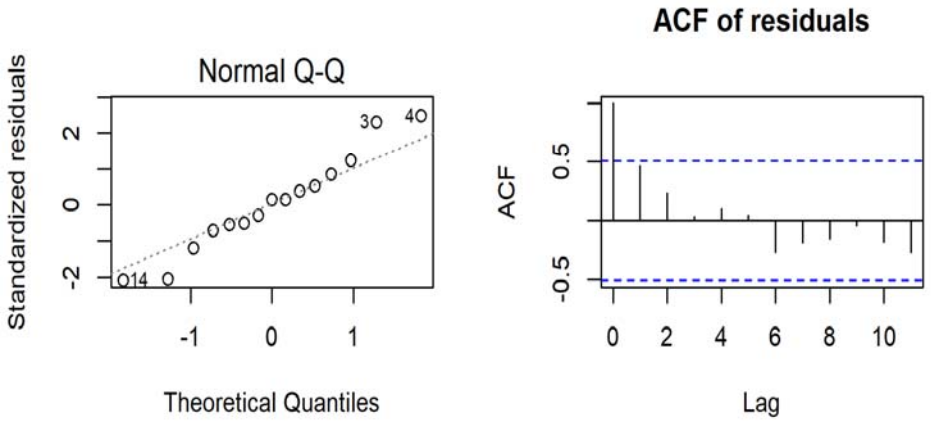



```
par <- oldpar
```

"Natural fires"

```
natural_rlm01 <- rlm(natural_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp, init="lts")

## Warning in lqs.default(structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, : only 105
## sets, so all sets will be tried

summary(natural_rlm01)

##
## Call: rlm(formula = natural_emissions_anm ~ globe_tmp_anm, data = GFED4_tmp,
##     init = "lts")
## Residuals:
##      Min       1Q   Median       3Q      Max
## -159.173  -47.778    7.788   53.976  194.953
##
## Coefficients:
##               Value    Std. Error t value
## (Intercept)    -3.7032  28.5399    -0.1298
## globe_tmp_anm 143.5702 240.3179     0.5974
##
## Residual standard error: 80.46 on 13 degrees of freedom

oldpar <- par(mfcol=c(2,2))
plot(natural_rlm01, which=c(1,2,4))
acf(natural_rlm01$residuals, main="ACF of residuals", font.main=1)
```



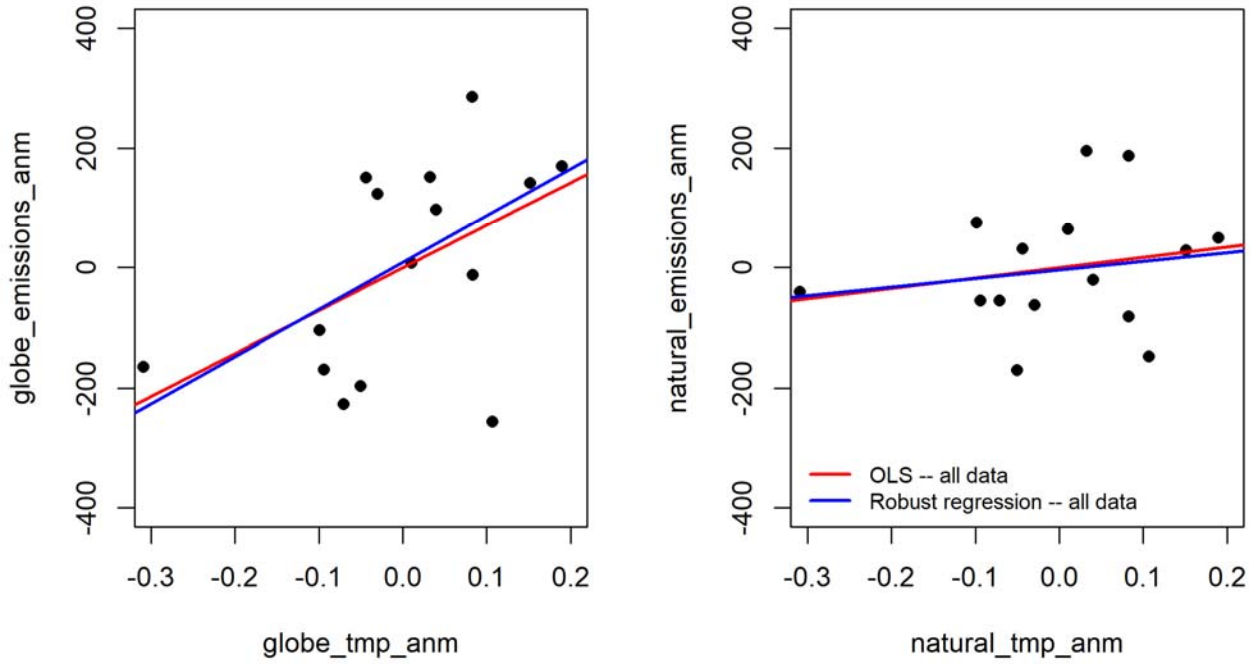



```
par <- oldpar
```

Plot robust regression results

```
oldpar <- par(mfrow=c(1,2))
plot(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(globe_lm01, lwd=2, col="red")
abline(globe_rlm01, lwd=2, col="blue")
```

```
plot(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(natural_lm01, lwd=2, col="red")
abline(natural_rlm01, lwd=2, col="blue")
legend("bottomleft", bty="n", legend=c("OLS -- all data", "Robust regression -- all data"),
       cex=0.8, lwd=2, col=c("red", "blue"))
```




```
par <- oldpar
```

Although fairly conservative choices were made in the design of this regression, the robust regression results do not differ much from the standard OLS results.

Explicit outlier removal

The data set here is quite small (`n = 14`), and so the removal of individual points that may be unusal or influential necessarily requires strong support from other, external to the regression analysis, sources. Because we have already noted that there are larger uncertainties in the GFED4 data prior to 2000, removing that point might be warranted.

Remove 2000

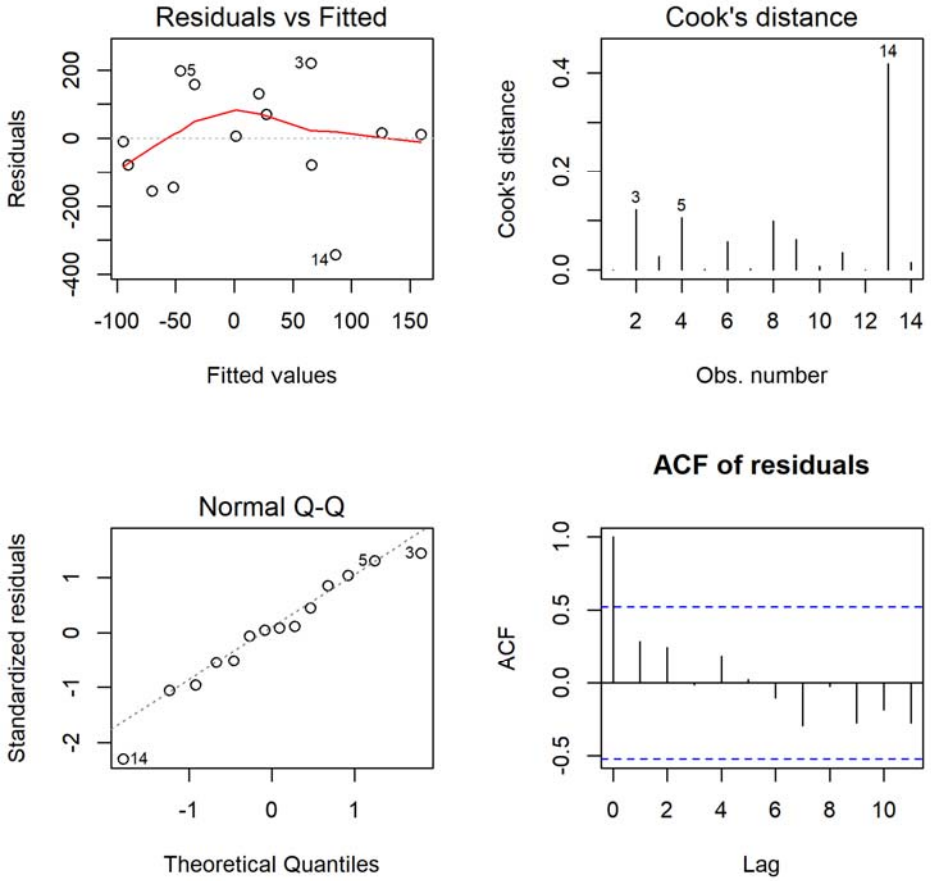
Remove the observation for the year 2000 from the data set.

```
GFED4_tmp_no2000 <- GFED4_tmp[GFED4_tmp$year != 2000, ]
```

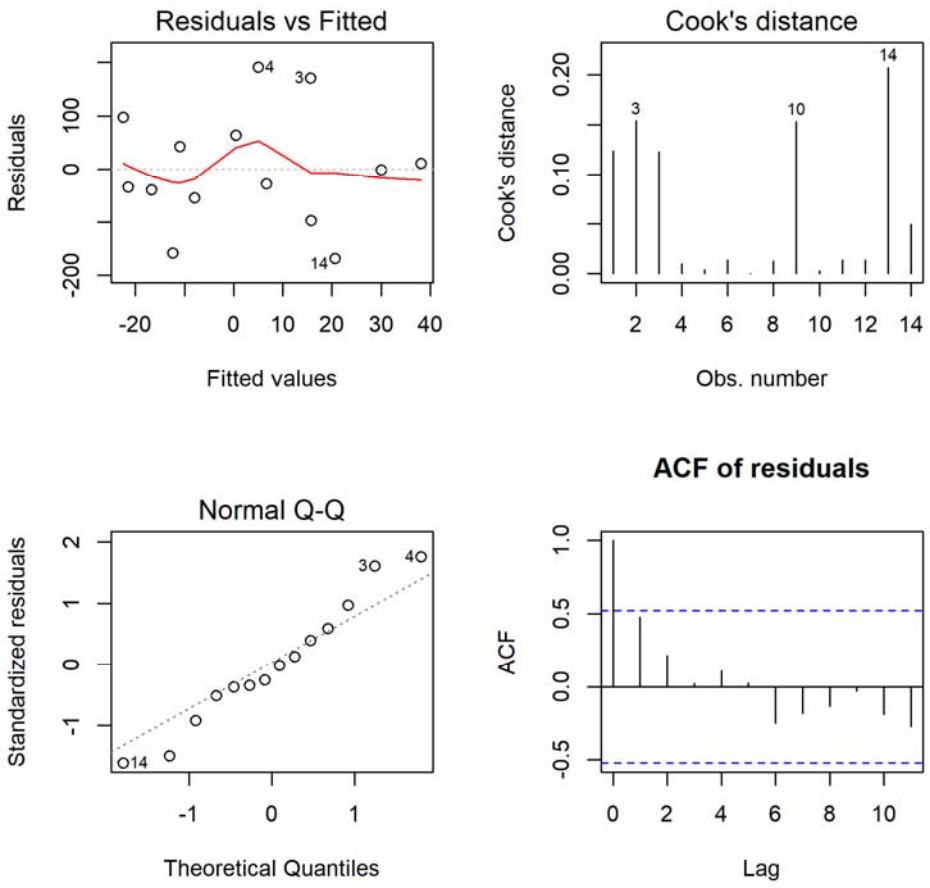
New regressions:

```
globe_lm02 <- lm(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp_no2000)
summary(globe_lm02)

##
## Call:
## lm(formula = globe_emissions_anm ~ globe_tmp_anm, data = GFED4_tmp_no2000)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -342.43  -78.44    8.76  115.89  219.53
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.718     44.258  -0.174   0.8645
## globe_tmp_anm  880.516    486.509   1.810   0.0954 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

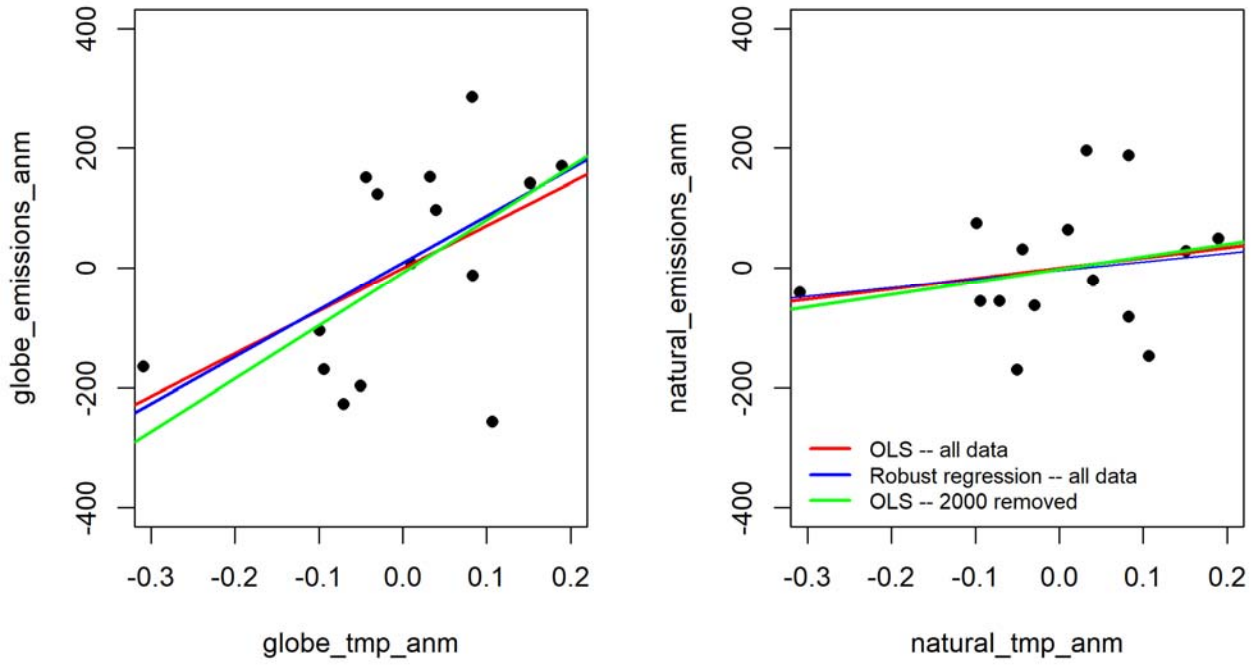
```
##
## Residual standard error: 160.6 on 12 degrees of freedom
## Multiple R-squared:  0.2144, Adjusted R-squared:  0.149
## F-statistic: 3.276 on 1 and 12 DF,  p-value: 0.09541

oldpar <- par(mfcol=c(2,2))
plot(globe_lm02, which=c(1,2,4))
acf(globe_lm02$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar

natural_lm02 <- lm(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp_no2000)
summary(natural_lm02)

##
## Call:
## lm(formula = natural_emissions_anm ~ natural_tmp_anm, data = GFED4_tmp_no2000)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -167.96  -50.33  -14.06   58.55  190.88
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -1.77      30.82  -0.057    0.955
## natural_tmp_anm   210.40     338.97   0.621    0.546
##
## Residual standard error: 111.9 on 12 degrees of freedom
## Multiple R-squared:  0.03111,    Adjusted R-squared:  -0.04963
## F-statistic: 0.3853 on 1 and 12 DF,  p-value: 0.5464

oldpar <- par(mfcol=c(2,2))
plot(natural_lm02, which=c(1,2,4))
acf(natural_lm02$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar

oldpar <- par(mfrow=c(1,2))
plot(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(globe_lm01, lwd=2, col="red")
abline(globe_rlm01, lwd=2, col="blue")
abline(globe_lm02, lwd=2, col="green")
plot(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(natural_lm01, lwd=2, col="red")
abline(natural_rlm01, lwd-2, col="blue")
abline(natural_lm02, lwd=2, col="green")
legend("bottomleft", bty="n", legend=c("OLS -- all data","Robust regression -- all data", "OLS -- 2000
removed"),
       cex=0.8, lwd=2, col=c("red", "blue", "green"))
```




```
par <- oldpar
```

As can be observed, the observation representing the year 2000, is in fact not that influential, which could have been anticipated from the robust regerssion results.

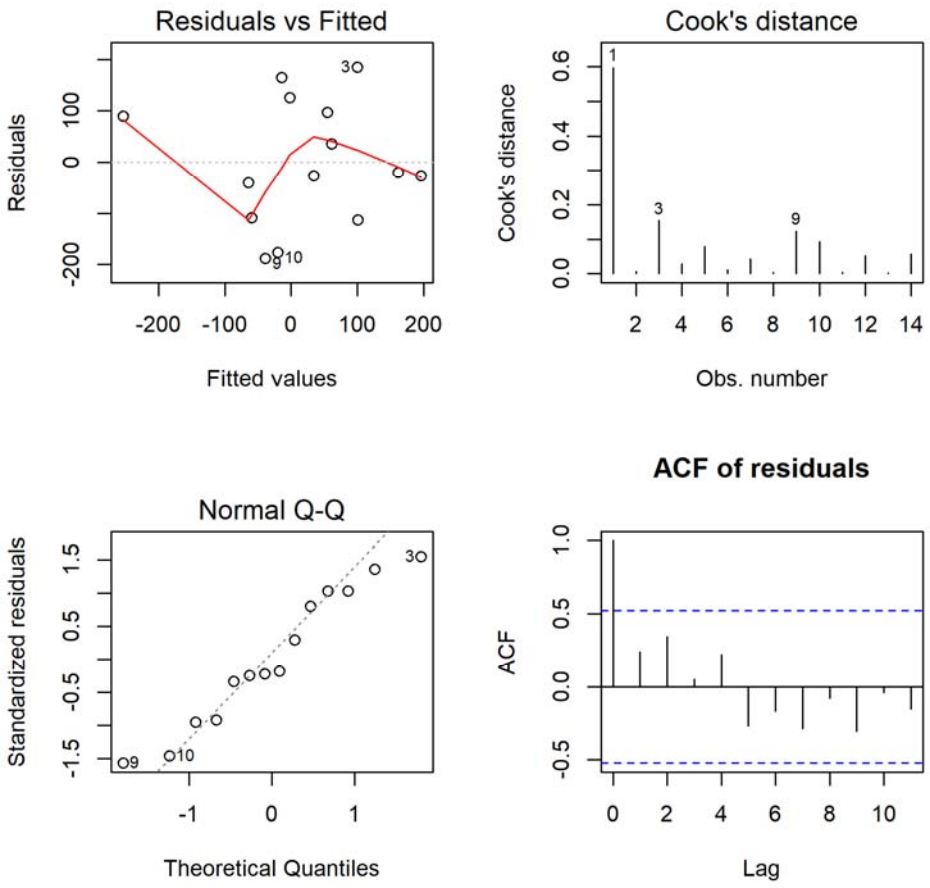
Remove 2013

Here, we remove what is clearly the most influential point, representing the year 2013. There is no real support for doing so, other than that point is influential.

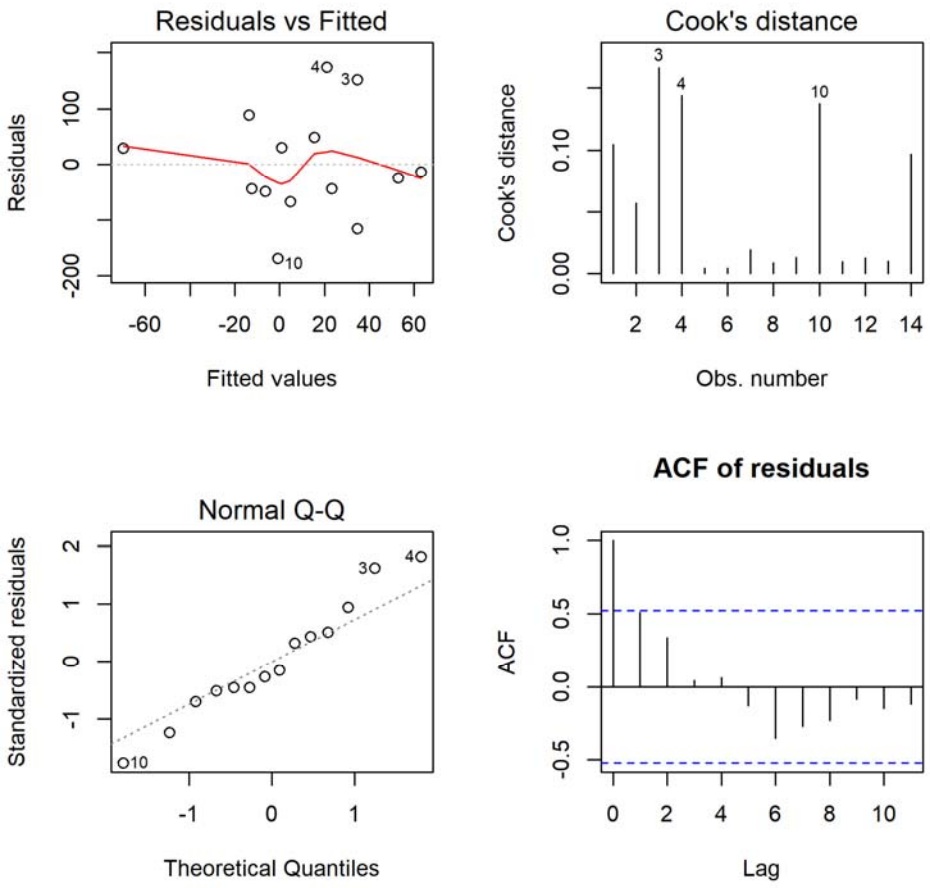
```
GFED4_tmp_no2013 <- GFED4_tmp[GFED4_tmp$year != 2013, ]
```

The regressions with 2013 omited:

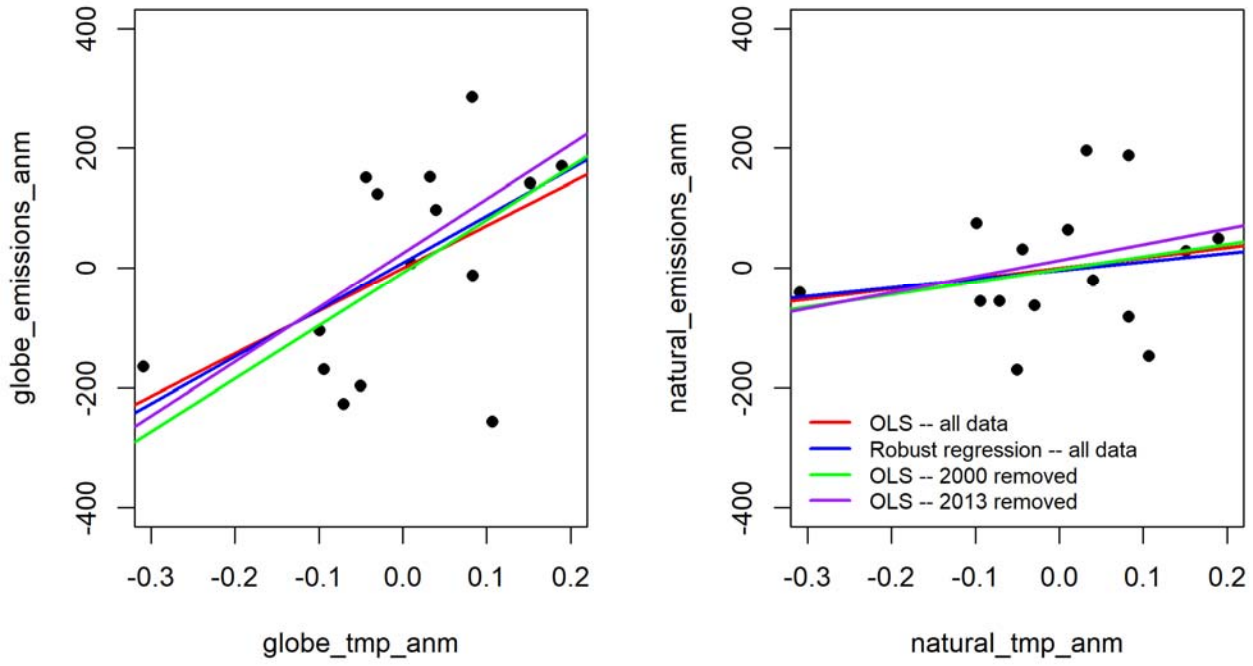
```
globe_lm03 <- lm(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp_no2013)
summary(globe_lm03)

##
## Call:
## lm(formula = globe_emissions_anm ~ globe_tmp_anm, data = GFED4_tmp_no2013)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -188.25  -92.06  -22.91   95.68  184.86
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      25.17      33.76   0.746  0.47023
## globe_tmp_anm   901.96     282.37   3.194  0.00771 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 126.1 on 12 degrees of freedom
## Multiple R-squared:  0.4595, Adjusted R-squared:  0.4145
## F-statistic:  10.2 on 1 and 12 DF,  p-value: 0.007715

oldpar <- par(mfcol=c(2,2))
plot(globe_lm03, which=c(1,2,4))
acf(globe_lm03$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar

natural_lm03 <- lm(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp_no2013)
summary(natural_lm03)

##
## Call:
## lm(formula = natural_emissions_anm ~ natural_tmp_anm, data = GFED4_tmp_no2013)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -169.21  -47.49  -18.58   44.40  174.74
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        12.55      26.70   0.470    0.647
## natural_tmp_anm   266.46     223.34   1.193    0.256
##
## Residual standard error: 99.71 on 12 degrees of freedom
## Multiple R-squared:  0.106,  Adjusted R-squared:  0.03155
## F-statistic: 1.423 on 1 and 12 DF,  p-value: 0.2559

oldpar <- par(mfcol=c(2,2))
plot(natural_lm03, which=c(1,2,4))
acf(natural_lm03$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar

oldpar <- par(mfrow=c(1,2))
plot(globe_emissions_anm ~ globe_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(globe_lm01, lwd=2, col="red")
abline(globe_rlm01, lwd=2, col="blue")
abline(globe_lm02, lwd=2, col="green")
abline(globe_lm03, lwd=2, col="purple")
plot(natural_emissions_anm ~ natural_tmp_anm, data=GFED4_tmp,
     ylim=c(-400, 400), xlim=c(-0.3, 0.2), pch=16)
abline(natural_lm01, lwd=2, col="red")
abline(natural_rlm01, lwd=2, col="blue")
abline(natural_lm02, lwd=2, col="green")
abline(natural_lm03, lwd=2, col="purple")
legend("bottomleft", bty="n", legend=c("OLS -- all data","Robust regression -- all data", "OLS -- 2000
removed",
  "OLS -- 2013 removed"), cex=0.8, lwd=2, col=c("red", "blue", "green", "purple"))
```




```
par <- oldpar
```

Once again, there is little overall difference in the OLS regression that includes all data and regressions produced using other approaches.

Conclusions

There is little support for using an alternative (to the original OLS fits using all observations) model to represent the post-2000 CE relationships between global average temperature and emissions. The fact that there is little practical difference between a "plain" OLS regression using all data, and either a robust regression, or ones with outliers or overly influential points removed implies that the simple regression is relatively robust, despite the small data-set size.

9) Mann et al. (2009) temperature data

Land vs. Land + Ocean Regressions

The goal here was to compare the temperature trends in the Mann et al. (2009) data, stratified by land and ocean.

Read the data

The data were obtained from the text file `allproxyfieldrecon.dat` in the Supporting Online Material of Mann et al. (2009) Science 326:1256-1260 DOI: 10.1126/science.1177303. Area-weighted averages were calculated for the the globe and hemispheres, separately for land and ocean grid points. Land points do not include ice-covered points (on a 5-deg grid). "All" points do include ice-covered points.

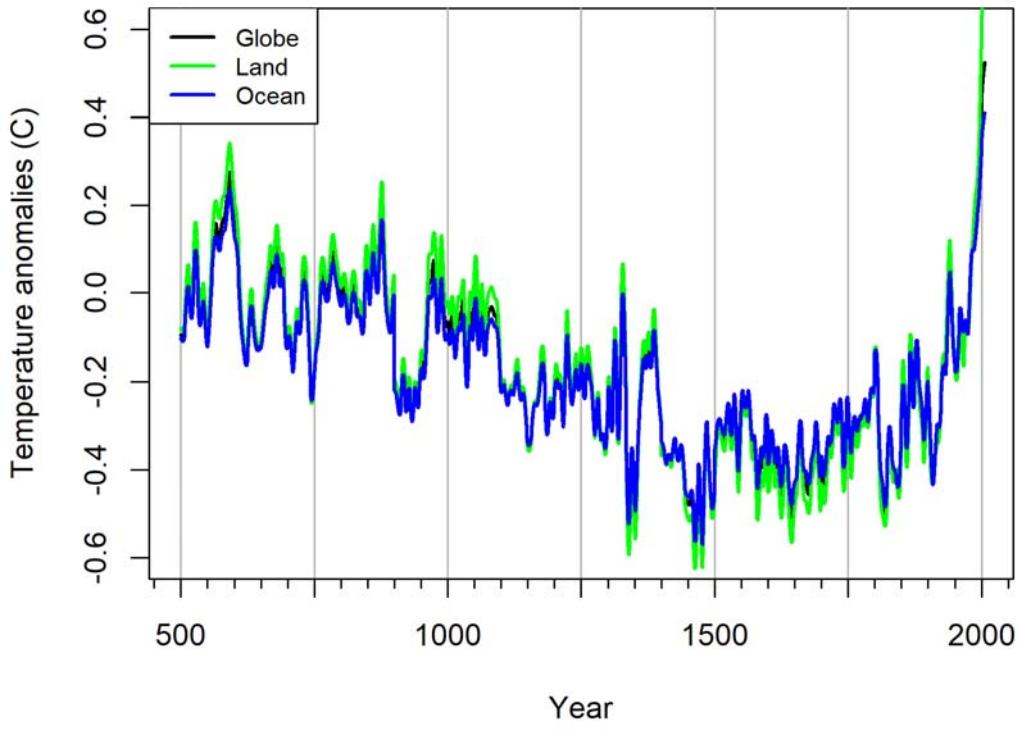
```
# compare land and ocean temperatures
# read data
csvfile <-"/Projects/GPWG/work/feedback/data/sources/Mann-etal-2009-aaves.csv"
aaves <- read.csv(csvfile)
names(aaves)
```

```
##  [1] "Year"             "Globe_land"       "Nhemis_land"      "Shemis_land"
##  [5] "Americas_land"    "NonAmericas_land" "Globe_ocean"      "Nhemis_ocean"
##  [9] "Shemis_ocean"     "Americas_ocean"   "NonAmericas_ocea" "Globe_all"
## [13] "Nhemis_all"       "Shemis_all"       "Americas_all"     "NonAmericas_all"
```

```
attach(aaves)
```

Time-series plots of the global averages:

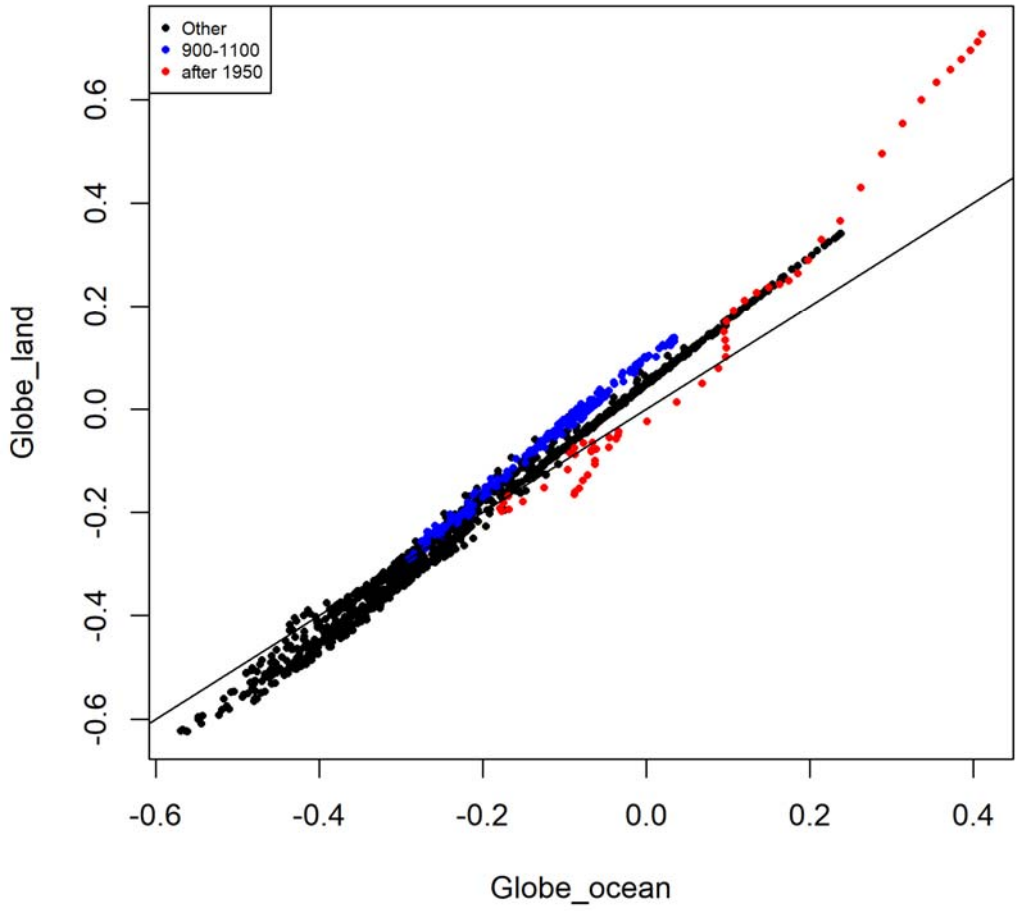
```
xmin <- 500; xmax <- 2000; ymin <- -0.6; ymax <- 0.6
xlim <- c(xmin, xmax); ylim <- c(ymin, ymax); xminortick <- 50
plot(NULL, NULL, ylim=ylim, xlim=xlim, type="n", ylab="Temperature anomalies (C)", xlab="Year")
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tck=1.0, fg="gray70")
axis(side = 1, at = seq(xmin-xminortick, xmax+xminortick, by = xminortick), labels = FALSE, tcl = -.25)
axis(side = 1, at = seq(xmin, xmax, by = xminortick*5), labels = FALSE, tcl = -.5)
lines(Year, Globe_all, col="black", type="l", lwd=2, cex=0.5)
lines(Year, Globe_land, col="green", type="l", lwd=2, cex=0.5)
lines(Year, Globe_ocean, col="blue", type="l", lwd=2, cex=0.5)
legend("topleft", legend=c("Globe", "Land", "Ocean"), lwd=2,
       col=c("black", "green", "blue"), cex=0.8)
```



Land vs. land + ocean -- 500 - 1950 CE

Before doing a regression of land (not including ice-covered points) as the response vs. globe as the predictor, excluding data after 1950 CE in the fit, plot the data:

```
cut_Year <- 1950
plot(Globe_land ~ Globe_ocean, type="n")
points(Globe_land ~ Globe_ocean, pch=16, cex=0.6, col="black")
points(Globe_land[Year >= cut_Year] ~ Globe_ocean[Year >= cut_Year],
       pch=16, cex=0.6, col="red")
points(Globe_land[Year >= 900 & Year <= 1100] ~ Globe_ocean[Year >= 900 & Year <= 1100],
       pch=16, cex=0.6, col="blue")
abline(0,1)
legend("topleft", legend=c("Other", "900-1100", "after 1950"), pch=16, cex=0.6,
       col=c("black", "blue", "red"))
```




```
# text(Globe_land ~ Globe_ocean, label=Year, cex=0.2)
```

As might be expected, the land and ocean data diverge after 1950 CE, and there seem to be two distinct patterns in the land-ocean relationship, one for data between 900 and 1100 CE, and the other for other times.

OLS regression:

```
lm_01 <- lm(Globe_land[Year <= cut_Year] ~ Globe_all[Year <= cut_Year])
summary(lm_01)

##
## Call:
## lm(formula = Globe_land[Year <= cut_Year] ~ Globe_all[Year <=
##     cut_Year])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.026209 -0.007937 -0.004199  0.009348  0.041929
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.0356983  0.0004551   78.44   <2e-16 ***
## Globe_all[Year <= cut_Year] 1.1491131  0.0017817  644.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01163 on 1449 degrees of freedom
## Multiple R-squared:  0.9965, Adjusted R-squared:  0.9965
## F-statistic: 4.16e+05 on 1 and 1449 DF,  p-value: < 2.2e-16
```

There is no clear distinction between variables as to which might be the response, and which the predictor, so is more appropriate to fit a reduced major axis regression, using the `lmodel2` package:

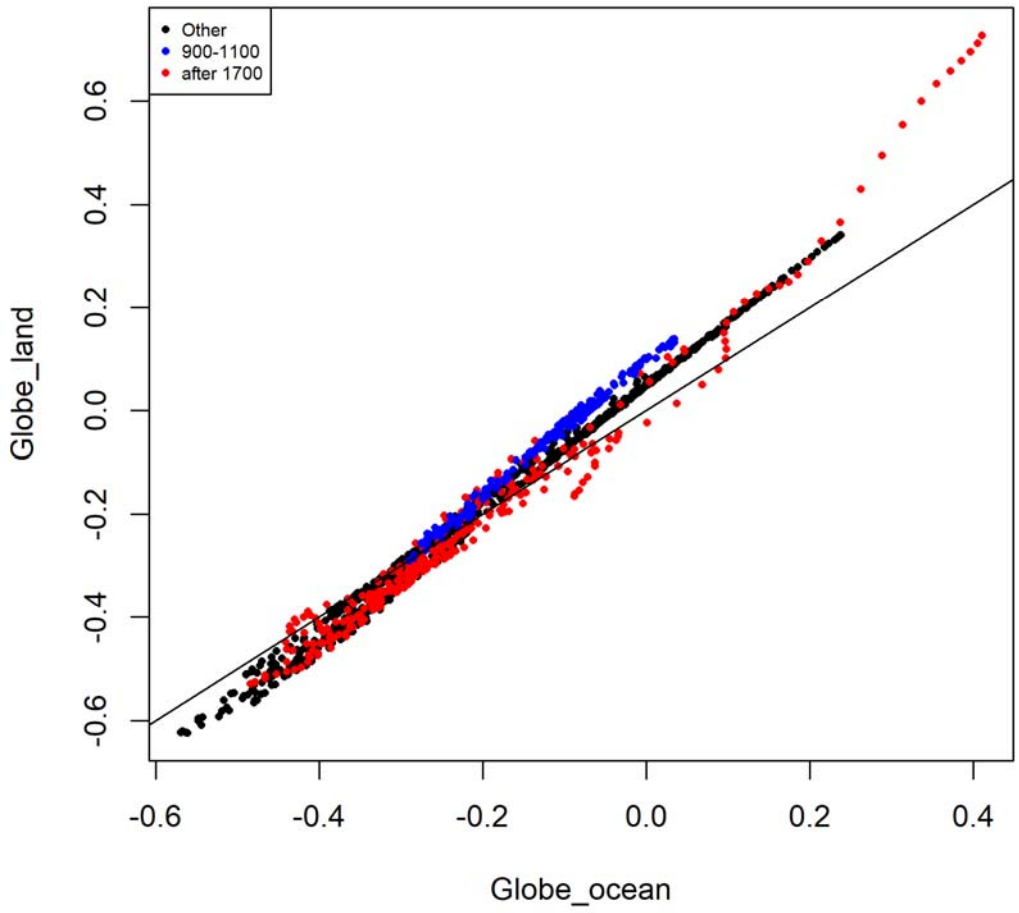
```
library(lmodel2)
lmod2_01 <- lmodel2(Globe_land[Year <= cut_Year] ~ Globe_all[Year <= cut_Year])
lmod2_01

##
## Model II regression
##
## Call: lmodel2(formula = Globe_land[Year <= cut_Year] ~ Globe_all[Year <=
## cut_Year])
##
## n = 1451   r = 0.9982629   r-square = 0.9965287
## Parametric P-values:   2-tailed = 0    1-tailed = 0
## Angle between the two OLS regression lines = 0.09863928 degrees
##
## Regression results
##   Method  Intercept    Slope Angle (degrees) P-perm (1-tailed)
## 1    OLS 0.03569825 1.149113        48.96902                NA
## 2     MA 0.03612997 1.151393        49.02525                NA
## 3    SMA 0.03607693 1.151113        49.01835                NA
##
## Confidence intervals
##   Method 2.5%-Intercept 97.5%-Intercept 2.5%-Slope 97.5%-Slope
## 1    OLS     0.03480552      0.03659098   1.145618    1.152608
## 2     MA     0.03546796      0.03679428   1.147897    1.154901
## 3    SMA     0.03541609      0.03673978   1.147623    1.154613
##
## Eigenvalues: 0.06831068 5.822169e-05
##
## H statistic used for computing C.I. of MA: 2.267203e-06
```

The land:globe ratio of temperature anomalies in these data is 1.151 (i.e the "MA" method above; note that in the `lmodel2` package, "RMA" stands for "ranged major axis")

Land vs. globe -- 500 - 1700 CE

Regression of land (not including ice-covered points) as the response vs. globe as the predictor, excluding data after 1701 CE.

```
cut_Year <- 1700
plot(Globe_land ~ Globe_ocean, type="n")
points(Globe_land ~ Globe_ocean, pch=16, cex=0.6, col="black")
points(Globe_land[Year >= cut_Year] ~ Globe_ocean[Year >= cut_Year],
       pch=16, cex=0.6, col="red")
points(Globe_land[Year >= 900 & Year <= 1100] ~ Globe_ocean[Year >= 900 & Year <= 1100],
       pch=16, cex=0.6, col="blue")
abline(0,1)
legend("topleft", legend=c("Other", "900-1100", "after 1700"), pch=16, cex=0.6,
       col=c("black", "blue", "red"))
```




```
# text(Globe_land ~ Globe_ocean, label=Year, cex=0.2)
```

OLS regression:

```
lm_02 <- lm(Globe_land[Year <= cut_Year] ~ Globe_all[Year <= cut_Year])
summary(lm_02)

##
## Call:
## lm(formula = Globe_land[Year <= cut_Year] ~ Globe_all[Year <=
##     cut_Year])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.024513 -0.007825 -0.003831  0.008953  0.026003
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.0360694  0.0004328   83.33   <2e-16 ***
## Globe_all[Year <= cut_Year] 1.1463298  0.0017749  645.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01078 on 1199 degrees of freedom
## Multiple R-squared:  0.9971, Adjusted R-squared:  0.9971
## F-statistic: 4.171e+05 on 1 and 1199 DF,  p-value: < 2.2e-16
```

Major-axis regression:

```
lmod2_02 <- lmodel2(Globe_land[Year <= cut_Year] ~ Globe_all[Year <= cut_Year])
lmod2_02

##
## Model II regression
##
## Call: lmodel2(formula = Globe_land[Year <= cut_Year] ~ Globe_all[Year <=
```

```
## cut_Year])
##
## n = 1201   r = 0.9985659   r-square = 0.9971338
## Parametric P-values:   2-tailed = 0    1-tailed = 0
## Angle between the two OLS regression lines = 0.08145194 degrees
##
## Regression results
##   Method  Intercept    Slope Angle (degrees) P-perm (1-tailed)
## 1    OLS 0.03606939 1.146330        48.90021               NA
## 2     MA 0.03638689 1.148202        48.94652               NA
## 3    SMA 0.03634855 1.147976        48.94094               NA
##
## Confidence intervals
##   Method 2.5%-Intercept 97.5%-Intercept 2.5%-Slope 97.5%-Slope
## 1    OLS     0.03522021      0.03691857   1.142848    1.149812
## 2     MA     0.03579647      0.03697934   1.144720    1.151696
## 3    SMA     0.03575898      0.03693991   1.144499    1.151464
##
## Eigenvalues: 0.07120177 5.013092e-05
##
## H statistic used for computing C.I. of MA: 2.263502e-06
```

Virtually the same ratio (1.148) obtains for 500-1700 CE data as for 500-1950 CE (1.151).

10) Normalized anomalies of charcoal vs. Mann et al. temperature

Introduction

This is a charcoal vs. temperature regression, using data from "binboot"" fits to "normans" (normalized anomalies) of charcoal fromGCDv3 (version v3i, the public release) and the Mann et al. (2009) temperature data, summarized the same way, expressed as anomalies from a 1961-1990 base period, land areas. These data are higher resolution that those used in section 6 above, and employ a differnt approach for developing composite or smooth curves, and therefore provide another perspecive on the long-term charcoal vs. temperature relationship.

The specific data are the bin-average values for each data set, produced at 20-yr intervals, and spanning the interval 500 through 2000 CE, i.e. the range of the Mann et al. data. The input data are produced using the "binboot" method described at http://geog.uoregon.edu/bartlein/GPWG/GPWGAnalysis/.

Read data

Read the appropriate data.

```
# filenames
datapath <- "/Projects/GPWG/work/feedback/data/curves/"
char_filename <- "v3i_nsa_globe_binboot_nt2kb_pbw10_bw20_200.csv"
char_file <- paste(datapath, char_filename , sep="")
char_label <- "Normans (binboot, bw=20)"
tmp_filename <- "aaves_tmp_land_binboot_v1_bw20_200.csv"
tmp_file <- paste(datapath, tmp_filename, sep="")
tmp_label <- "Mann et al. temp. (land) (binboot, bw=20)"

# read data
char_data <- read.csv(char_file); names(char_data)
tmp_data <- read.csv(tmp_file); names(tmp_data)
```

Merge the two data sets. The data span the interval from 500 CE to 2000 CE, at the 50-year intervals of the `locfit()` fitted values for each series.

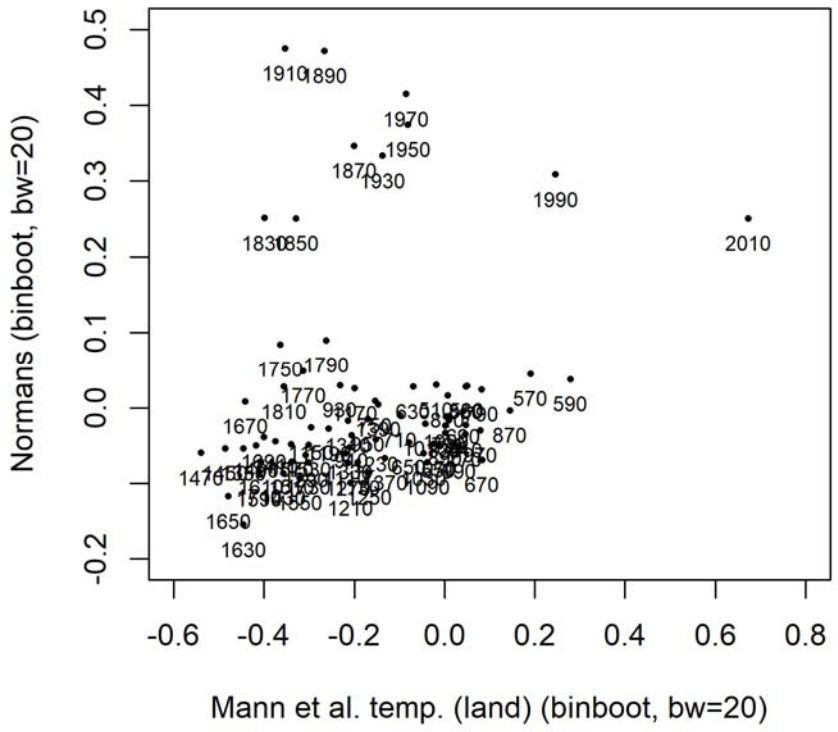
```
# merge data
data_all <- as.data.frame(matrix(NA, nrow=dim(char_data)[1], ncol=4))
names(data_all) <- c("age", "YearCE", "char", "tmp")
data_all[1] <- char_data[1]
data_all[2] <- 1950-data_all[1]
data_all[3] <- char_data[3]
data_all$tmp[1:dim(tmp_data)[1]] <- tmp_data[1:dim(tmp_data)[1], 3]
data_all <- data_all[1:dim(char_data)[1], ]
head(data_all); tail(data_all)
```

```
##   age YearCE      char        tmp
## 1 -60   2010 0.2508099  0.6727354
## 2 -40   1990 0.3089820  0.2449755
## 3 -20   1970 0.4148257 -0.0859434
## 4   0   1950 0.3746422 -0.0810824
## 5  20   1930 0.3335772 -0.1373332
## 6  40   1910 0.4749070 -0.3541846

##      age YearCE         char tmp
## 109 2100   -150  0.003760789  NA
## 110 2120   -170  0.018230746  NA
## 111 2140   -190  0.013705510  NA
## 112 2160   -210 -0.008828988  NA
## 113 2180   -230 -0.010852265  NA
## 114 2200   -250  0.250809908  NA

# remove observations with missing temperature data, and reverse the order of observations in the data
set.
data_all <- na.exclude(data_all)
data_all <- data_all[nrow(data_all):1,]
head(data_all); tail(data_all)

##     age YearCE        char         tmp
## 76 1440    510  0.03058155 -0.01815385
## 75 1420    530  0.02938115  0.04936185
## 74 1400    550 -0.04931369 -0.01358275
## 73 1380    570  0.04527504  0.19040840
## 72 1360    590  0.03806594  0.27868140
## 71 1340    610 -0.02301377  0.04771920

##   age YearCE      char        tmp
## 6  40   1910 0.4749070 -0.3541846
## 5  20   1930 0.3335772 -0.1373332
## 4   0   1950 0.3746422 -0.0810824
## 3 -20   1970 0.4148257 -0.0859434
## 2 -40   1990 0.3089820  0.2449755
## 1 -60   2010 0.2508099  0.6727354
```

Scatter plots

Get a basic scatter plot.

```
plot(data_all$tmp, data_all$char, pch=16, cex=0.5, xlim=c(-.6, .8), ylim=c(-.2,.5), xlab=tmp_label,
ylab=char_label)
text(data_all$tmp, data_all$char, lab=data_all$YearCE, cex=0.7, pos=1)
```

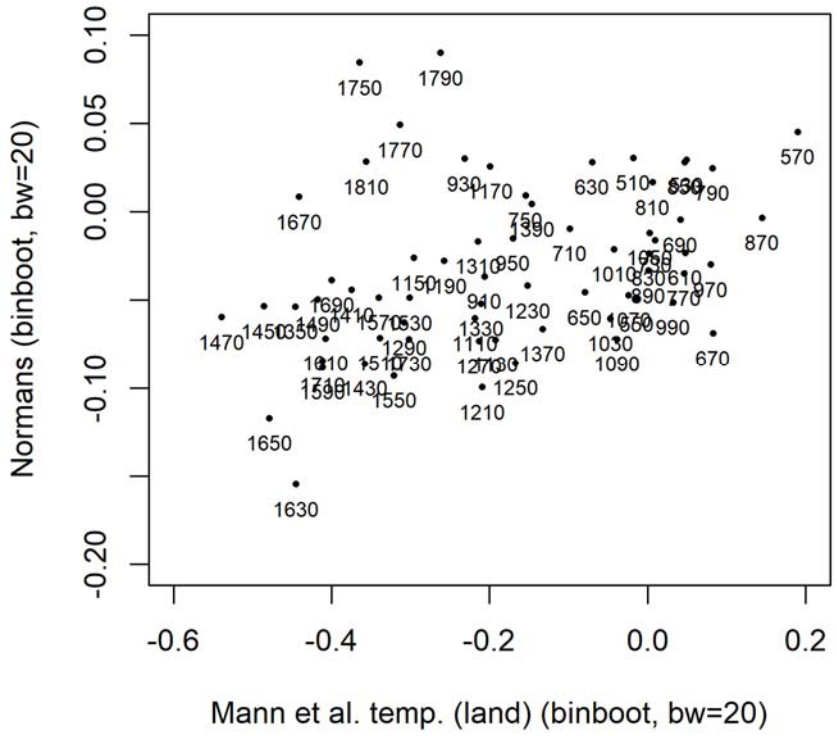


Note that 1830 to 2000 are outliers, and together these ten points have a negative slope of normalized anomalies with respect to temperature.

Pre-industrial era data

Next, restrict the data to the interval 500 to 1810 CE.

```
data_preind <- data_all[data_all$YearCE <= 1810, ]
head(data_preind); tail(data_preind)

##     age YearCE        char         tmp
## 76 1440    510  0.03058155 -0.01815385
## 75 1420    530  0.02938115  0.04936185
## 74 1400    550 -0.04931369 -0.01358275
## 73 1380    570  0.04527504  0.19040840
## 72 1360    590  0.03806594  0.27868140
## 71 1340    610 -0.02301377  0.04771920

##    age YearCE        char        tmp
## 16 240   1710 -0.08423580 -0.4117626
## 15 220   1730 -0.07204584 -0.3022804
## 14 200   1750  0.08443445 -0.3647577
## 13 180   1770  0.04913077 -0.3138278
## 12 160   1790  0.08981755 -0.2627838
## 11 140   1810  0.02828497 -0.3563674

plot(data_all$tmp, data_all$char, pch=16, cex=0.5, xlim=c(-.6, .2), ylim=c(-.2, .1), xlab=tmp_label,
ylab=char_label)
text(data_all$tmp, data_all$char, lab=data_all$YearCE, cex=0.7, pos=1)
```



Outliers are less apparent in the preindustrial data, but it will turn out that the last four observations (1750, 1770, 1790 and 1810 CE) in this "long" preindustrial-period data set are still relatively unusual, and will be truncated later.

OLS Regression

Fit a linear regression with

- Normans (binboot, bw=20) as the response (`char`), and
- Mann et al. temp. (land) (binboot, bw=20) as the predictor (`tmp`).

```
char_lm01 <- lm(char ~ tmp, data=data_preind)
summary(char_lm01)
```

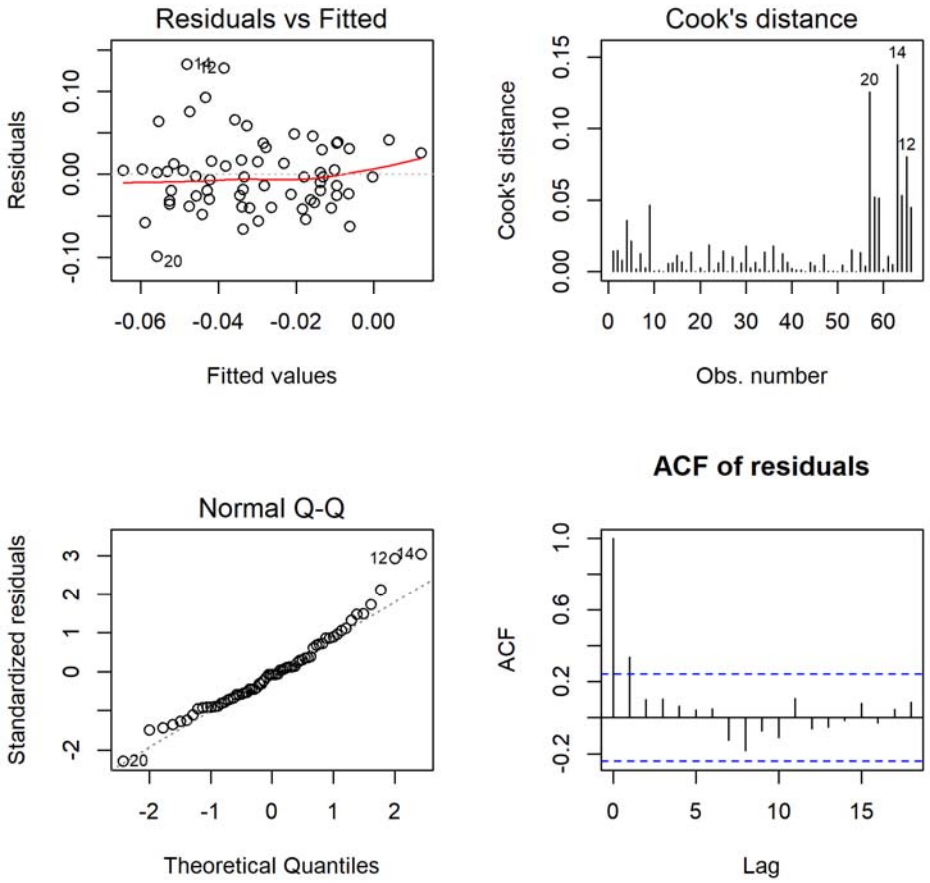
```
##
## Call:
## lm(formula = char ~ tmp, data = data_preind)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.098767 -0.030494 -0.002981  0.023815  0.132619
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.013998   0.007391  -1.894  0.06275 .
## tmp          0.093724   0.028651   3.271  0.00173 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04429 on 64 degrees of freedom
## Multiple R-squared:  0.1432, Adjusted R-squared:  0.1299
## F-statistic:  10.7 on 1 and 64 DF,  p-value: 0.001727
```

There are no issues apparent in the summary output.

Diagnostic analysis

Get the basic diagnostic plots.

```
oldpar <- par(mfcol=c(2,2))
plot(char_lm01, which=c(1,2,4))
acf(char_lm01$residuals, main="ACF of residuals", font.main=1)
```



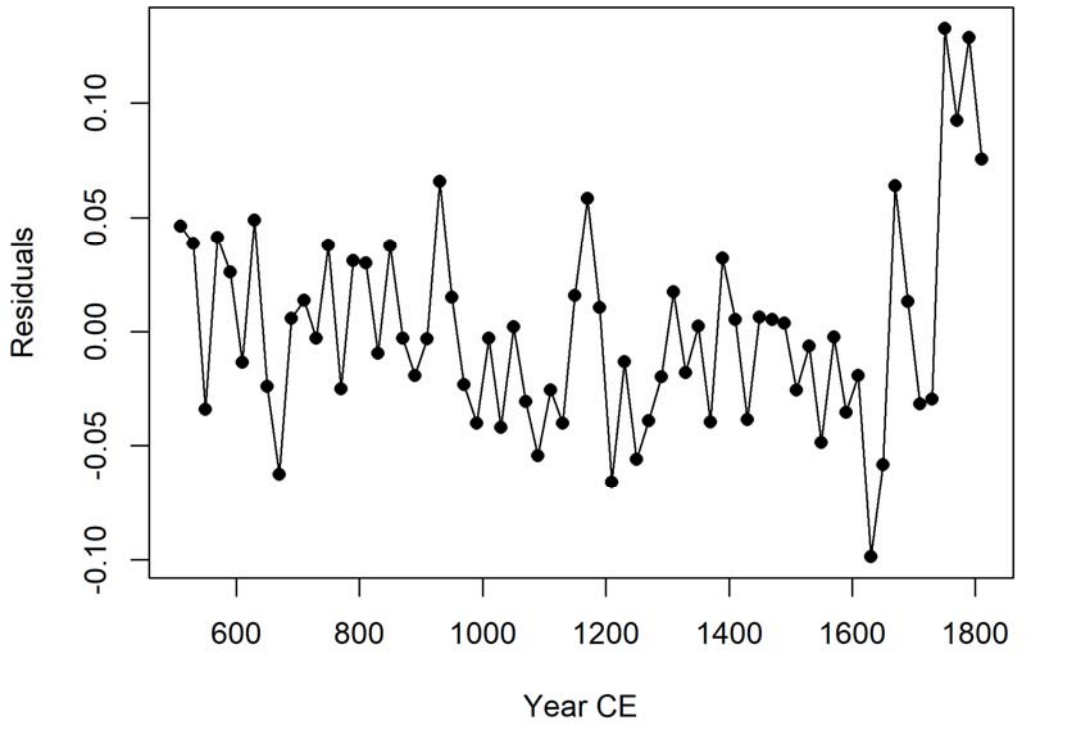

```
par <- oldpar
```

There is a little pattern in residual scatter plot, but probably not enough to warrent chaging the basic model. Cook's distance indicates that most of the influential points are from the last part of the record, The normal probability plot suggests the residual are normally distributed, with the exception of a few points, again from the lasst part of the record. The lag-1 residual autocorrelation coefficient is significant, but the notion that there is residual autocorrelation is no supported by the Ljung-Box test for serial independence in a time series.

```
Box.test(char_lm01$residuals, lag=12, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  char_lm01$residuals
## X-squared = 16.456, df = 12, p-value = 0.1712
```

Plot the time series of the residuals:

```
plot(data_preind$YearCE, char_lm01$residuals, type="o", pch=16, xlab="Year CE", ylab="Residuals")
```



The time series of residuals suggests that the "pre-industrial" should be defined to begin just before 1750 CE.

Refit the OLS regression with a shorter pre-industrial interval

Redefine the "preindustrial" interval, and redo the regression:

```
data_preind <- data_all[data_all$YearCE < 1750, ]
head(data_preind); tail(data_preind)

##     age YearCE        char         tmp
## 76 1440    510  0.03058155 -0.01815385
## 75 1420    530  0.02938115  0.04936185
## 74 1400    550 -0.04931369 -0.01358275
## 73 1380    570  0.04527504  0.19040840
## 72 1360    590  0.03806594  0.27868140
## 71 1340    610 -0.02301377  0.04771920

##    age YearCE         char        tmp
## 20 320   1630 -0.154530437 -0.4456272
## 19 300   1650 -0.117116429 -0.4792193
## 18 280   1670  0.008537671 -0.4417401
## 17 260   1690 -0.038539663 -0.4002747
## 16 240   1710 -0.084235795 -0.4117626
## 15 220   1730 -0.072045843 -0.3022804
```

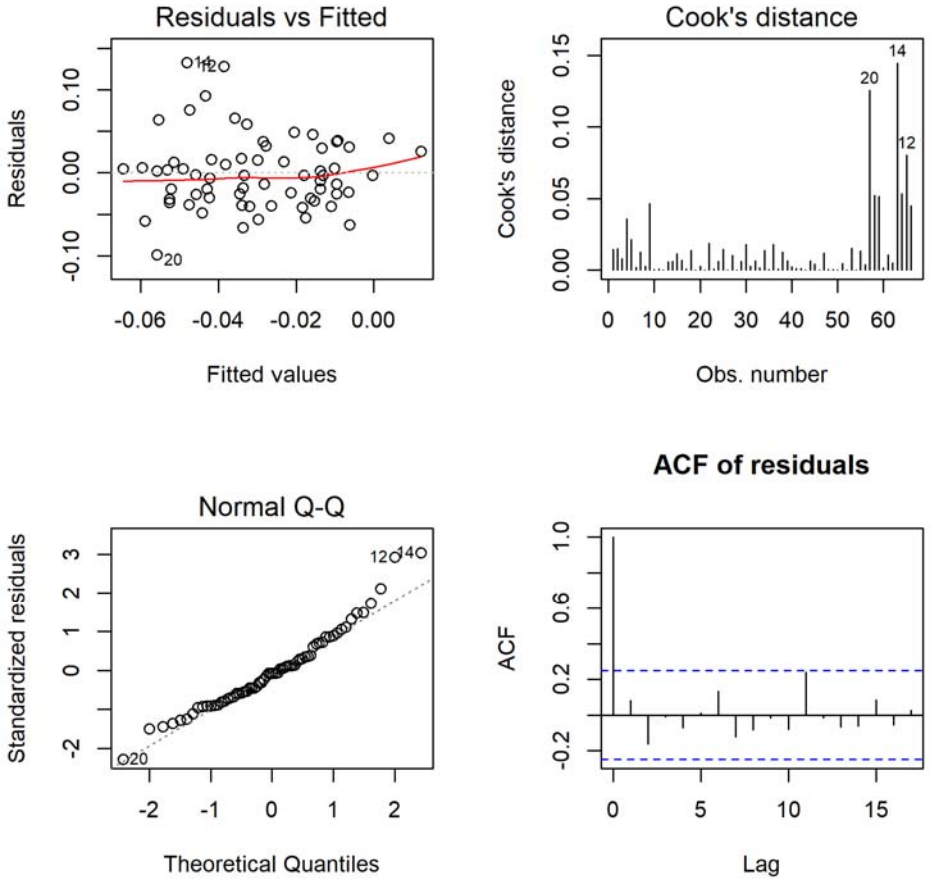
Regression with the redefined pre-industrial period:

```
char_lm02 <- lm(char ~ tmp, data=data_preind)
summary(char_lm01)

##
## Call:
## lm(formula = char ~ tmp, data = data_preind)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.098767 -0.030494 -0.002981  0.023815  0.132619
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.013998   0.007391  -1.894  0.06275 .
## tmp          0.093724   0.028651   3.271  0.00173 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04429 on 64 degrees of freedom
## Multiple R-squared:  0.1432, Adjusted R-squared:  0.1299
## F-statistic:  10.7 on 1 and 64 DF,  p-value: 0.001727
```

The fit is a lot better.

```
oldpar <- par(mfcol=c(2,2))
plot(char_lm01, which=c(1,2,4))
acf(char_lm02$residuals, main="ACF of residuals", font.main=1)
```




```
par <- oldpar
```

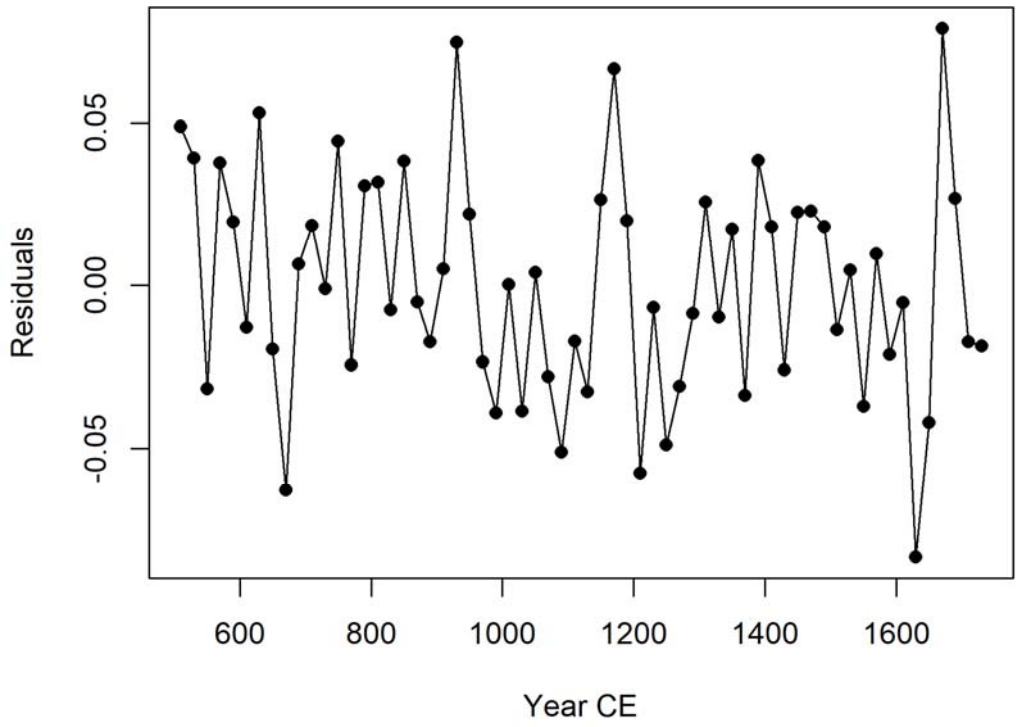
Observations close to the end of the record are still a little influential (which is not unusual for timeseries data), but the diagnostic plots look a lot better. There is a slight concave upwards pattern in the normal probability, which is suggestive of a distribution of the residuals that is slightly fatter in the tails than in a normal distribution.

```
Box.test(char_lm02$residuals, lag=12, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  char_lm02$residuals
## X-squared = 10.457, df = 12, p-value = 0.5759
```

Again, the Ljung-Box statistic suggests that there is little autocorrelation in the residuals.

Residual time series:

```
plot(data_preind$YearCE, char_lm02$residuals, type="o", pch=16, xlab="Year CE", ylab="Residuals")
```



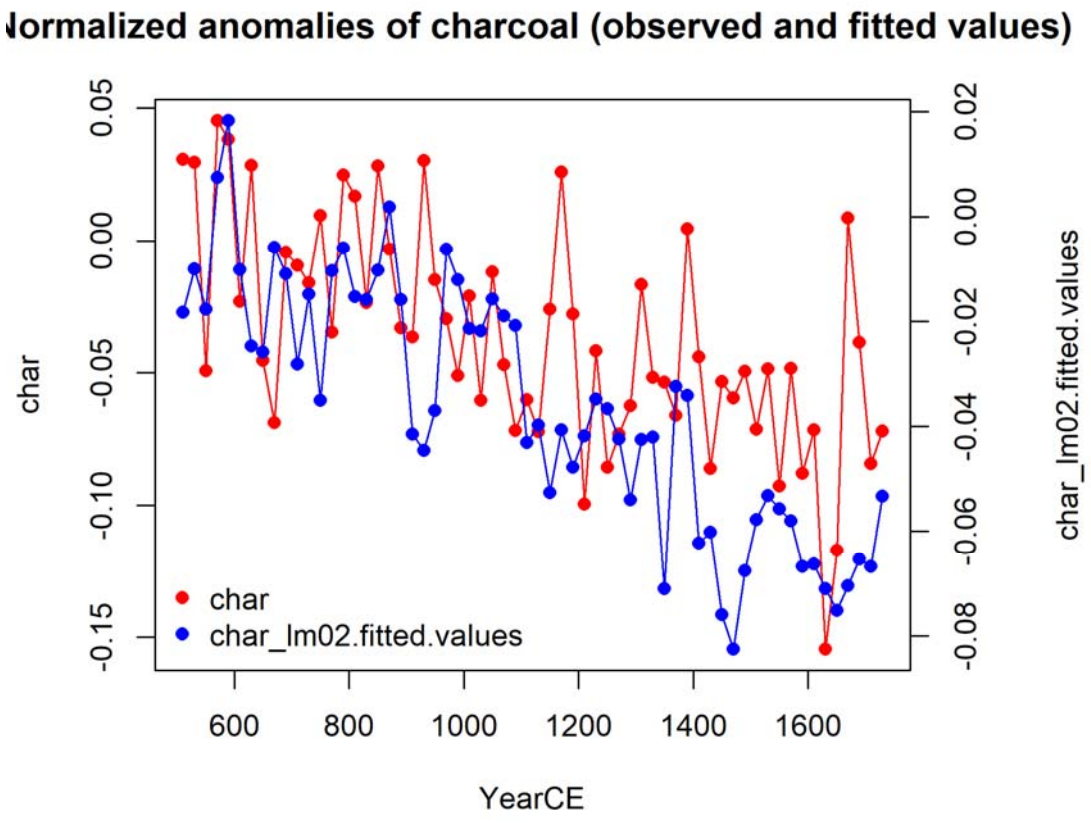
Next, examine the fit:

Function for plotting two series:

```
# function for plotting two series at a time
plotseries <- function(title, dataset, xvarin, yvar1in, yvar2in) {
  xvar <- dataset[,xvarin]
  yvar1 <- dataset[,yvar1in]; yvar2 <- dataset[,yvar2in]
  opar <- par(mar=c(5,4,4,5)+0.1) # space for second axis
  plot(yvar1 ~ xvar, pch=16, type="o", xlab=xvarin, ylab=yvar1in, col="red", data=dataset,
    main=title)
  par(new=T)          # second plot is going to get added to first
  plot(yvar2 ~ xvar, xlab="", pch=16, type="o", axes=F, ylab="", col="blue", data=dataset)
  axis(side=4); mtext(side=4,line=3.8,yvar2in)
  legend("bottomleft", bty="n", c(yvar1in,yvar2in), pch=16, col=c("red","blue"))
  par(opar)  # restore plot parameters
}
```

Observed and fitted values of the normalized anomalies of charcoal:

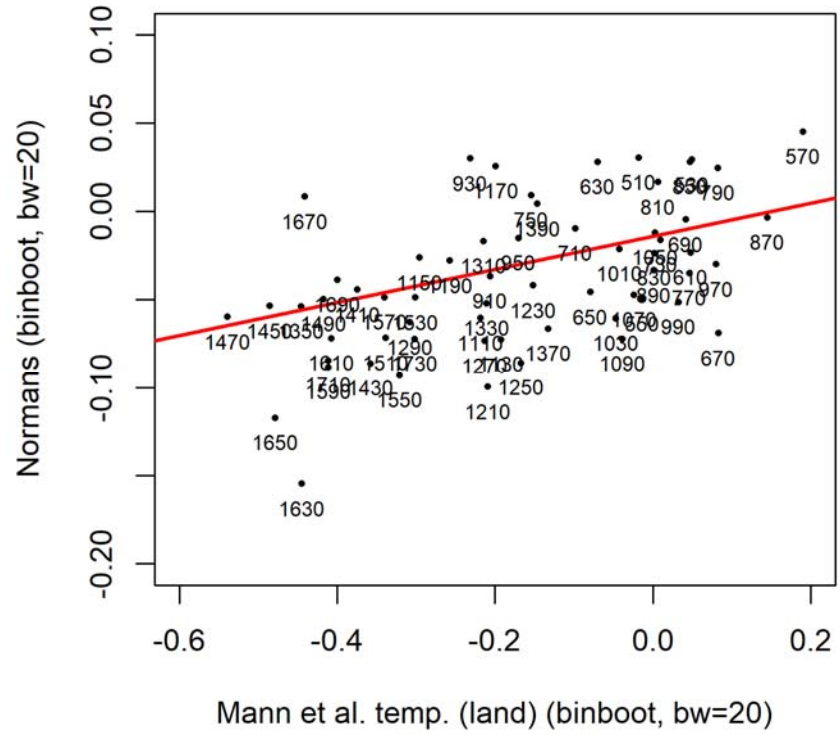
```
fitted <- data.frame(cbind(data_preind, char_lm02$residuals, char_lm02$fitted.values))
plotseries("Normalized anomalies of charcoal (observed and fitted values)", fitted, "YearCE", "char",
"char_lm02.fitted.values")
```



Note that there is some divergence between the observations (in red) and the fitted values after 1400 CE.

The basic scatter plot with the regression curve added:

```
plot(data_preind$tmp, data_preind$char, pch=16, cex=0.5, xlim=c(-.6, .2), ylim=c(-.2, .1),
xlab=tmp_label, ylab=char_label)
abline(char_lm01, col="red", lwd=2)
text(data_preind$tmp, data_preind$char, lab=data_preind$YearCE, cex=0.7, pos=1)
```

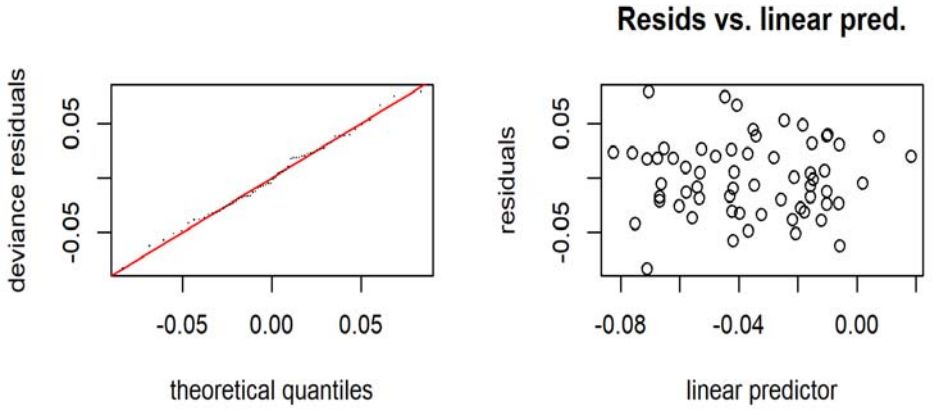


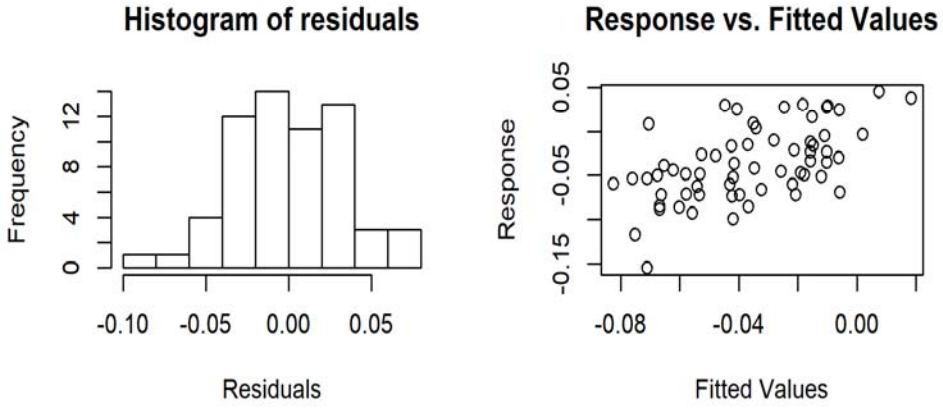
GAM Fitting

The plot above does not suggest any manifest non-linearity in the relationship between charcoal and tempeature, but we can relax the assumption of linearity by fitting a generalized additive model, which will expose any nonlinearity if it is there.

Here is a basic GAM fit, with no constraints on the nature of the model (i.e. smoothing parameters will be chosen as part of the optimization):

```
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-15. For overview type 'help("mgcv-package")'.

char_gam01 <- gam(char ~ s(tmp, bs="tp", k=4), gaussian(link = "identity"), data=data_preind)
summary(char_gam01)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## char ~ s(tmp, bs = "tp", k = 4)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.036339   0.004405   -8.25 1.84e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F  p-value
## s(tmp)   1      1 28.88 1.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.314   Deviance explained = 32.5%
## GCV = 0.001243  Scale est. = 0.0012029  n = 62

oldpar <- par(mfcol=c(2,2))
gam.check((char_gam01))
```



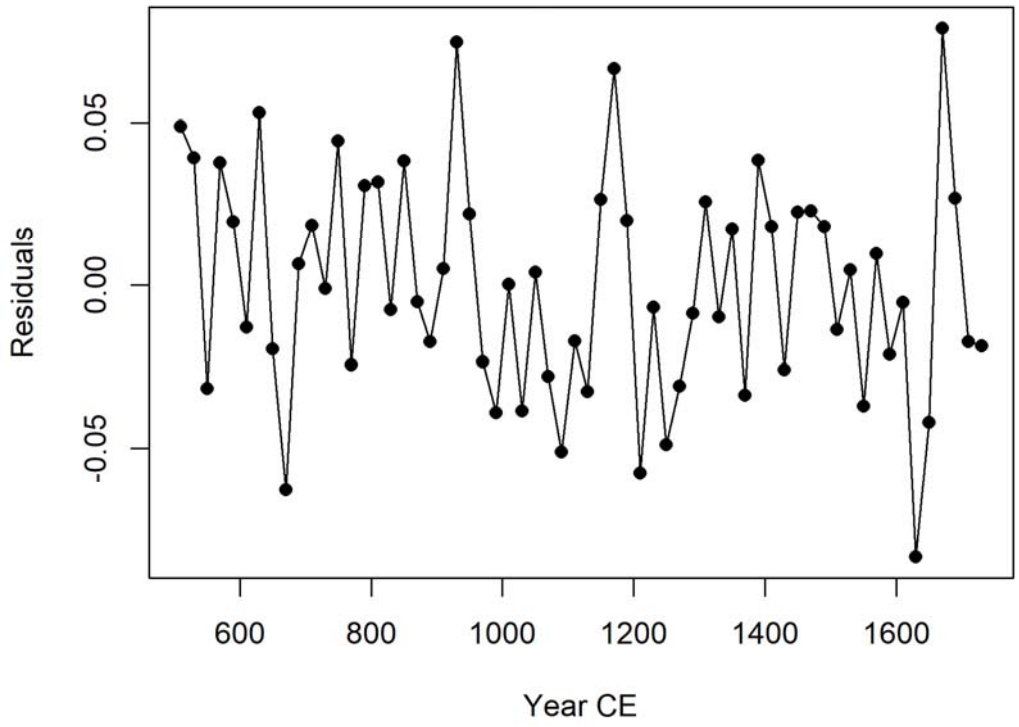



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 5 iterations.
## The RMS GCV score gradient at convergence was 1.137433e-07 .
## The Hessian was positive definite.
## Model rank =  4 / 4
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##          k'  edf k-index p-value
## s(tmp) 3.00 1.00    1.36    0.99

par <- oldpar
```

The return of 1.00 for the equivalent degrees of freedom (`edf`) implies the relationship is approximately or exactly linear.

Plot the residuals:

```
plot(data_preind$YearCE, char_gam01$residuals, type="o", pch=16, xlab="Year CE", ylab="Residuals")
```



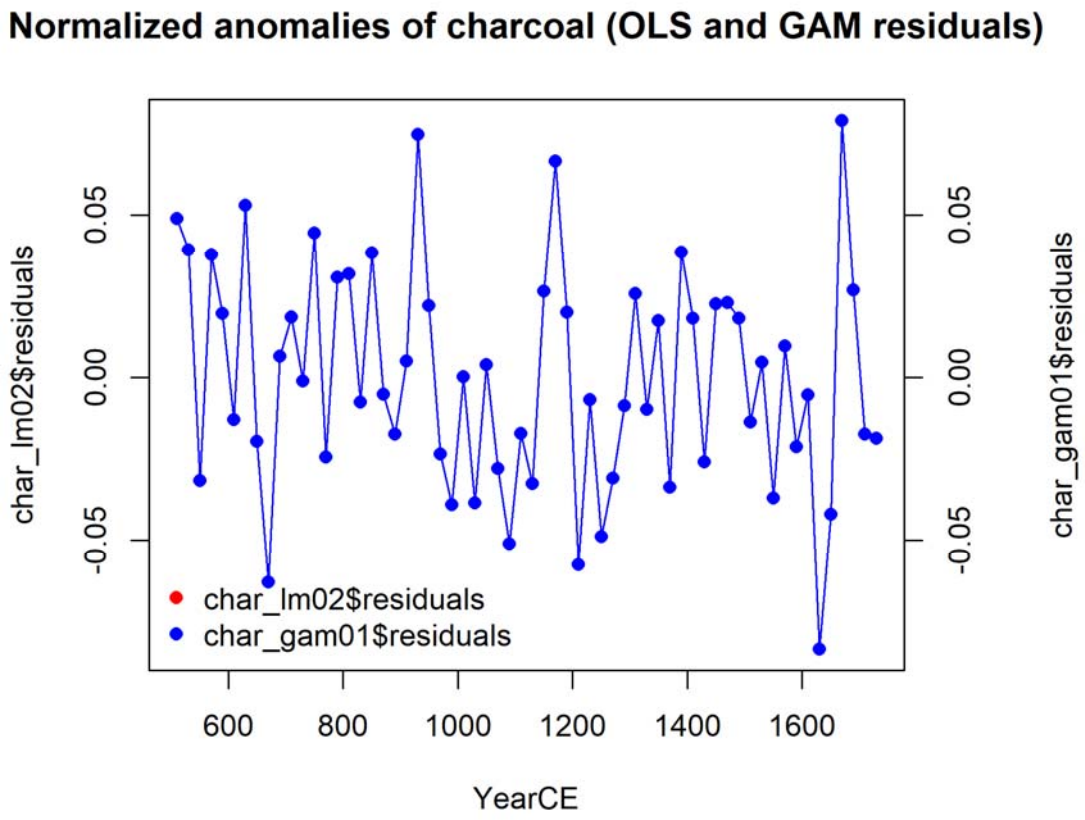
Check for residual autocorrelation:

```
Box.test(char_gam01$residuals, lag=12, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  char_gam01$residuals
## X-squared = 10.457, df = 12, p-value = 0.5759
```

Again, there is no support for the idea that the residuals are autocorrelated.

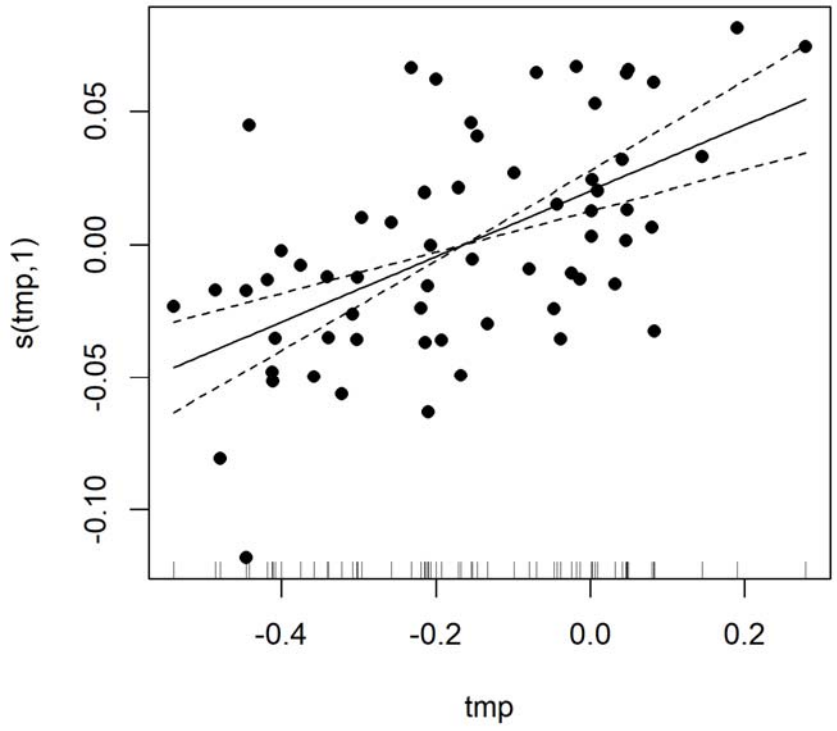
Get and plot the residual from the GAM, along with those from the OLS regression:

```
fitted2 <- data.frame(cbind(data_preind, char_lm02$residuals, char_gam01$residuals))
names(fitted2) <- c("age", "YearCE","char","tmp","char_lm02$residuals","char_gam01$residuals")
plotseries("Normalized anomalies of charcoal (OLS and GAM residuals)", fitted2, "YearCE",
"char_lm02$residuals", "char_gam01$residuals")
```



The OLS residuals are completely overplotted by the GAM residuals, again confirming that the optimal form of the charcoal vs temperature relationship is linear. This is also demonstrated by the partial residual plot:

```
plot(char_gam01, residuals=TRUE, pch=16, cex=1)
```



The results indicate that OLS regression of charcoal on global temperature fits that data adequately and provides a simple model of the relationship between charcoal and global-average temperature.