

1      Supplementary materials for: Modelling  
2      palaeoecological time series using  
3      generalized additive models

4      *Gavin L. Simpson*

5      *August 16, 2018*

6    **1 Introduction**

- 7    This document is an annotated version of the R code used to fit the GAMs and related analyses  
8    to the Small Water and Braya-Sø example data sets.  
9    The following packages are required: *mgcv*, *scam*, *ggplot2*, *cowplot*, and *tidyverse*.

```
library("mgcv")  
  
#> Loading required package: nlme  
  
#> This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.  
library("scam")
```

```
#> This is scam 1.2-2.  
  
library("ggplot2")  
library("cowplot")
```

```
#>  
#> Attaching package: 'cowplot'  
  
#> The following object is masked from 'package:ggplot2':  
#>  
#>     ggsave  
  
library("tidyverse")
```

- 18    In addition, the *gratia* package is required; this in-development package is not on CRAN but  
19    can be installed directly from GitHub using functions from the *devtools* package. To install the  
20    package, install *devtools* and then use *devtools::install\_github()* to install *gratia*, as shown  
21    below:

```
## gratia is not on CRAN, install from github:  
install.packages("devtools")  
devtools::install_github("gavinsimpson/gratia")
```

22 Assuming that the installation of *gratia* completes without error, the package can be loaded as  
23 usual

```
library("gratia") # need to change the name of the package
```

24 The final environment-preparation step is to set the default *ggplot* theme, which the loading of  
25 *cowplot* has over-ridden. Here I use the more-minimal black-and-white theme (*theme\_bw()*)

```
## Default ggplot theme
theme_set(theme_bw())
```

## 26 2 Load the data sets

27 The example data sets are also stored on GitHub; <https://github.com/gavinsimpson/>  
28 *frontiers-palaeo-additive-modelling*. Once downloaded the data are read in and processed a  
29 little

```
## source Small Water data
small <- readRDS("./data/small-water/small-water-isotope-data.rds")
head(small)

#>   Depth   d13C TotalC d15N TotalN DryWeight      Year
#> 1  0.2 -27.57  806.49 3.05  64.21       8.2 2007.254
#> 2  0.4 -27.67  949.33 3.01  73.26       7.6 2006.510
#> 3  0.8 -27.63 1305.52 2.93  93.25      11.6 2004.941
#> 4  1.2 -27.62 1136.04 2.33  86.09       9.6 2003.269
#> 5  1.6 -27.48 1028.27 2.09  93.80      10.9 2001.496
#> 6  2.0 -27.39  809.91 2.66  79.98       9.9 1999.626

## load braya so data set
braya <- read.table("./data/braya-so/DAndrea.2011.Lake Braya So.txt",
                     skip = 84)
## clean up variable names
names(braya) <- c("Depth", "DepthUpper", "DepthLower", "Year", "YearYoung",
                  "YearOld", "UK37")
## add a variable for the amount of time per sediment sample
braya <- transform(braya, sampleInterval = YearYoung - YearOld)
head(braya)

#>   Depth DepthUpper DepthLower      Year YearYoung YearOld    UK37
#> 1  0.25          0.0        0.5 1999.125  2006.00 1992.25 -0.640
#> 2  0.75          0.5        1.0 1985.375  1992.25 1978.50 -0.637
#> 3  1.25          1.0        1.5 1971.525  1978.50 1964.55 -0.614
#> 4  1.75          1.5        2.0 1957.575  1964.55 1950.60 -0.627
#> 5  2.25          2.0        2.5 1943.150  1950.60 1935.70 -0.633
#> 6  2.75          2.5        3.0 1928.250  1935.70 1920.80 -0.616
```

```

44 #> sampleInterval
45 #> 1           13.75
46 #> 2           13.75
47 #> 3           13.95
48 #> 4           13.95
49 #> 5           14.90
50 #> 6           14.90

## plot labels
d15n_label <- expression(delta^{15}N)
braya_ylabel <- expression(italic(U)[37]^{italic(k)})

```

51 Plots of the two data sets are prepared using `ggplot2`

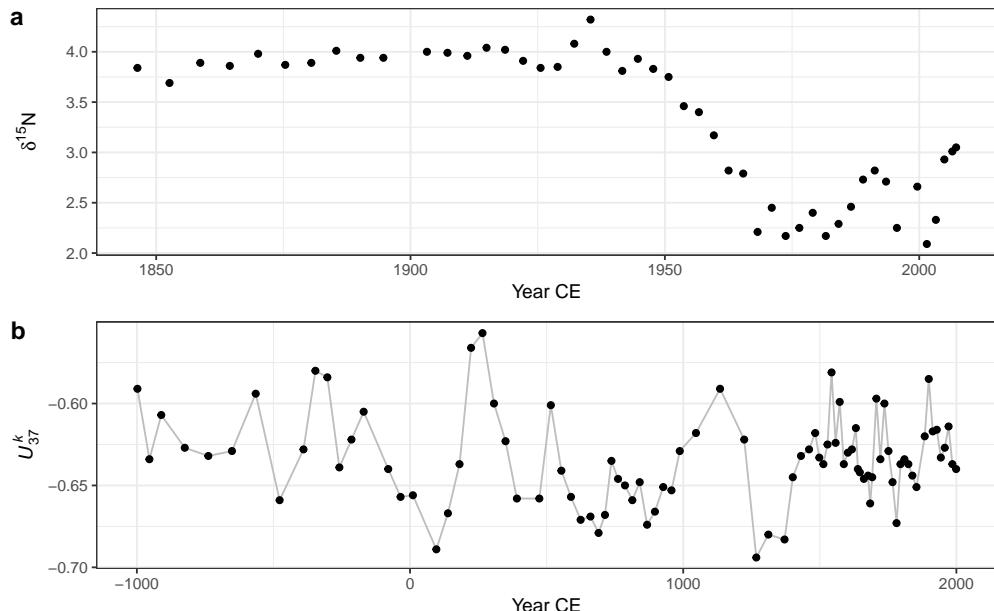
```

## plot Small Water data
small_plt <- ggplot(small, aes(x = Year, y = d15N)) +
  geom_point() +
  labs(y = d15n_label, x = "Year CE")

## plot Braya-So data
braya_plt <- ggplot(braya, aes(x = Year, y = UK37)) +
  geom_line(colour = "grey") +
  geom_point() +
  labs(y = braya_ylabel, x = "Year CE")

## Recreate plot from manuscript
plot_grid(small_plt, braya_plt, ncol = 1, labels = "auto", align = "hv",
          axis = "lr")

```



52

## 53 3 Fitting GAMs

54 Instead of following the structure of the paper exactly, the subsequent sections focus on the  
55 two example data sets in turn, beginning with Small Water.

### 56 3.1 Small Water

57 A GAM is typically fitted using the `gam()` function from the `mgcv` package. The typical call  
58 includes a formula describing the response variable and the linear predictor separated by the  
59 `~` (tilde) symbol. The linear predictor part of the model contains the smooth function of the  
60 time variable in the data set, and is indicated using the `s()` function. Unless other options are  
61 needed, we only need specify where the variables in the formula can be found via the `data`  
62 argument, and that we wish to use REML smoothness selection.

63 Putting this together we have the following function call to fit a simple GAM to the Small Water  
64 isotope data

```
m <- gam(d15N ~ s(Year, k = 15), data = small, method = "REML")
```

65 As discussed in the paper, this model assumes that residuals<sup>1</sup> are independent. To account  
66 for residual temporal autocorrelation we can include a continuous-time first-order autoregres-  
67 sive (CAR(1)) process in the model residuals for GAMs fitted to data that are conditionally  
68 distributed Gaussian.

69 The GAM plus CAR(1) process is fitted to the Small Water data set using the `gamm()` function.  
70 This fits GAMs as mixed effects models via the `nlme` package, which allows the use of corre-  
71 lation structures in the model residuals via the `correlation` argument. Here, the `corCAR1()`  
72 function is used to select the CAR(1) process and we specify the ordering of samples via the  
73 `Year` variable in `small`.

```
## fit small water GAM using gamm() with a CAR(1)
mod <- gamm(d15N ~ s(Year, k = 15), data = small,
             correlation = corCAR1(form = ~ Year), method = "REML")
```

74 The estimated value of  $\phi$  for the CAR(1) can be extracted from the fitted model via the `$lme`  
75 component. Here we just extract the correlation structure component.

```
## estimate of phi and confidence interval
smallPhi <- intervals(mod$lme, which = "var-cov")$corStruct
smallPhi
```

```
76 #>      lower      est.      upper
77 #> Phi 0.282388 0.6026967 0.8539689
78 #> attr(,"label")
79 #> [1] "Correlation structure:"
```

<sup>1</sup>or equivalently that the observations, conditional upon the model, are independent.

80 The object returned by `gamm()` includes both the linear mixed model and GAM faces of the  
81 model, and as a result we need to access the separate elements (`lme` and `gam` respectively)  
82 when proceeding to explore the model fit. The model summary is prepared from the `$gam`  
83 component of the fitted model

```
## summary object
summary(mod$gam)

#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> d15N ~ s(Year, k = 15)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 3.30909   0.03489   94.84 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>             edf Ref.df      F p-value
#> s(Year) 7.954 7.954 47.44 <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.929
#> Scale est. = 0.037268 n = 48
```

105 The output shows the estimated complexity of the fitted smooth, expressed in terms of the ef-  
106 fective degrees of freedom of the spline. An associated  $F$  statistic and test of the null hypoth-  
107 esis of no trend (effect) are also shown. Here the estimated trend provides strong evidence  
108 against this null.

109 The CAR(1) process plotted in Figure 11 of the manuscript was prepared using

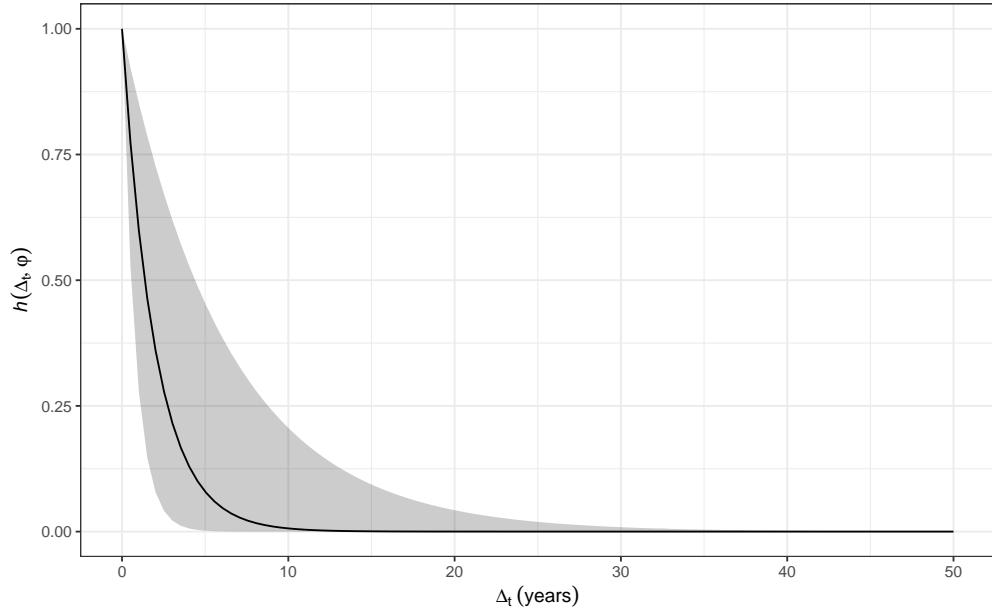
```
## plot CAR(1) process
S <- seq(0, 50, length = 100)
car1 <- setNames(as.data.frame(t(outer(smallPhi, S, FUN = `^`)[1, , ])),
                 c("Lower", "Correlation", "Upper"))
car1 <- transform(car1, S = S)

car1Plt <- ggplot(car1, aes(x = S, y = Correlation)) +
  geom_ribbon(aes(ymax = Upper, ymin = Lower),
              fill = "black", alpha = 0.2) +
  geom_line() +
```

```

    ylab(expression(italic(h) * (list(Delta[t], varphi)))) +
    xlab(expression(Delta[t] ~ (years)))
car1Plt

```



110

- 111 The exponential decline in correlation with increasing separation is evident here; once samples  
112 are  $\sim 10$  years apart, there is little estimated dependence between them.  
113 The next code chunk prepares a plot of the fitted GAMS. The general idea is to predict from  
114 the fitted model for a fine grid of points over the range of the time variable. Below I plot the  
115 trend for Small Water with an approximate 95% point-wise confidence interval that assumes  
116 asymptotic normality

```

N <- 300    # number of points at which to evaluate the smooth

## create new data to predict at; 200 evenly-spaced values over `Year`
newYear <- with(small, data.frame(Year = seq(min(Year), max(Year),
                                         length.out = 200)))

## Predict from the fitted model; note we predict from the $gam part
newYear <- cbind(newYear,
                  data.frame(predict(mod$gam, newYear, se.fit = TRUE)))

## Create the confidence interval
crit.t <- qt(0.975, df = df.residual(mod$gam))
newYear <- transform(newYear,
                      upper = fit + (crit.t * se.fit),
                      lower = fit - (crit.t * se.fit))

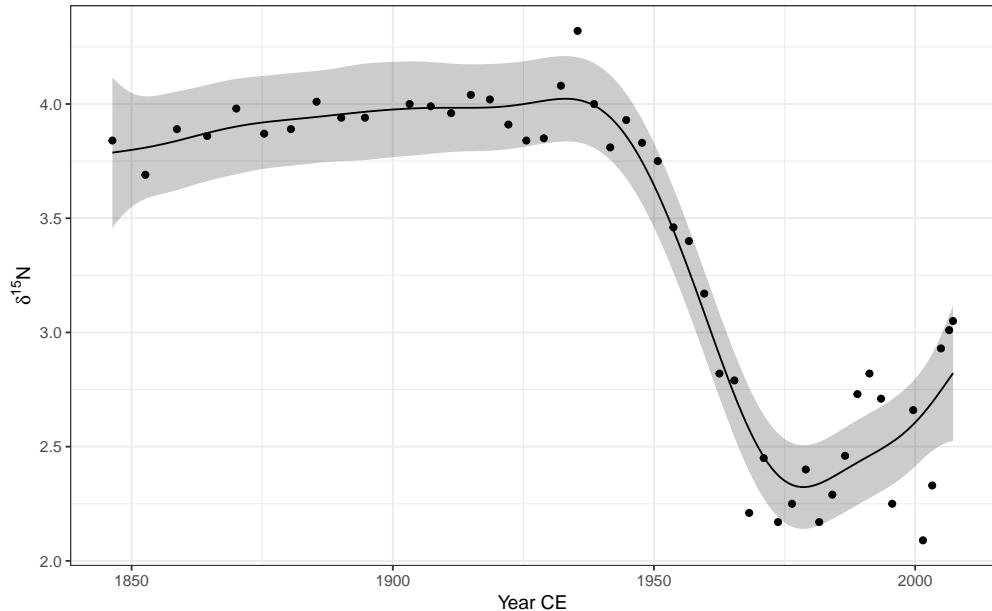
## Plot estimated trend
small_fitted <- ggplot(newYear, aes(x = Year, y = fit)) +
  geom_ribbon(aes(ymin = lower, ymax = upper, x = Year), alpha = 0.2,

```

```

            inherit.aes = FALSE, fill = "black") +
geom_point(data = small, mapping = aes(x = Year, y = d15N),
            inherit.aes = FALSE) +
geom_line() +
labs(y = d15n_label, x = "Year CE")
small_fitted

```



117

## 118 3.2 Braya-Sø

119 The same model as used for Small Water was also initially attempted with the Braya-Sø data.  
 120 However, in order to even fit the model with both a smooth and the CAR(1) process, I had  
 121 to change the default optimizer used to estimate the model parameters, and reduce the basis  
 122 dimension, k, to a small number. I also fit the model using GCV, which is, at the time of writing,  
 123 the default in `gam()`, hence no `method` argument

```

## fit the car(1) model --- needs optim as this is not a stable fit!
## also needs k setting lower than default
braya.car1 <- gamm(UK37 ~ s(Year, k = 5), data = braya,
                     correlation = corCAR1(form = ~ Year),
                     method = "REML",
                     control = list(niterEM = 0, optimMethod = "BFGS",
                                   opt = "optim"))

## fit model using GCV
braya.gcv <- gam(UK37 ~ s(Year, k = 30), data = braya)

## estimate of phi and confidence interval

```

```

brayaPhi <- intervals(braya.car1$lme)$corStruct
brayaPhi

124 #>           lower      est. upper
125 #> Phi 2.680033e-31 0.2000156     1
126 #> attr(,"label")
127 #> [1] "Correlation structure:"
```

128 Note the wide confidence interval — effectively 0–1 — on  $\phi$ . If you were to increase the value of  
129 k to be k = 10 in the s(Year) above, the model will fit but a warning message will be emitted  
130 when trying to extract  $\phi$  due to a non-positive definite model covariance matrix, indicating  
131 problems with the model.

132 To plot the GAMs fitted to the Braya-Sø time series, we repeat the process used with the Small  
133 Water GAM, but we do this for both models (GAMM + CAR(1) and GCV), using a critical value  
134 from the t distribution to form the confidence interval

```

N <- 300    # number of points at which to evaluate the smooth

## data to predict at
newBraya <- with(braya, data.frame(Year = seq(min(Year), max(Year),
                                              length.out = N)))

## add predictions from GAMM + CAR(1) model
newBraya <- cbind(newBraya,
                   data.frame(predict(braya.car1$gam, newBraya,
                                      se.fit = TRUE)))
crit.t <- qt(0.975, df = df.residual(braya.car1$gam))
newBraya <- transform(newBraya,
                       upper = fit + (crit.t * se.fit),
                       lower = fit - (crit.t * se.fit))

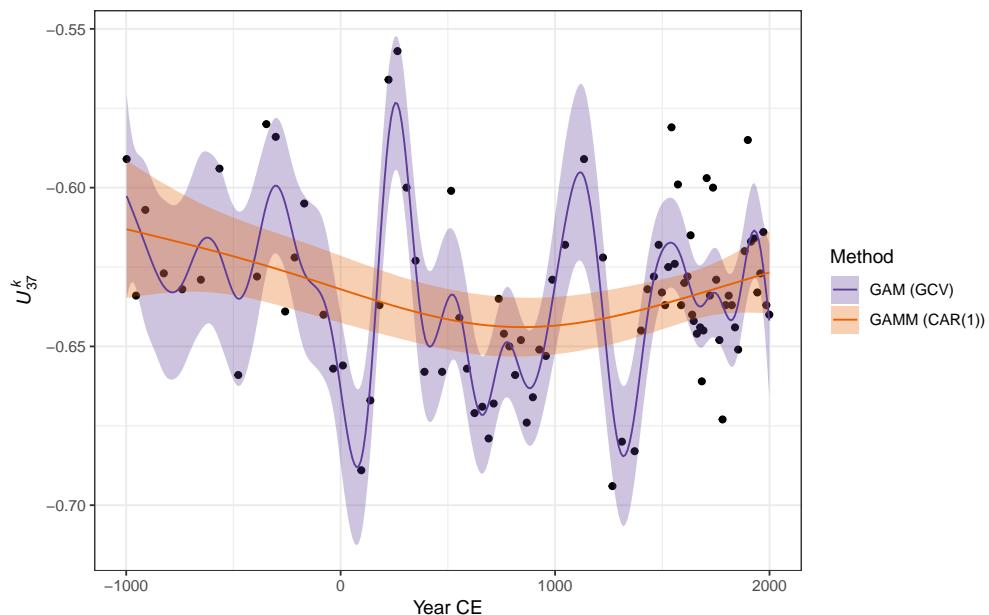
## add GAM GCV results
fit_gcv <- predict(braya.gcv, newdata = newBraya, se.fit = TRUE)
crit.t <- qt(0.975, df.residual(braya.gcv))
newGCV <- data.frame(Year = newBraya[["Year"]],
                      fit = fit_gcv$fit,
                      se.fit = fit_gcv$se.fit)
newGCV <- transform(newGCV,
                     upper = fit + (crit.t * se.fit),
                     lower = fit - (crit.t * se.fit))
newBraya <- rbind(newBraya, newGCV)          # bind on GCV results
## Add indicator variable for model
newBraya <- transform(newBraya,
                      Method = rep(c("GAMM (CAR(1))", "GAM (GCV)"),
                                   each = N))

```

```

## plot CAR(1) and GCV fits
braya_fitted <- ggplot(braya, aes(y = UK37, x = Year)) +
  geom_point() +
  geom_ribbon(data = newBraya,
    mapping = aes(x = Year, ymax = upper, ymin = lower,
                  fill = Method),
    alpha = 0.3, inherit.aes = FALSE) +
  geom_line(data = newBraya,
    mapping = aes(y = fit, x = Year, colour = Method)) +
  labs(y = braya_ylabel, x = "Year CE") +
  scale_color_manual(values = c("#5e3c99", "#e66101")) +
  scale_fill_manual(values = c("#5e3c99", "#e66101")) +
  theme(legend.position = "right")
braya_fitted

```



135

136 Figure 6 in the manuscript was produced using:

```

plot_grid(small_fitted, braya_fitted, ncol = 1, labels = "auto",
          align = "hv", axis = "lr")

```

### 137 3.2.1 Checking if the size of the basis expansion is sufficient

138 For the Braya-Sø data, clearly we are not able to identify both the wiggly trend and the CAR(1)  
 139 process in a single model. We proceed by fitting a simple GAM via `gam()` with REML smooth-  
 140 ness selection and a moderate number of basis functions — here we set `k = 15` for illustration.

```

braya_low_k <- gam(UK37 ~ s(Year, k = 15), data = braya, method = "REML")

```

141 Before proceeding, we should perform a check to determine if the number of basis functions re-  
142 quested is sufficient to capture the wiggleness in the data. This test is provided by `gam.check()`

```
gam.check(braya_low_k)
```

```
143 #>
144 #> Method: REML Optimizer: outer newton
145 #> full convergence after 6 iterations.
146 #> Gradient range [-4.468978e-08,8.696768e-10]
147 #> (score -187.2524 & scale 0.000689802).
148 #> Hessian positive definite, eigenvalue range [0.5525272,43.5153].
149 #> Model rank = 15 / 15
150 #>
151 #> Basis dimension (k) checking results. Low p-value (k-index<1) may
152 #> indicate that k is too low, especially if edf is close to k'.
153 #>
154 #>          k'    edf k-index p-value
155 #> s(Year) 14.00  2.62    0.56  <2e-16 ***
156 #> ---
157 #> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

158 The first few lines of output from `gam.check()` include information on the model fit, whether  
159 the algorithm converged, and some diagnostics from the optimization at convergence — you  
160 are unlikely to need this information, but it can be useful in diagnosing problems with model  
161 fitting or when reporting problems to Simon Wood, the developer of the `mgcv` package. In  
162 the table at the bottom of the output from `gam.check()`, the column labelled `k'` is the size of  
163 the basis used to fit the model (the number of basis functions). Note that the stated value is  
164 14 and not the requested 15 functions because the constant basis function has been removed  
165 due to it being confounded with the model constant term, the intercept. The column labelled  
166 `edf` indicates the *effective degrees of freedom* for the estimated smooth, which is a measure of the  
167 complexity or wiggleness of the final smooth. The `k-index` column contains the value of a  
168 test statistic for a test of a sufficient number of basis functions — ideally, we'd like the value of  
169 `k-index` to be close to or greater than 1. The quoted *p* value measures the support for the null  
170 hypothesis that enough basis functions were available. In practical terms, the low `k-index` and  
171 very low *p* value shown above strongly suggest that the requested number of basis functions  
172 used to fit the model was too low.

173 The solution is to increase `k`, say by doubling the original value, and refit the model and per-  
174 form the basis dimension check using `gam.check()`, repeating this process until `k-index` is  
175 close to or greater than 1 and the *p* value is larger than a desired threshold. In my experience  
176 fitting GAMs to palaeoenvironmental time series, it may not be possible with some data sets  
177 to increase `k` large enough to result in a `k-index` > 1 and a *p* value > 0.05, given the available  
178 number of samples. It is important to remember that the test provided by `gam.check()` is only  
179 an heuristic one and shouldn't be seen as infallible. If you find yourself increasing `k` to large  
180 values relative to the number of samples in your data set without achieving a `k-index` > 1 or *p*  
181 > 0.05, then note the value in the `edf` column as you increase `k`. If the `edf` of the model changes

182 little as you continue to increase  $k$ , this may be evidence that the test is failing in this particular  
183 instance and that the estimated smooth is not dependent on the value of  $k$  chosen.

### 184 3.3 Accounting for heteroscedasticity due to time averaging

185 To proceed with the Braya-Sø example, we need to increase the basis dimension, fit using  
186 `method = "REML"`. As discussed in the manuscript, we should also account for the non-  
187 constant variance, or *heteroscedasticity*, that may arise due to variation in the amount of time  
188 (or “lake years”) that each sample represents. All else equal, we expect that samples that av-  
189 erage more time have lower variance than those that average a smaller amount of time. To  
190 include information about the expected heteroscedasticity in the GAM we can use observa-  
191 tional weights<sup>2</sup>. Here, I use the `sampleInterval` variable that I created earlier as the measure  
192 of lake years per sample, and, to avoid changing the model likelihood, the weights use are the  
193 values of `sampleInterval` divided by the mean of `sampleInterval`:

```
braya_reml <- gam(UK37 ~ s(Year, k = 40), data = braya,  
                     method = "REML",  
                     weights = sampleInterval / mean(sampleInterval))  
summary(braya_reml)
```

```
194 #>  
195 #> Family: gaussian  
196 #> Link function: identity  
197 #>  
198 #> Formula:  
199 #> UK37 ~ s(Year, k = 40)  
200 #>  
201 #> Parametric coefficients:  
202 #>             Estimate Std. Error t value Pr(>|t|)  
203 #> (Intercept) -0.633741  0.001929 -328.5  <2e-16 ***  
204 #> ---  
205 #> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
206 #>  
207 #> Approximate significance of smooth terms:  
208 #>             edf Ref.df   F p-value  
209 #> s(Year)    27.45  32.36 7.782  <2e-16 ***  
210 #> ---  
211 #> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
212 #>  
213 #> R-sq.(adj) =  0.744  Deviance explained = 82.4%  
214 #> -REML = -175.61  Scale est. = 0.00024757 n = 89
```

<sup>2</sup>This simple solution is for a Gaussian GAM. For models with other distributions, the exact interpretation of observational weights will vary. For such data, where the conditional mean and variance of the observations are linked, the use of location-scale, or distributional, models may be more appropriate and informative.

215 If we now check if 39 ( $k = 40$ ) basis functions is sufficient

```
gam.check(braya_reml)

216 #>
217 #> Method: REML Optimizer: outer newton
218 #> full convergence after 9 iterations.
219 #> Gradient range [-1.105422e-06,1.500959e-07]
220 #> (score -175.6093 & scale 0.0002475669).
221 #> Hessian positive definite, eigenvalue range [2.644434,47.78257].
222 #> Model rank = 40 / 40
223 #>
224 #> Basis dimension (k) checking results. Low p-value (k-index<1) may
225 #> indicate that k is too low, especially if edf is close to k'.
226 #>
227 #>          k'   edf k-index p-value
228 #> s(Year) 39.0 27.5     1.27    0.99
```

229 we note that the value of  $k$ -index is now greater than 1 and the  $p$  value suggests there is  
 230 very little evidence against the null hypothesis. This result strongly suggests that there were  
 231 sufficient basis functions used to estimated the trend.

232 Also note that the edf of the estimated smooth is considerably below the maximum ( $k'$ ). This  
 233 is the effect of the smoothness penalty removing the excessive wigginess possible with 39 basis  
 234 functions. If the edf were close to  $k'$ , you might consider increasing  $k$  by a modest amount (in  
 235 this instance by 5 or 10 additional basis functions) to assure yourself that you have sufficient  
 236 basis functions. This has the additional advantage of allowing a richer set of smooths of a  
 237 given complexity (edf) to be represented using this larger set of basis functions, which may  
 238 improve the model fit to the data without changing the edf too much.

## 239 4 Posterior simulation

240 Samples from the posterior distribution of a GAM can be drawn using the `simulate()` meth-  
 241 ods from the *gratia* package.

```
set.seed(1) # set the random seed to make this reproducible
nsim <- 20 # how many simulations to draw

## do the simulations
sims <- simulate(mod, nsim = nsim, newdata = newYear, unconditional = TRUE)

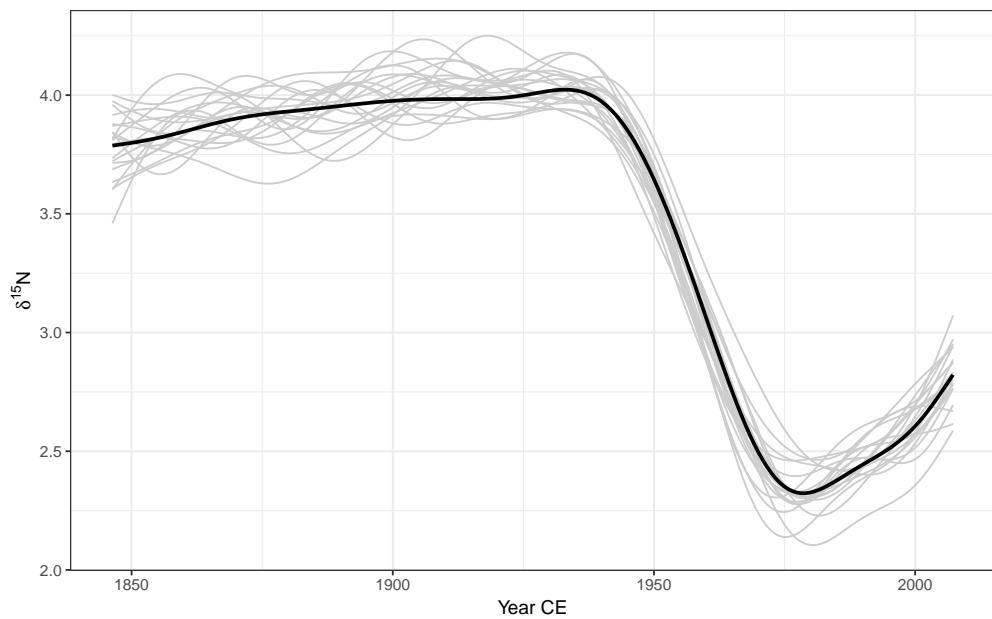
## rearrange the output into a long/tidy format
colnames(sims) <- paste0("sim", seq_len(nsim))
sims <- setNames(stack(as.data.frame(sims)), c("simulated", "run"))
sims <- transform(sims, Year = rep(newYear$Year, nsim),
```

```

    simulated = simulated)

## Plot simulated trends
smallSim.plt <- ggplot(newYear, aes(x = Year, y = fit)) +
  geom_line(data = sims,
             mapping = aes(y = simulated, x = Year, group = run),
             colour = "grey80") +
  geom_line(lwd = 1) +
  labs(y = d15n_label, x = "Year CE")
smallSim.plt

```



242

243 We repeat the same simulation for Braya-Sø

```

## data points to simulate at
newBraya <- with(braya,
                  data.frame(Year = seq(min(Year), max(Year),
                                         length.out = N)))
braya_pred <- cbind(newBraya,
                      data.frame(predict(braya_reml, newBraya,
                                         se.fit = TRUE)))

## simulate
set.seed(1)
sims2 <- simulate(braya_reml, nsim = nsim, newdata = newBraya,
                   unconditional = TRUE)

## rearrange the output into a long/tidy format
colnames(sims2) <- paste0("sim", seq_len(nsim))

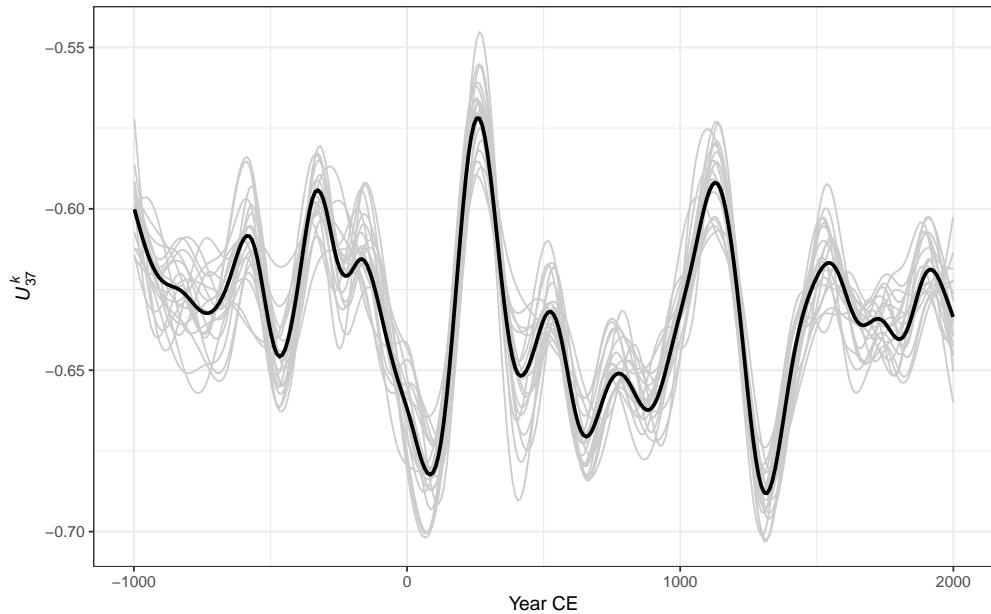
```

```

sims2 <- setNames(stack(as.data.frame(sims2)),
                  c("simulated", "run"))
sims2 <- transform(sims2, Year = rep(newBraya$Year, nsim),
                   simulated = simulated)

brayaSim.plt <- ggplot(braya_pred, aes(x = Year, y = fit)) +
  geom_line(data = sims2,
             mapping = aes(y = simulated, x = Year, group = run),
             colour = "grey80") +
  geom_line(lwd = 1) +
  labs(y = braya_ylabel, x = "Year CE")
brayaSim.plt

```



244

245 Figure 8 in the manuscript was prepared using

```

plot_grid(smallSim.plt, brayaSim.plt, ncol = 1, labels = "auto",
          align = "hv", axis = "lr")

```

## 246 5 Confidence and simultaneous intervals

247 Across-the-function and simultaneous confidence intervals are computed using the  
 248 `confint()` method. The type of interval required is given via the `type` argument with  
 249 options "confidence" and "simultaneous". For example, for Small Water we would use

```

sw.cint <- confint(mod, parm = "Year", newdata = newYear,
                     type = "confidence")
sw.sint <- confint(mod, parm = "Year", newdata = newYear,

```

```

          type = "simultaneous")
head(sw.sint)

250 #>   smooth      Year      est       se      crit    lower     upper
251 #> 1 s(Year) 1846.319 3.787215 0.1616144 3.072403 3.290671 4.283760
252 #> 2 s(Year) 1847.128 3.789747 0.1509100 3.072403 3.326090 4.253403
253 #> 3 s(Year) 1847.937 3.792304 0.1411439 3.072403 3.358653 4.225955
254 #> 4 s(Year) 1848.745 3.794914 0.1324961 3.072403 3.387832 4.201995
255 #> 5 s(Year) 1849.554 3.797602 0.1251070 3.072403 3.413222 4.181981
256 #> 6 s(Year) 1850.363 3.800394 0.1190552 3.072403 3.434608 4.166180

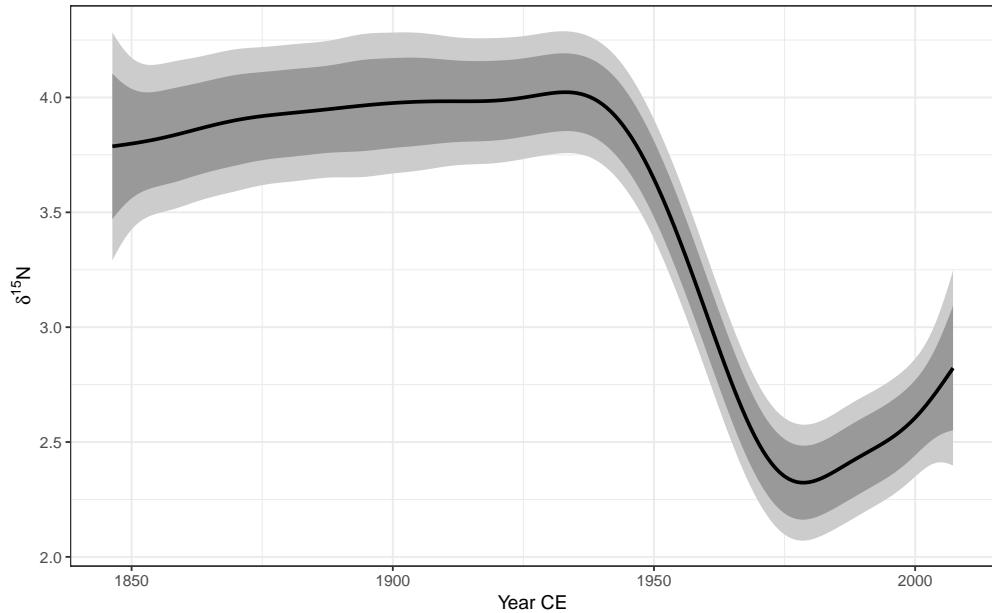
257 The confint() methods return data frames suitable for subsequent plotting with ggplot. The
258 columns labelled est and se are the estimate values of the smooth and its standard error,
259 respectively. The variables lower and upper contain the values of the lower and upper bounds
260 on the requested interval. The intervals can be plotted as follows

```

```

smallInt.plt <- ggplot(sw.cint, aes(x = Year, y = est)) +
  geom_ribbon(data = sw.sint,
               mapping = aes(ymin = lower, ymax = upper, x = Year),
               fill = "grey80", inherit.aes = FALSE) +
  geom_ribbon(mapping = aes(ymin = lower, ymax = upper, x = Year),
               fill = "grey60", inherit.aes = FALSE) +
  geom_line(lwd = 1) +
  labs(y = d15n_label, x = "Year CE")
smallInt.plt

```



261

262 The intervals for Braya-Sø are created

```

bs.cint <- confint(braya_reml, parm = "Year", newdata = newBraya,
                     type = "confidence")
bs.sint <- confint(braya_reml, parm = "Year", newdata = newBraya,
                     type = "simultaneous")

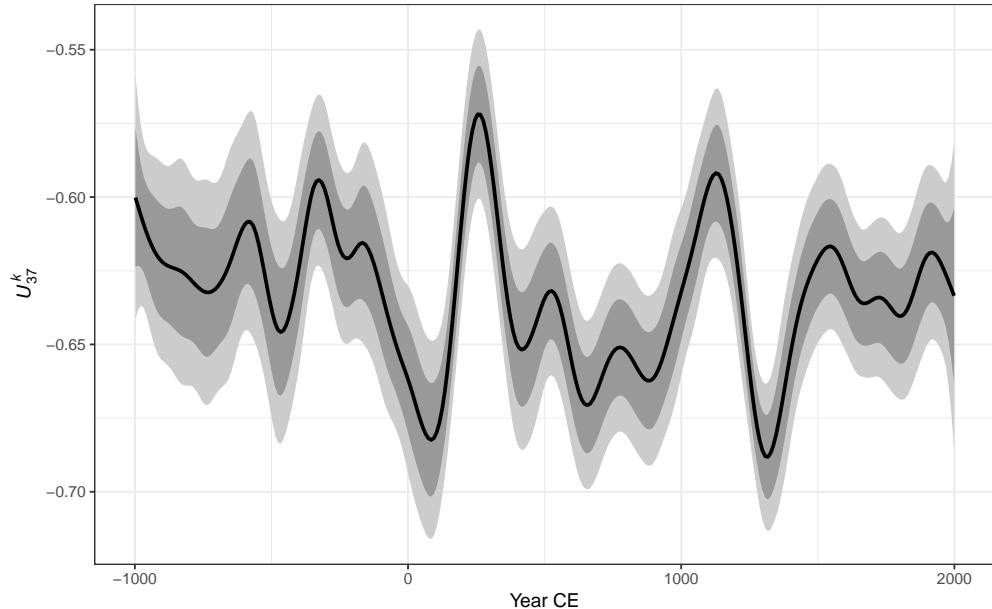
```

263 and plotted

```

brayaInt.plt <- ggplot(bs.cint, aes(x = Year, y = est)) +
  geom_ribbon(data = bs.sint,
               mapping = aes(ymin = lower, ymax = upper, x = Year),
               fill = "grey80", inherit.aes = FALSE) +
  geom_ribbon(mapping = aes(ymin = lower, ymax = upper, x = Year),
               fill = "grey60", inherit.aes = FALSE) +
  geom_line(lwd = 1) +
  labs(y = braya_ylabel, x = "Year CE")
brayaInt.plt

```



264

265 in the same way

266 Figure 9 in the manuscript was prepared using

```

plot_grid(smallInt.plt, brayaInt.plt, ncol = 1, labels = "auto",
          align = "hv", axis = "lr")

```

## 267 6 Derivatives of the estimated trend

268 The first derivative of the estimated trend is calculated using finite differences using the  
 269 `fderiv()` function. There is also a `confint()` method for objects produced by `fderiv()`. The

270 first derivatives and a 95% simultaneous confidence interval for the Small Water trend were  
271 computed and plotted using

```
small.d <- fderiv(mod, newdata = newYear, n = N)
small.sint <- with(newYear,
  cbind(confint(small.d, nsim = nsim,
    type = "simultaneous"),
  Year = Year))

small_deriv_plt <- ggplot(small.sint, aes(x = Year, y = est)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2,
    fill = "black") +
  geom_line() +
  labs(x = "Year CE", y = "First derivative")
```

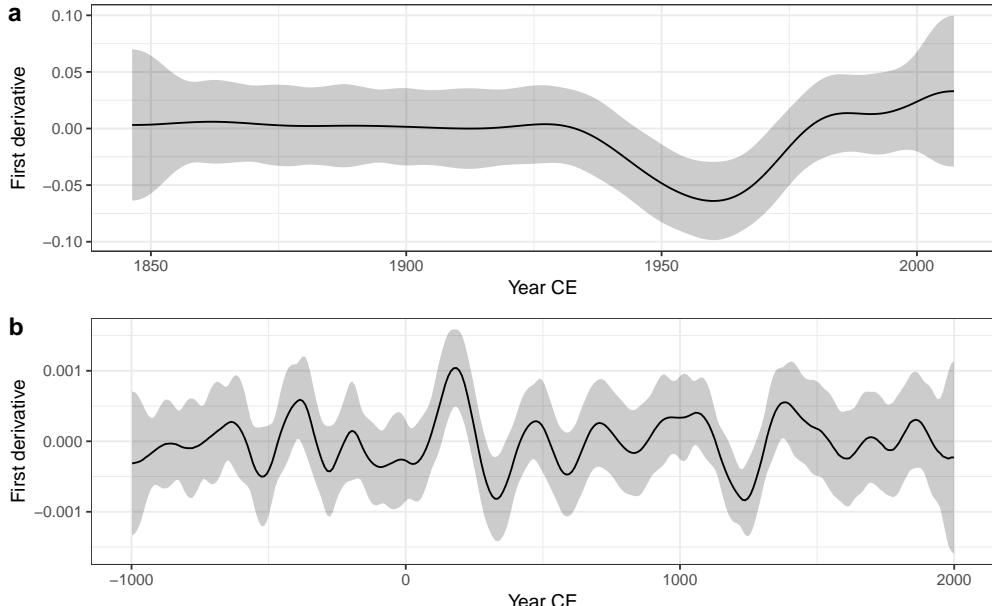
272 whilst for Braya-Sø, the following was used

```
braya.d <- fderiv(braya_reml, newdata = newBraya, n = N)
braya.sint <- with(newBraya,
  cbind(confint(braya.d, nsim = nsim,
    type = "simultaneous"),
  Year = Year))

braya_deriv_plt <- ggplot(braya.sint, aes(x = Year, y = est)) +
  geom_ribbon(aes(ymin = lower, ymax = upper),
    alpha = 0.2, fill = "black") +
  geom_line() +
  labs(x = "Year CE", y = "First derivative")
```

273 Figure 10 in the manuscript was prepared using

```
plot_grid(small_deriv_plt, braya_deriv_plt, ncol = 1, labels = "auto",
  align = "hv", axis = "lr")
```



274

## 275 7 Gaussian process smooths

276 For the Gaussian process smooth to fit within the GAM framework described in the  
 277 manuscript, we need to supply the value of  $\phi$  for the effective range of the correlation function.  
 278 To estimate  $\phi$ , we need to repeatedly fit the required GAM using a range of plausible values for  
 279  $\phi$ , which we do using a loop. In the chunk below I fit 200 models with  $\phi$  in the range 15–500.  
 280 For each value of  $\phi$  I fit the required GAM and extract the REML score (from component  
 281 `gcv.ubre`) and store it in the numeric vectors `Mat` or `SEx`, for Matérn and Squared Exponential  
 282 correlation functions, respectively. The final line of the chunk prepares the REML scores in  
 283 long or tidy format suitable for plotting with `ggplot2`.

```
nn <- 200      # number of points at which to evaluate profile likelihood
dseq <- seq(15, 500, length.out = nn) # effective ranges to fit at
Mat <- SEx <- numeric(length = nn)    # object to hold model fits

## loop over the sequence of separation distances and fit the models
for (i in seq_along(dseq)) {
  ## iterate over dseq, fit GP GAM w Matérn covariance
  Mat[i] <- gam(UK37 ~ s(Year, k = 45, bs = "gp", m = c(3, dseq[i])),
                 weights = sampleInterval / mean(sampleInterval),
                 data = braya, method = "REML",
                 family = gaussian())[[ "gcv.ubre" ]]

  ## fit squared exponential
  SEx[i] <- gam(UK37 ~ s(Year, k = 45, bs = "gp", m = c(2, dseq[i], 1)),
                 weights = sampleInterval / mean(sampleInterval),
                 data = braya, method = "REML",
```

```

        family = gaussian())[[ "gcv.ubre" ]]
}

## extract the REML score into ggplot-friendly object
reml.scr <- data.frame(cor = rep(c("Matérn", "Exponential"), each = nn),
                         effrange = rep(dseq, 2),
                         reml = c(Mat, SEx))

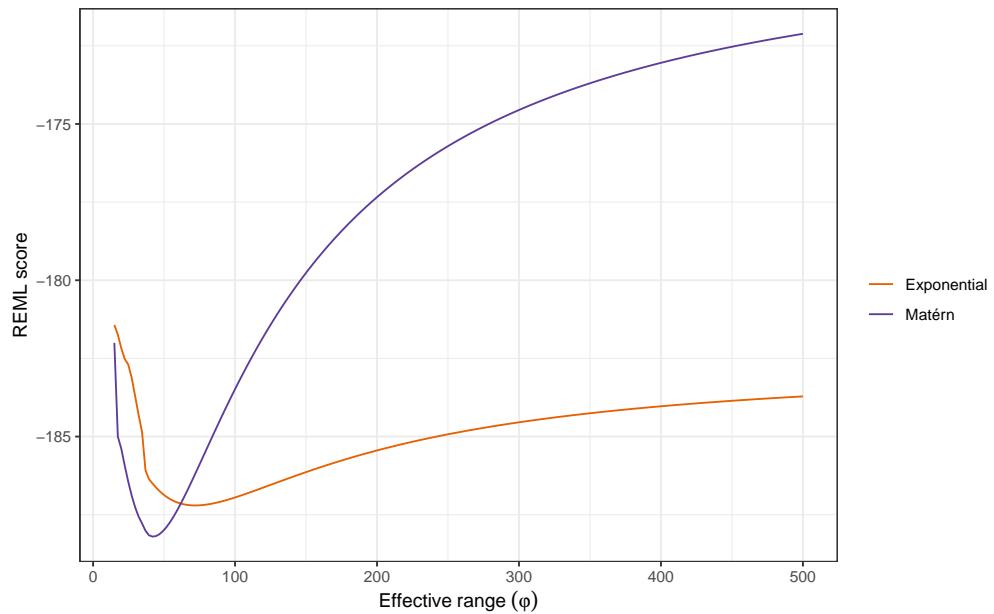
```

284 The REML scores for the models are plotted with *ggplot2* using

```

## profile-likelihood plot
proflik.plt <- ggplot(reml.scr, aes(x = effrange, y = reml, colour = cor)) +
  geom_line() +
  scale_colour_manual(name = "", values = c("#e66101", "#5e3c99")) +
  labs(y = "REML score", x = expression(Effective ~ range ~ (varphi)))
proflik.plt

```



285

286 Next we extract the minimum of the REML scores for the two correlation functions and refit  
 287 those models (we threw away all the models in the for () loop earlier to avoid storing lots  
 288 of model objects). Then we fit GAMs with Gaussian process smooths using the values of  $\phi$   
 289 that produced the minimum REML scores, and predict using the fitted models to visualize  
 290 the trends.

```

## effective range minima from profile likelihood
effRange2 <- with(subset(reml.scr, cor == "Matérn"), dseq[which.min(reml)])
effRange3 <- with(subset(reml.scr, cor == "Exponential"), dseq[which.min(reml)])

## Refit these models: Matern
gp2 <- gam(UK37 ~ s(Year, k = 45, bs = "gp", m = c(3, effRange2)),

```

```

    data = braya,
    method = "REML", weights = sampleInterval / mean(sampleInterval))
## Refit these models: Power exponential
gp3 <- gam(UK37 ~ s(Year, k = 45, bs = "gp", m = c(2, effRange3, 1)),
            data = braya,
            method = "REML", weights = sampleInterval / mean(sampleInterval))

## data to predict at
newd <- with(braya, data.frame(Year = seq(min(Year), max(Year),
                                             length.out = 1000)))
## create predictions on response scale for both covariance functions
p.gp2 <- transform(newd,
                     fitted = predict(gp2, newdata = newd, type = "response"),
                     effRange = round(effRange2))
p.gp3 <- transform(newd,
                     fitted = predict(gp3, newdata = newd, type = "response"),
                     effRange = round(effRange3))
## joint the two sets of predictions together
pred <- rbind(p.gp2, p.gp3)
## add some categorical variables for plotting
pred <- transform(pred,
                   effRange = factor(effRange),
                   cor = rep(c("Matérn", "Exponential"), each = nrow(newd)))

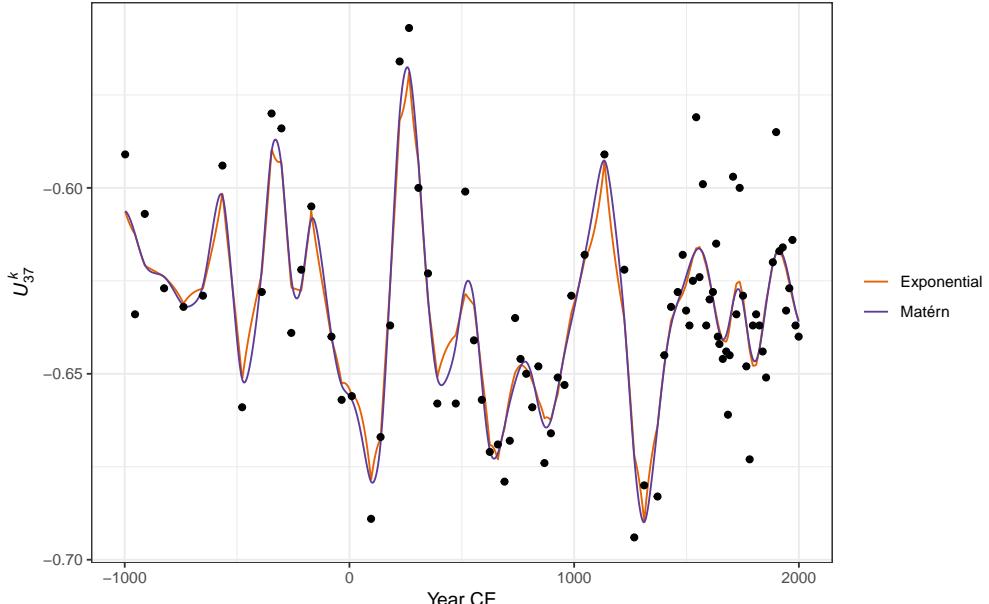
```

- 291 The estimated Gaussian process trends are plotted using

```

gp.plt2 <- ggplot(pred, aes(x = Year, y = fitted, colour = cor)) +
  geom_line() + theme(legend.position = "right") +
  geom_point(aes(x = Year, y = UK37), data = braya, inherit.aes = FALSE) +
  scale_colour_manual(name = "", values = c("#e66101", "#5e3c99")) +
  labs(y = braya_ylabel, x = "Year CE")
gp.plt2

```



292

293 whilst Figure 13 in the manuscript was prepared using

```
plot_grid(proflik.plt, gp=plt2, ncol = 1, labels = c("a", "b"),
          align = "hv", axis = "lr")
```

## 294 8 Adaptive smooths

295 The adaptive smooth was fitted to the Braya-Sø data by adding `bs = "ad"` to the `s()` term in  
 296 the model formula. The other aspects of the fit are as previously used for the other models,  
 297 REML smoothness selection and observational weights:

```
## Adaptive spline, weights as sampleInterval
mod_ad <- gam(UK37 ~ s(Year, k = 45, bs = "ad"), data = braya,
               method = "REML",
               weights = sampleInterval / mean(sampleInterval))
```

## 298 9 Comparing trends

299 For the model comparison I refitted all the models for consistency; the code to fit each of the

- 300 1. Thin plate regression spline (TPRS),  
 301 2. Gaussian process spline (Matérn correlation functions), and  
 302 3. Adaptive smoother

303 is shown below.

```

## effective range parameter from profile-likelihood of Matern model
effRange <- effRange2

## TPRS, weights as sampleInterval
mod_tprs <- gam(UK37 ~ s(Year, k = 45, bs = "tp"), data = braya,
                 method = "REML",
                 weights = sampleInterval / mean(sampleInterval))

## Gaussian process, Matern, kappa = 1.5, weights as sampleInterval
mod_gp <- gam(UK37 ~ s(Year, k = 45, bs = "gp", m = c(3, effRange)),
                data = braya,
                method = "REML",
                weights = sampleInterval / mean(sampleInterval))

## Adaptive spline, weights as sampleInterval
mod_ad <- gam(UK37 ~ s(Year, k = 45, bs = "ad"), data = braya,
                 method = "REML",
                 weights = sampleInterval / mean(sampleInterval))

```

304 I wrote a small function to predict from each model over the range of Year and return the data  
 305 in tidy format for plotting.

```

## wrap this in a function that will return all the plots & derived objects
processGAM <- function(mod) {
  ## Predict from model
  N <- 500
  newYear <- with(braya,
                   data.frame(Year = seq(min(Year), max(Year),
                                         length.out = N)))
  newYear <- cbind(newYear,
                    data.frame(predict(mod, newYear, se.fit = TRUE)))

  out <- list(objects = newYear)
  out
}

plts_gp   <- processGAM(mod = mod_gp) # Gaussian process smooth with weights
plts_ad   <- processGAM(mod = mod_ad) # Adaptive smooth with weights
plts_tprs <- processGAM(mod = mod_tprs) # TPRS with weights

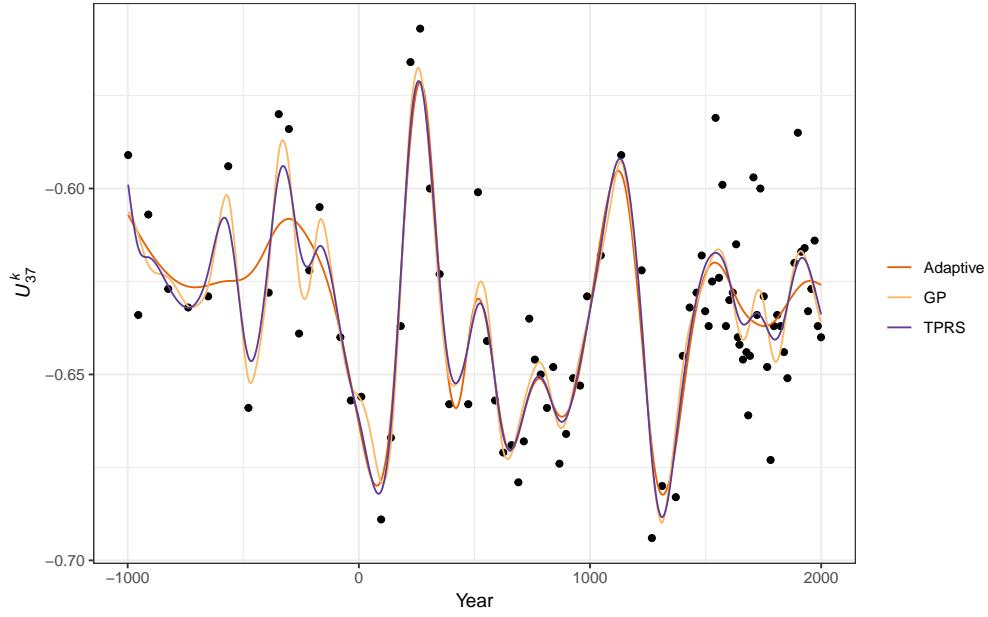
pltData <- do.call("rbind", lapply(list(plts_gp, plts_ad, plts_tprs),
                                     `[[~, "objects"]))
pltData <- transform=pltData,
           Model = rep(c("GP", "Adaptive", "TPRS"),
                        each = nrow(plts_gp$objects)))

```

```

allFits <- ggplot(pltData, aes(x = Year, y = fit)) +
  geom_point(aes(x = Year, y = UK37), data = braya) +
  geom_line(aes(colour = Model)) + labs(y = braya_ylabel, x = "Year") +
  theme(legend.position = "right") +
  scale_colour_manual(name = "",
    values = c("#e66101", "#fdb863", "#5e3c99"))
allFits

```



306

307 The plot produced reproduces Figure 14 in the manuscript.

## 308 10 Accounting for age-model uncertainty

309 The manuscript proposed to simulate from the posterior distribution of the fitted age model  
 310 as a way to account for age-model uncertainty. The first step in the process is to fit the age  
 311 model from which to simulate new age models. This was done using the *scam* package for a  
 312 *shape-constrained GAM*, with the age-model spline constrained to be monotonic decreasing (*bs*  
 313 = "mpd").

314 To make this section self-contained, I refitted the Small Water GAM plus CAR(1) model. Be-  
 315 cause we will be sampling ages for each observation from the posterior of the age-model GAM,  
 316 the oldest sample will be somewhat younger than the expected value from the model in  
 317 some samples. This will cause problems when we come to simulate from the GAMs fitted to  
 318 the resampled age-models as each GAM will cover a slightly different range of time. To avoid  
 319 this problem I fix the knots at the extremes of the observed values of Year and spread the re-  
 320 maining 12 knots evenly inbetween. The knot locations are stored in the vector *knots*, which  
 321 is passed to the argument *knots* when fitting the GAM

```

knots <- with(small, list(Year = seq(min(Year), max(Year), length = 14)))
mod <- gamm(d15N ~ s(Year, k = 15), data = small, method = "REML",
            correlation = corCAR1(form = ~ Year),
            knots = knots)

```

322 Setting the knots like this doesn't change the model estimated above; I'm only repeating what  
 323 happens internally if you don't supply knots. However, later we will need to specify this set  
 324 of knots explicitly when accounting for age model uncertainty.

325 Next we load the  $^{210}\text{Pb}$  dating results for the dated core sections.

```
swAge <- read.csv("./data/small-water/small1-dating.csv")
```

326 before fitting the shape-constrained GAM. Currently, *scam* can only fit models using GCV  
 327 smoothness selection. I used the *gamma* argument here to add a larger penalty for more-  
 328 complex models. Each effective degree of freedom used by the spline is counted as 1.4  
 329 degrees of freedom in the GCV score.

```

## monotonic spline age-depth model
swAge$Error[1] <- 1.1
swAgeMod <- scam(Date ~ s(Depth, k = 5, bs = "mpd"), data = swAge,
                  weights = 1 / swAge$Error, gamma = 1.4)

```

330 Note that I added a small amount of error to the surface sample age as the model cannot be  
 331 fitted if an observation has 0 weight.

332 Next, predict from the estimated age model, and draw 25 samples from the posterior distribution  
 333 using *simulate()*. The results are tidied into a format suitable for further processing and  
 334 plotting. Note that the posterior samples here are only used for plotting.

```

## predict from the age model for a smooth set of points in `Depth`
newAge <- with(swAge, data.frame(Depth = seq(min(Depth), max(Depth),
                                             length.out = 200)))
newAge <- transform(newAge,
                     fitted = predict(swAgeMod, newdata = newAge,
                                       type = "response"))
newSims <- as.data.frame(simulate(swAgeMod, nsim = 25, newdata = newAge))
newSims <- cbind(Depth = newAge$Depth, newSims)
newSims <- gather(newSims, Simulation, Age, -Depth)

```

335 In the next code chunk, I draw 100 samples from the posterior distribution of the age model,  
 336 but notice that I pass in the *small* data to *newdata* in the call to *simulate()* as the locations  
 337 I want new age estimates for are the depths for which we have  $\delta^{15}\text{N}$  values. A small function  
 338 (*fitSWModels*) is written to prepare each simulation for fitting and then actually fit the GAM  
 339 plus CAR(1) model using the updated age information.

```

## simulate from age model; each column is a simulation
ageSims <- simulate(swAgeMod, nsim = 100, newdata = small, seed = 42)

```

```

ageSims <- as.data.frame(ageSims)

fitSWModels <- function(x, y, knots) {
  dat <- data.frame(d15N = y, Year = x)
  m <- gamm(d15N ~ s(Year, k = 15), data = dat, method = "REML",
             correlation = corCAR1(form = ~ Year), knots = knots)
}

## generate new trends using draws from age-model posterior
simTrendMods <- lapply(ageSims, fitSWModels, y = small$d15N, knots = knots)

## function wrapper to predict new trends at locations over the
## range of `Year`
predSWModels <- function(mod, newdata) {
  predict(mod$gam, newdata = newdata, type = "response")
}

## predict from fitted model to produce a smooth trend for each posterior
## sample
simTrends <- lapply(simTrendMods, predSWModels, newdata = newYear)

## arrange in a tidy format for plottings
simTrends <- data.frame(Year = with(newYear, rep(Year, length(simTrends))),
                        Trend = unlist(simTrends),
                        Group = rep(seq_along(simTrends),
                                   times = lengths(simTrends)))

```

340 The next chunk does the final step in the process. For each of the models we just fitted, we  
 341 simulate 50 draws from the model posterior distribution. We start with a wrapper function  
 342 around the `simulate()` code we want to run on each model, then do the actual posterior draws  
 343 for each model using `lapply()`. The final step just arranges data for plotting.

```

## wrapper to simulate from a fitted GAM with the required arguments
simulateSWModels <- function(mod, newdata, nsim, seed = 42) {
  sims <- simulate(mod, nsim = nsim, newdata = newdata, seed = seed)
  as.vector(sims)
}

## now do the posterior simulation
NSIM <- 50      # number of posterior samples *per* model
simSimulate <- lapply(simTrendMods, simulateSWModels, newdata = newYear,
                      nsim = NSIM, seed = 42)

## arrange in a tidy format

```

```

simSimulate <-
  data.frame(Year = with(newYear,
                        rep(Year, times = NSIM * length(simSimulate))),
             Trend = unlist(simSimulate),
             Group = rep(seq_len(NSIM * length(simSimulate)),
                         each = nrow(newYear)))

```

- <sup>344</sup> Each of the steps is visualized using the plot code shown below.

```

## plot the estimated age model ad posterior simulations from it
plt1 <- ggplot(swAge, aes(y = Date, x = Depth)) +
  geom_line(data = newSims,
            mapping = aes(y = Age, x = Depth, group = Simulation),
            alpha = 1, colour = "grey80") +
  geom_line(data = newAge, mapping = aes(y = fitted, x = Depth)) +
  geom_point(size = 1.5, colour = "red") +
  geom_errorbar(aes(ymax = Date + Error, ymin = Date - Error, width = 0),
                colour = "red") +
  labs(y = "Year CE", x = "Depth (cm)")

## plot the simulated trends showing the effect of age-model uncertainty
plt2 <- ggplot(simTrends, aes(x = Year, y = Trend, group = Group)) +
  geom_line(alpha = 0.1, colour = "grey80") +
  geom_line(data = newYear,
            mapping = aes(x = Year, y = fit), inherit.aes = FALSE) +
  geom_point(data = small,
             mapping = aes(x = Year, y = d15N),
             inherit.aes = FALSE, size = 0.7) +
  labs(x = "Year", y = d15n_label)

## plot simulated trends showing the effect of age-model uncertainty and
## the effect of uncertainty in the estimated trend itself
plt3 <- ggplot(simSimulate, aes(x = Year, y = Trend, group = Group)) +
  geom_line(alpha = 0.2, colour = "grey80") +
  geom_point(data = small,
             mapping = aes(x = Year, y = d15N),
             inherit.aes = FALSE,
             size = 0.7) +
  geom_line(data = newYear,
            mapping = aes(x = Year, y = fit),
            inherit.aes = FALSE) +
  labs(x = "Year", y = d15n_label)

## align all plots vertically
plots <- align_plots(plt1, plt2, plt3, align = 'v', axis = 'l')

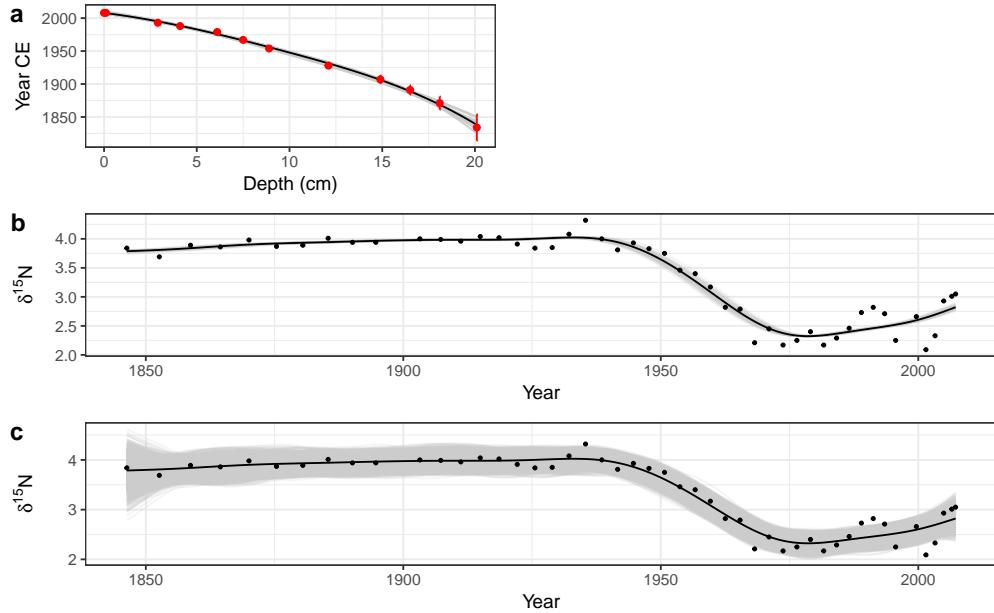
```

```

## create the two rows of figures, from `plots`
top_row <- plot_grid(plots[[1]], NULL, ncol = 2, labels = "a")
bot_row <- plot_grid(plots[[2]], plots[[3]], ncol = 1, labels = c("b", "c"))

## combine the two rows, top row has 1 plot row, bottom row has 2, hence
## the rel_heights to even this out
plot_grid(top_row, bot_row, ncol = 1, rel_heights = c(0.5, 1))

```



345

346 This reproduces Figure 15 from the manuscript.

## 347 11 Session information

```

devtools::session_info()

#> Session info -----
#>   setting  value
#>   version R version 3.4.3 Patched (2018-01-16 r74126)
#>   system   x86_64, linux-gnu
#>   ui        X11
#>   language (EN)
#>   collate  en_CA.UTF-8
#>   tz       America/Regina
#>   date     2018-08-16

```

```

357 #> Packages -----
358 #> package      * version date       source
359 #> assertthat    0.2.0   2017-04-11 CRAN (R 3.4.3)
360 #> backports     1.1.2   2017-12-13 CRAN (R 3.4.3)
361 #> base          * 3.4.3   2018-01-16 local
362 #> bindr         0.1.1   2018-03-13 CRAN (R 3.4.3)
363 #> bindrcpp      0.2.2   2018-03-29 CRAN (R 3.4.3)
364 #> codetools     0.2-15  2016-10-05 CRAN (R 3.4.3)
365 #> colorspace    1.3-2   2016-12-14 CRAN (R 3.4.3)
366 #> compiler      3.4.3   2018-01-16 local
367 #> cowplot        * 0.9.3   2018-07-15 CRAN (R 3.4.3)
368 #> crayon         1.3.4   2017-09-16 CRAN (R 3.4.3)
369 #> datasets      * 3.4.3   2018-01-16 local
370 #> devtools       1.13.6  2018-06-27 CRAN (R 3.4.3)
371 #> digest         0.6.15  2018-01-28 CRAN (R 3.4.3)
372 #> dplyr          0.7.6   2018-06-29 CRAN (R 3.4.3)
373 #> evaluate       0.11    2018-07-17 CRAN (R 3.4.3)
374 #> ggplot2        * 3.0.0   2018-07-03 CRAN (R 3.4.3)
375 #> glue           1.3.0   2018-07-17 CRAN (R 3.4.3)
376 #> graphics       * 3.4.3   2018-01-16 local
377 #> gratia         * 0.0-6   2018-08-16 local
378 #> grDevices      * 3.4.3   2018-01-16 local
379 #> grid            3.4.3   2018-01-16 local
380 #> gtable          0.2.0   2016-02-26 CRAN (R 3.4.3)
381 #> htmltools       0.3.6   2017-04-28 CRAN (R 3.4.3)
382 #> knitr          1.20    2018-02-20 CRAN (R 3.4.3)
383 #> labeling        0.3     2014-08-23 CRAN (R 3.4.3)
384 #> lattice         0.20-35  2017-03-25 CRAN (R 3.4.3)
385 #> lazyeval        0.2.1   2017-10-29 CRAN (R 3.4.3)
386 #> magrittr        1.5     2014-11-22 CRAN (R 3.4.3)
387 #> MASS            7.3-50  2018-04-30 CRAN (R 3.4.3)
388 #> Matrix          1.2-14  2018-04-09 CRAN (R 3.4.3)
389 #> memoise         1.1.0   2017-04-21 CRAN (R 3.4.3)
390 #> methods         3.4.3   2018-01-16 local
391 #> mgcv            * 1.8-24  2018-06-18 CRAN (R 3.4.3)
392 #> munsell         0.5.0   2018-06-12 CRAN (R 3.4.3)
393 #> nlme            * 3.1-137 2018-04-07 CRAN (R 3.4.3)
394 #> pillar          1.3.0   2018-07-14 CRAN (R 3.4.3)
395 #> pkgconfig       2.0.1   2017-03-21 CRAN (R 3.4.3)
396 #> plyr            1.8.4   2016-06-08 CRAN (R 3.4.3)
397 #> purrr           0.2.5   2018-05-29 CRAN (R 3.4.3)
398 #> R6               2.2.2   2017-06-17 CRAN (R 3.4.3)
399 #> Rcpp             0.12.18 2018-07-23 CRAN (R 3.4.3)
400 #> rlang            0.2.1   2018-05-30 CRAN (R 3.4.3)

```

```
401 #> rmarkdown * 1.10    2018-06-11 CRAN (R 3.4.3)
402 #> rprojroot 1.3-2    2018-01-03 CRAN (R 3.4.3)
403 #> scales      0.5.0   2017-08-24 CRAN (R 3.4.3)
404 #> scam        * 1.2-2   2017-09-24 CRAN (R 3.4.3)
405 #> splines     3.4.3   2018-01-16 local
406 #> stats       * 3.4.3   2018-01-16 local
407 #> stringi     1.2.4   2018-07-20 CRAN (R 3.4.3)
408 #> stringr     1.3.1   2018-05-10 CRAN (R 3.4.3)
409 #> tibble      1.4.2   2018-01-22 CRAN (R 3.4.3)
410 #> tidyverse    * 0.8.1   2018-05-18 CRAN (R 3.4.3)
411 #> tidyselect   0.2.4   2018-02-26 CRAN (R 3.4.3)
412 #> tools       3.4.3   2018-01-16 local
413 #> utils       * 3.4.3   2018-01-16 local
414 #> withr       2.1.2   2018-03-15 CRAN (R 3.4.3)
415 #> yaml        2.2.0   2018-07-25 CRAN (R 3.4.3)
```