Supplement 1: Analysing community change through time with palaeo data

Olivia Rata Burge

Manaaki Whenua - Landcare Research

1 Example dataset

This example uses data from Rotonuiaha, a site in the North Island of New Zealand. The data is described in Wilmshurst & McGlone (1996); the data here are count data, not relative abundance data. This example reproduces analytical steps from Olivia R. Burge, Richardson, Wood, & Wilmshurst (2023). This is primarily around implementing different parts of the code - note that where we use a linear model, you might consider a GAM fits your data better! Also note that we do not go into the specifics of assumption testing for each different model, aside from an illustration of temporal autocorrelation - please investigate the correct tests for whatever model you choose to implement.

1.1 Loading the data

The data is available from the `baselines` package: see here You need to have loaded (and potentially installed, if you haven't already) the package: `require(baselines)`. Note you may need to install other packages to install baselines!

```
data(rotonuiaha)
set.seed(888)

# metadata columns
metaCols <- names(rotonuiaha) %in% c("YEAR", "DEPTH", "SITE")

# metadata dataframe
imps <- rotonuiaha[metaCols]

# our time periods (refer paper)
imps$period <- with(imps,
                    ifelse(YEAR < 1280,
                           "Pre-Human",
                           ifelse(YEAR >= 1840,
                                  "Post-European",
                                  "Post-Polynesian")))
```

1.2 Other packages

You will need to load some other packages. How many depends on whether you are running the whole tutorial or just bit by bit. You will almost definitely need `vegan` and `dplyr`, `ggplot2` (or `tidyverse` if you have that already), you may need `boral` for the Bayesian ordination if you run that, `nlme` for the temporal autocorrelation model, `rgeos` for calculating distances, `gratia`, `analogue`, `broom`, and `MuMIn`. We've tried to specify which package is required where in the free text below too.

2 Calculating distance from baseline (ellipse; NMDS)

First we demonstrate calculating baseline incorporating variability using conventional (NMDS) methods. To do this we use the vegan package (Oksanen et al., 2019) with function `metaMDS`. We use the Jaccard distance and two dimensions (`k`). We were able to fit 2 dimensions for all sites; it will require some customisation to get distance from baseline in 3 dimensions, and we do not cover this edge case.

```
veg <- rotonuiaha[!metaCols]
tVeg <- decostand(decostand(veg, "total"), "log")
set.seed(888)
okMeta2 <- metaMDS(tVeg,
                  distance = "jaccard", k = 2, try = 200,
                  try.max = 700,
                  autotransform = FALSE)
set.seed(888)
okMeta <- metaMDS(tVeg,
                  distance = "jaccard", k = 2, try = 200,
                  try.max = 700,
                  autotransform = FALSE,
                  previous.best = okMeta2)
```

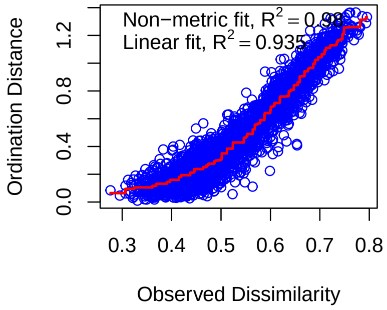

Figure 1: Stressplot for NMDS - shows ordination distance vs dissimilarity (in Jaccard distance, in our case)

We then use the `baselines` package (Olivia Rata Burge, 2022) to calculate the pre-human baseline. It also creates a spatial ellipse for the baseline and a spatial points object. Note that the metadata ('metadf') needs to have the same order as the data that went into the ordination! This requirement also applies to most of the functions in `vegan` too, by the way.

```
distsOut <- calcEllipseDists(metadf = imps,
                        ord = okMeta,
                        group = "period",
                        reflev = "Pre-Human")
```

```
# distsOut returns a list - what are the names
names(distsOut)
```

```
## [1] "distDF"              "baseline_polygon"    "all_points"
## [4] "baseline_polygon_DF"
```

The `distDF` is the dataframe with the original metadata supplied to the function, the NMDS scores for each point, the centroid for each period, and the distance from ellipse and baseline for the selected baseline level. This is what you will typically use. The `baseline_polygon` and `all_points` are spatial objects and can be used with base plotting methods, and other spatial functions. `baseline_polygon_DF` is the baseline ellipse converted to a dataframe for easier plotting with `ggplot2` (Wickham, 2016).

```
metaScores <- distsOut[["distDF"]] %>%
  mutate(period = as.factor(period))

head(metaScores, 2)
```

```
##        YEAR DEPTH       SITE    period      NMDS1       NMDS2  centroid1
## 1  84.83404   560 Rotonuiaha Pre-Human -0.4353609 -0.12586729 -0.2746514
## 2 129.42979   550 Rotonuiaha Pre-Human -0.4996545 -0.02659776 -0.2746514
##     centroid2 distEllipse disCentroid
## 1 -0.00574629           0   0.2006405
## 2 -0.00574629           0   0.2259671
```

Here we extract the `baseline_polygon_DF` to a new standalone object for more concise plotting code, and get the centroid scores for the pre-human period.

```
baselineEllipse <- distsOut[["baseline_polygon_DF"]]

# get centroid for pre-human ellipse

centroidCoord <- metaScores %>%
  filter(period == "Pre-Human") %>%
  slice(1) %>%
  dplyr::select(SITE, centroid1, centroid2)
```

2.1 Plot NMDS and baseline ellipse

We can now plot the results of the ordination. `coord_equal()` is required when plotting ordinations in `ggplot2` (Wickham, 2016) - base plot uses a 1:1 aspect ratio for the x and y axes automatically for ordinations.

```
ptSize <- 2.5
nmdsPlot <- ggplot() +
  coord_equal() +
  geom_hline(colour = "grey", linetype = "dashed",
             yintercept = 0) +
  geom_vline(colour = "grey", linetype = "dashed",
             xintercept = 0) +
  geom_polygon(data = baselineEllipse,
               aes(x = NMDS1, y = NMDS2,
                   fill = group),
               alpha = 0.5, show.legend = FALSE) +
  geom_point(data = metaScores,
             aes(x = NMDS1, y = NMDS2,
                 fill = period),
             size = ptSize, colour = "black", shape = 21,
             stroke = 0.8) +
    geom_text(data = centroidCoord,
            aes(x = centroid1, y = centroid2),
            label = "X",
            colour = "white",
            size = 12) +
  scale_fill_brewer("Time period",
                    palette = "YlGnBu",
                    direction = -1,
                    limits = c("Pre-Human",
                               "Post-Polynesian",
                               "Post-European"),
                    labels = c("Pre-Human",
                               "Post-Polynesian",
                               "Post-European")) +
  guides(fill = guide_legend(title.position = "top",
                             title.hjust = 0.5,
                             nrow = 2, byrow = TRUE))+
  theme_bw() +
  labs(tag = "(a)") +
  theme(legend.position = "bottom",
        panel.grid = element_blank(),
        plot.tag.position = "topleft")
```

2.2 Plot distance methods figure

We don't plot the nmds plot yet however; we also demonstrate the distance between each point and the reference baseline. The function `gDistance` (rgeos package) calculates the minimum distance from each of our points to the polygon automatically. To demonstrate, we convert the reference baseline polygon to points and then select the polygon-point with the minimum distance to our sample points and plot that. You will also need the sp package.

```r
# convert the polygon to points
baselinePoints <- SpatialPoints(distsOut[[
  "baseline_polygon"]]@polygons[[
    1]]@Polygons[[1]]@coords)

# extract all the points that are not
# pre-human
samplePoints <- distsOut[[
  "all_points"]][distsOut[[
    "all_points"]]$period != "Pre-Human", ]

# calculate distance from each sample point
# to each baseline point
dists <- gDistance(samplePoints, baselinePoints,
                   byid = TRUE)


# get the id of the closest point from the baseline
# to each sample point
theMins <- as.numeric(
  sapply(USE.NAMES = FALSE,
         1:length(samplePoints),
         function(x) which.min(dists[, x])))

# create a df for the arrows that go from each point
# to the closest point of the ellipse
# NB this is not required for your own analysis
segDF <- do.call(
  "rbind",
  lapply(1:length(samplePoints),
         function(x){
           data.frame(
             x1 = samplePoints@coords[x,1],
             y1 = samplePoints@coords[x,2],
             x0 = as.numeric(
               baselinePoints@coords[theMins[x], 1]),
             y0 = as.numeric(
               baselinePoints@coords[theMins[x], 2]))
         })
```

```
)


# plot the distance methods plot
distancePlot <- ggplot() +
  geom_point(data = as.data.frame(baselinePoints),
             aes(x = coords.x1, y = coords.x2),
             shape = 1) +
  geom_point(data = as.data.frame(samplePoints),
             aes(x = NMDS1, y = NMDS2,
                 shape = period),
             size = ptSize) +
  geom_hline(colour = "grey", linetype = "dashed",
             yintercept = 0) +
  geom_vline(colour = "grey", linetype = "dashed",
             xintercept = 0) +
  geom_segment(data = segDF,
               colour = "grey",
               aes(x = x0, y = y0,
                   xend = x1, yend = y1),
               arrow = arrow(length = unit(0.1, "inches"),
                             angle = 20)) +
    scale_shape_manual("Time period",
                       values = c(16, 17),
                    limits = c("Post-Polynesian",
                               "Post-European"),
                    labels = c("Post-Polynesian",
                               "Post-European")) +
  coord_equal() +
  theme_bw() +
  guides(shape = guide_legend(title.position = "top",
                              title.hjust = 0.5,
                              nrow = 2))+
  labs(tag = "(b)", x = "NMDS1", y = "NMDS2") +
  theme(legend.position = "bottom",
        panel.grid = element_blank(),
        plot.tag.position = "topleft")


# we use package egg to put the plots together.
# Others are fine too.
egg::ggarrange(nmdsPlot, distancePlot, nrow = 1,
               labels = c("a", "b"),
               label.args = list(gp = grid::gpar(
                 font = 2,
                 cex = 1.2)))
```

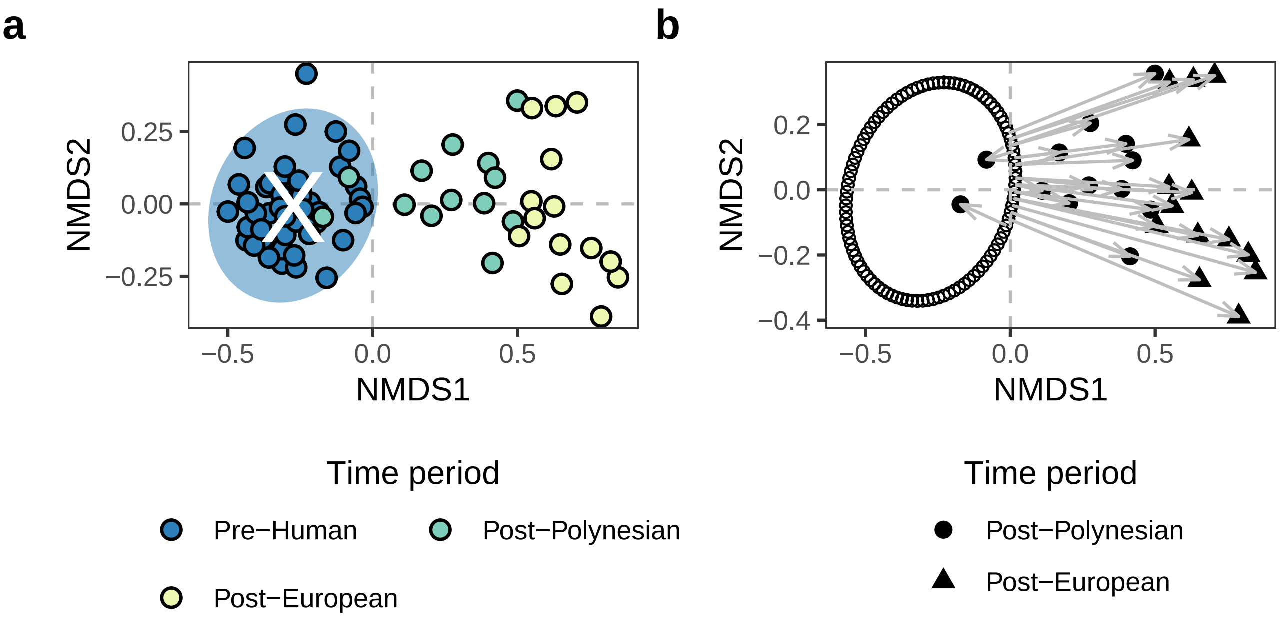

Figure 2: NMDS showing pre-human baselines of a 95% CI ellipse (blue polygon) and the centroid (white X) (a); and distance from both baselines (b). NB points within the polygon get a distance of zero

2.3 Distance from baseline over time: statistical model

Here we run a linear model, checking whether the post-Polynesian or post-European period distance from baseline differs to zero, and whether it changes over time. Read the paper (Wilmshurst & McGlone, 1996) to see why that might be!

```
excludingPreHuman <- metaScores[
  metaScores$period != "Pre-Human", ]

lmModEllipse <- lm(data = excludingPreHuman,
   formula = distEllipse ~ YEAR * period,
   na.action = "na.fail")
```

```
modComp <- dredge(lmModEllipse)

knitr::kable(modComp %>%
               as.data.frame() %>%
               mutate(across(where(is.numeric),
                         \(x) sprintf(x,
                         fmt = "%.3f"))) %>%
               rename(Intercept = `(Intercept)`),
             caption = "Model comparison table for distance from baseline over time. The year column is
```

Table 1: Model comparison table for distance from baseline over time. The year column is the coefficient estimate for year.

	Intercept	period	YEAR	period:YEAR	df	logLik	AICc	delta	weight
3	-1.268	NA	0.001	NA	3.000	25.969	-44.894	0.000	0.548
8	-3.416	+	0.002	+	5.000	28.209	-43.562	1.333	0.281
4	-1.103	+	0.001	NA	4.000	26.194	-42.570	2.324	0.171
2	0.670	+	NA	NA	3.000	13.918	-20.793	24.101	0.000
1	0.480	NA	NA	NA	2.000	-0.293	5.085	49.980	0.000

```
# preferred model
simpleLM <- lm(data = excludingPreHuman,
   formula = distEllipse ~ YEAR)
```

The diagnostic plots from the model look pretty good (Fig. 3; just one of several plots shown).

```
plot(lmModEllipse, which = 1)
```

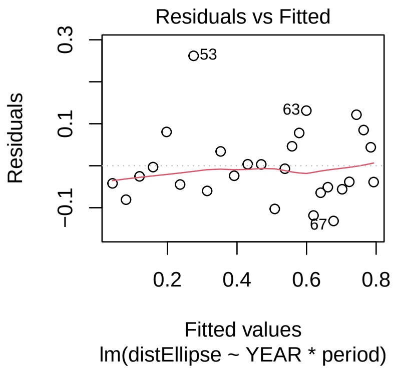

Figure 3: Diagnostic plot from linear model

We inspect the residuals of the linear model for temporal autocorrelation (Fig. 4). These look ok (you don't consider lag = 0!), so we won't go on to fit a model with autocorrelation. We demonstrate how to compare models with lag 1 and lag 2 autocorrelations using the `gls` function from package `nlme`

```
acf(resid(simpleLM), main = "")
```

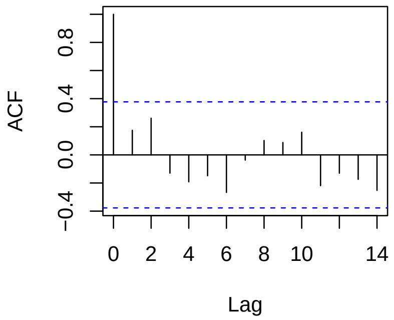

Figure 4: Checking for temporal autocorrelation in model residuals

2.4 Demonstration of a gls model

In this case, had we used a gls model, it would have been slightly more conservative in terms of *T* values but no drastic changes. We show a gls model with lag 1 and lag 2 temporal autocorrelation, and a 'null' model of no temporal autocorrelation - but all have the same fixed effect of year. As the fixed effects are the same we compare using REML (if they differed, ML would be a better choice see discussion here).

```
# model with lag 1 correlation
simpleGLS <- gls(distEllipse ~ YEAR,
                           data = excludingPreHuman,
                 correlation = corARMA(p = 1),
                  method = "REML",
                 na.action = "na.fail")


# same model but with lag 2 correlation
simpleGLS2 <- update(simpleGLS,
                     correlation = corARMA(p = 2))

# same model but with no temporal autocorrelation
simpleGLS0 <- update(simpleGLS,
                     correlation = NULL)

# compare models
anova(simpleGLS, simpleGLS2)
```

```
##              Model df       AIC       BIC   logLik   Test  L.Ratio p-value
## simpleGLS        1  4 -22.35919 -17.48369 15.17960
## simpleGLS2       2  5 -23.06715 -16.97277 16.53357 1 vs 2 2.707956  0.0998
```

```
anova(simpleGLS, simpleGLS2)
```

```
##            Model df       AIC       BIC   logLik   Test  L.Ratio p-value
## simpleGLS      1  4 -22.35919 -17.48369 15.17960
## simpleGLS2     2  5 -23.06715 -16.97277 16.53357 1 vs 2 2.707956  0.0998
```

```
# combine model summaries into dataframes
# to print as a table
glsSummary <- data.frame(Model = "gls",
                         summary(simpleGLS)$tTable) %>%
  rownames_to_column("Parameter") %>%
  rename(Estimate = Value)

lmSummary <- data.frame(Model = "lm",
                        summary(simpleLM)$coefficient) %>%
  rownames_to_column("Parameter") %>%
  rename(p.value = Pr...t..,
       Std.Error = Std..Error)

knitr::kable(bind_rows(glsSummary, lmSummary) %>%
               mutate(p.value = sprintf(p.value, fmt = "%.3f")),
                     digits = 3,
             caption = "GLS and LM model summaries")
```

Table 2: GLS and LM model summaries

Parameter	Model	Estimate	Std.Error	t.value	p.value
(Intercept)	gls	-1.272	0.185	-6.880	0.000
YEAR	gls	0.001	0.000	9.546	0.000
(Intercept)	lm	-1.268	0.144	-8.808	0.000
YEAR	lm	0.001	0.000	12.243	0.000

2.5 Interpretation of distance-from-baseline model

For Rotonuiaha, we see that the intercept differs significantly to zero, which, where we have a continuous predictor only, can be interpreted as the model estimate when the continuous predictor (year) is set to zero. This is not particularly meaningful in our case. However, the year estimate means that for every 1 unit increase in year (i.e. 1 year), there is a 0.001 increase in ecological (ordination) distance from baseline, thus we see that distance from baseline is increasing from time. We see the fitted relationship and raw data in Fig. 5. Code to make Fig. 5 is set out below.

```
newYears <- data.frame(
  YEAR = seq(from = min(excludingPreHuman$YEAR),
             to = max(excludingPreHuman$YEAR),
             by = 1)
  )

modelPreds <- predict(simpleLM, newdata = newYears,
                      interval = "confidence",
                      level = 0.95,
                      se.fit = TRUE)
newYears <- data.frame(newYears, modelPreds[[1]])

ggplot(newYears, aes(x = YEAR, y = fit)) +
    geom_point(data = excludingPreHuman,
               aes(y = distEllipse),
             shape = 1) +
  geom_ribbon(aes(ymin = lwr, ymax = upr),
              fill = adjustcolor("grey50", 0.4),
              linetype = "dashed") +
  geom_line() +
  theme_classic() +
  labs(x = "Year",
       y = "Distance from baseline in ordination space")
```

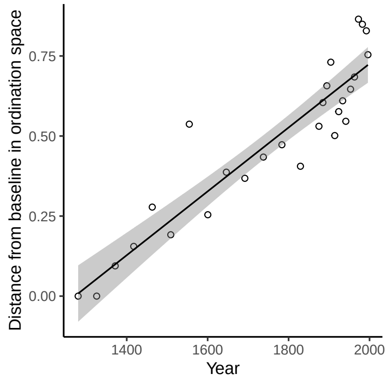

Figure 5: Predictions from linear model

3 Calculating distance from baseline (ellipse; Bayesian)

We set out the code rather more briefly for calculating the ordination distance from reference baseline (i.e. an ellipse) for a Bayesian analysis using the `boral` package (Hui, 2016). The only differences are in the calculation of the ellipse (code in `baselines` package Olivia Rata Burge (2022)) and in the ordination itself.

Here we use a negative binomial family (poisson also possible, among others) and allow for 2 dimensions with the argument `lv.control = list(num.lv = 2)`, and a random effect for each row (plot).

The diagnostic plots can be examined to check for fit - ideally there is no 'funnel' effect in the residuals. See Fig 6 (poisson) and Fig 7 (negative binomial) below.

```
# this is not run in an interactive session,
# as it takes a long time.
# If the rdata objects (below) are not in
# your working directory
# you will need to run this code.
```

```
if(!interactive()){
  boralMod  <- boral(y = rotonuiaha[!metaCols],
                     family = "negative.binomial",
                     lv.control = list(num.lv = 2),
                     mcmc.control = list(seed = 888),
                     row.eff = "fixed",
                     save.model = FALSE,
                     calc.ics = FALSE)

  boralMod2  <- boral(y = rotonuiaha[!metaCols],
                      family = "poisson",
                      lv.control = list(num.lv = 2),
                      mcmc.control = list(seed = 888),
                      row.eff = "fixed",
                      save.model = FALSE,
                      calc.ics = FALSE)

  save(boralMod, file = "boralMod1.Rda")
  save(boralMod2, file = "boralMod2.Rda")

}
```

```
if(interactive()){
  load("boralMod1.Rda")
  load("boralMod2.Rda")
}
```

```
par(mfrow = c(2,2))

plot(boralMod2)
```

```
par(mfrow = c(2,2))

plot(boralMod)
```

```
par(mfrow = c(1,1))
```

We will use the negative binomial which looks better in terms of the diagnostic plots (note the ‘funnel’ effect with the poisson model 6). Note that we set the random seed within the function. If you amend this, there will be slight differences in the positioning of the points.

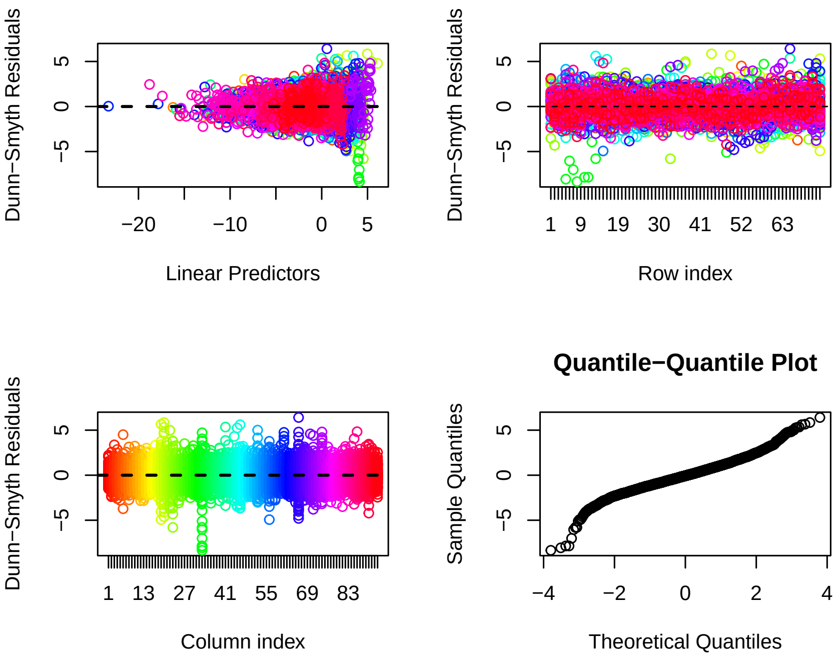

Figure 6: Boral model with poisson error structure

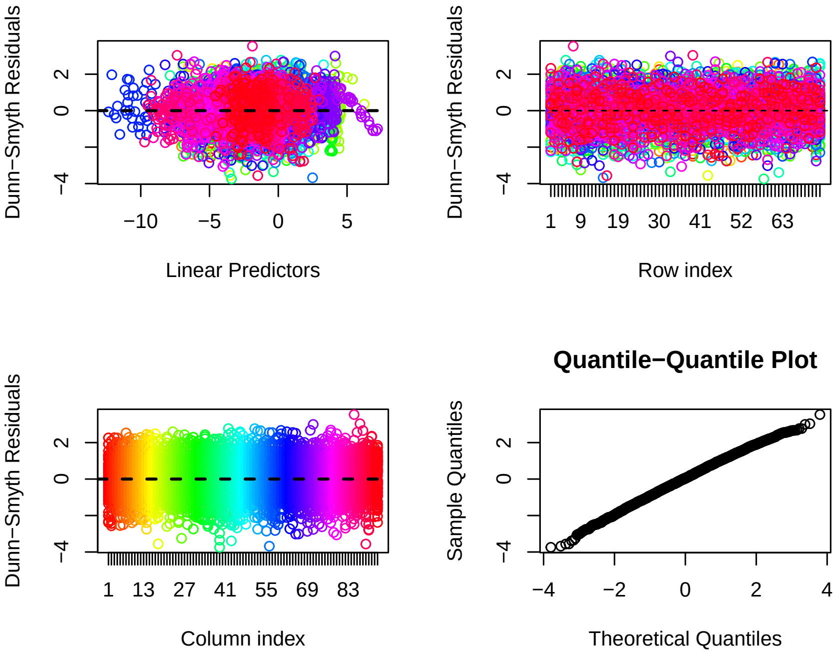

Figure 7: Boral model with negative binomial error structure

3.1 Distance from baseline

We can calculate distance from baseline in a similar way to that with an NMDS ordination (see Fig. 8).

```
# get distances
boralDists <- calcEllipseDists(metadf = imps,
                               ord = boralMod,
                               group = "period",
                               reflev = "Pre-Human")

# extract scores
boralDistDF <- boralDists[["distDF"]]
boralRefPolygon <- boralDists[["baseline_polygon_DF"]]

# make plot
boralOrdPlot <- ggplot(boralDistDF,
                       aes(x = lvs1, y = lvs2)) +
  coord_equal() +
  geom_polygon(data = boralRefPolygon,
               aes(group = group),
               fill = adjustcolor("grey", 0.5)) +
  geom_point(aes(fill = period),
             shape = 21,
             size = 2,
             colour = "black") +
  scale_fill_brewer("Time period",
                    palette = "YlGnBu",
                    direction = -1,
                    limits = c("Pre-Human",
                               "Post-Polynesian",
                               "Post-European"),
                    labels = c("Pre-Human",
                               "Post-Polynesian",
                               "Post-European")) +
  theme_bw() +
  labs(x = "Dim1", y = "Dim2") +
  guides(fill = guide_legend(nrow = 2,
                             title.position ="top",
                             byrow = TRUE)) +
  theme(legend.position = "top",
        panel.grid = element_blank())
boralOrdPlot
```

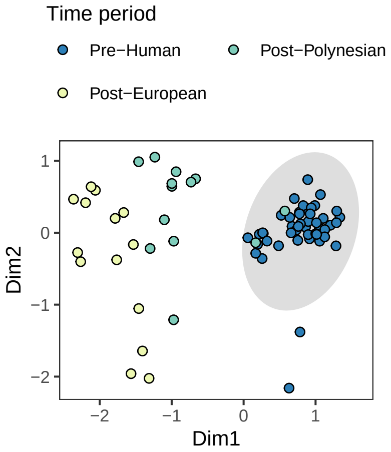

Figure 8: Boral ordination with baseline ellipse shown in grey.

3.2 Distance from baseline over time - statistical model

Overall, the ordination looks very similar to the NMDS ordination, so we do not expect anything to change substantially with the models.

```
excludingPreHumanBoral <- boralDistDF[
  boralDistDF$period != "Pre-Human", ]

lmModEllipseBoral <- lm(data = excludingPreHumanBoral,
   formula = distEllipse ~ YEAR * period,
   na.action = "na.fail")
```

Here we see actually a difference between the NMDS and the boral methods, in terms of what the best model is. With the boral method, we see that the favoured model is one that includes year, time period, and an interaction between the two. Using the NMDS method, we found the favoured model included year only. Two key points here: the first is that the in both cases the 'other' model is very close to the favoured model - delta AICc is 2 or less in both cases. Second, in this case you might up the number of simulations that boral uses or try several different random seeds (by setting the seed within the function) to see if that affects results.

```
# compare all subsets of global model
modTabBoral <- dredge(lmModEllipseBoral)
knitr::kable(model.sel(modTabBoral) %>%
               as.data.frame() %>%
           mutate(across(where(is.numeric),
                         \(x) sprintf(x,
                         fmt = "%.3f"))) %>%
               rename(Intercept = `(Intercept)`),
             digits = 3,
             caption = "Model comparison - distance from baseline")
```

Table 3: Model comparison - distance from baseline

	Intercept	period	YEAR	period:YEAR	df	logLik	AICc	delta	weight
8	-10.592	+	0.006	+	5.000	5.771	1.315	0.000	0.662
3	-2.978	NA	0.003	NA	3.000	1.968	3.107	1.792	0.270
4	-3.049	+	0.003	NA	4.000	1.975	5.868	4.553	0.068
2	1.964	+	NA	NA	3.000	-12.723	32.489	31.174	0.000
1	1.496	NA	NA	NA	2.000	-25.472	55.444	54.128	0.000

The model diagnostics look like some further work needs to be done! The earliest two points have zero distance (no distance from baseline - they fall within the ellipse on Fig. 8) and are affecting the slope of the line for the first time period. We could (although don't) consider further whether we want the average rate of change (as now) or the rate of change *once the ecosystem starts to be affected* i.e. by excluding the earliest two points. Best to be explicit either way. As for the NMDS model, the autocorrelation looks fine.

```r
# favoured model
simpleLMBoral <- lm(data = excludingPreHumanBoral,
   formula = distEllipse ~ YEAR * period)

# set up 3 plots in a row
par(mfrow = c(1, 3))

# first plot - raw vs fitted
with(excludingPreHumanBoral,
     plot(x = YEAR, y = distEllipse))
title(main = "Fitted relationship to raw data")
pp <- excludingPreHumanBoral$period=="Post-Polynesian"
with(excludingPreHumanBoral[pp,],
     lines(x = YEAR, y = fitted(simpleLMBoral)[pp]))
with(excludingPreHumanBoral[!pp,],
     lines(x = YEAR, y = fitted(simpleLMBoral)[!pp]))

# second plot - residuals vs fitted
plot(simpleLMBoral, which = 1,
     caption = "")
title(main = "Residuals v fitted")

# third plot - autocorrelation.
acf(residuals(simpleLMBoral), main = "Autocorrelation plot")
```

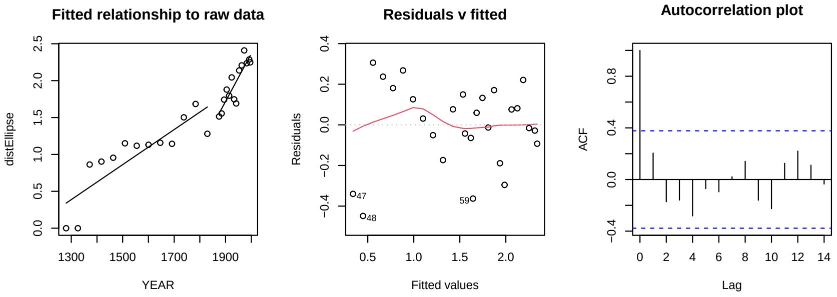

Figure 9: Model inspection

```
# return to one plot per line
par(mfrow = c(1, 1))
```

Although we come to non-linear models (GAMs) later, there are several other options: use a model accounting for temporal autocorrelation (gls; above) or modify the year term (e.g. include a quadratic term, or similar).

4 Distance from start point

This approach is particularly useful where the pre-disturbance period is unknown. For an example of published application, see Magurran, Dornelas, Moyes, Gotelli, & McGill (2015). Having decided on a distance metric (e.g., Bray-Curtis, Jaccard), and having ordered the dataset by depth (important, for the function requires it), we can calculate the distance from start point.

You can also calculate this yourself. It is easiest to do so by converting the distance object created by vegdist (or dist) to a matrix (as.matrix()), which for n samples, will by a matrix of n rows and n columns, with 0 on the diagonal (indicating that a sample is no different to itself). Here we give the species data as the rotonuiaha dataframe, but remove the SITE, YEAR, and DEPTH columns, as these are not species data, but rather metadata. We include the metadata using the imps dataframe we created earlier. The idCol (which will be typically either a year or depth column) is given by YEAR in our case and we have selected the Jaccard distance with which to calculate ecological distance. The idColType is set to 'year' appropriately too (change this if you are using depth).

```
pkStart <- calculateDistanceStart(
  speciesData = decostand(decostand(
    rotonuiaha[!metaCols],
    "total"), "log"),
  metaData = imps,
  idCol = "YEAR",
  distMethod = "jaccard",
  idColType = "YEAR")
```

pkStart is the object returned by calculateDistanceStart and has all the columns of the metadata dataframe imps, and one extra column: the distance from start (dist_jaccard). The earliest sample will always have a distance of 0 (we are comparing it to itself) and is removed using the notation pkStart[-1,], which removes the first row.

Before we get to running a model you can see that prior to human arrival, distance from start averaged just below 0.5; increasing in the post-Polynesian period and remaining high in the post-European period (Fig. 10).

```
ggplot(pkStart[ -1, ], aes(x = YEAR, y = dist_jaccard)) +
  geom_vline(xintercept = 1280, linetype = "dashed",
             colour = "grey") +
  geom_vline(xintercept = 1840, linetype = "dashed",
             colour = "grey") +
  geom_point(shape = 1) +
  geom_line() +
  coord_cartesian(ylim= c(0, 1)) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  labs(x = "Year", y = "Ecological distance (Jaccard)")
```

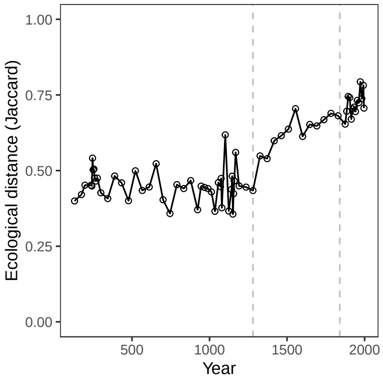

Figure 10: Distance from initial sample. Dashed lines indicate dates of Polynesian and Euro settlement

4.1 GAM on distance from start

Note that there are a few points which look a bit squiffy on the diagnostic plots. Without allowing the fitted smooth to 'wiggle' wildly to take these into account (or some other transformation of the data), these will persist.

```
m2 <- gamm(dist_jaccard ~  s(YEAR, bs = "cr", k = 10),
           correlation = corCAR1(form = ~ YEAR),
           data = pkStart[ -1, ])

par(mfrow = c(2,2))

gam.check(m2$gam)
```

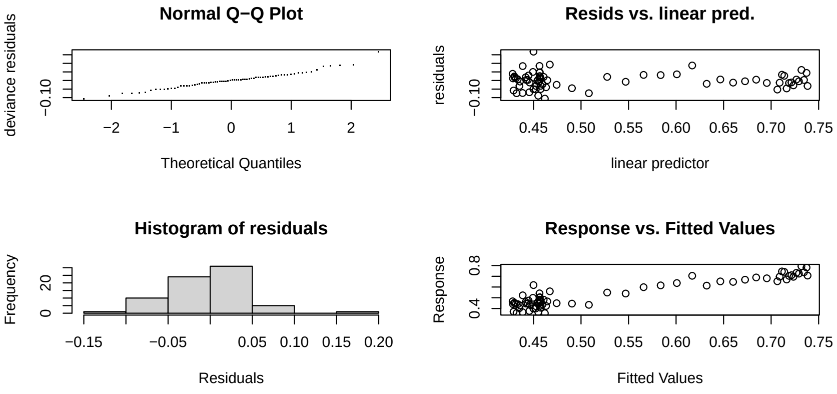

Figure 11: Diagnostic plots from GAMM model

```
##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## 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(YEAR) 9.00 4.48    1.14    0.89
```

```
par(mfrow = c(1,1))
```

Below is the code to plot the raw and fitted data for Fig. 12.

```
with(pkStart[ -1, ], plot(x = YEAR, y = dist_jaccard,
                          xlab = "Year",
                          ylab = "Distance",
                          col = "grey50"))
lines(y = predict(m2$gam), x = pkStart[-1, "YEAR"],
      col = "darkorange",
      lwd = 2)
```

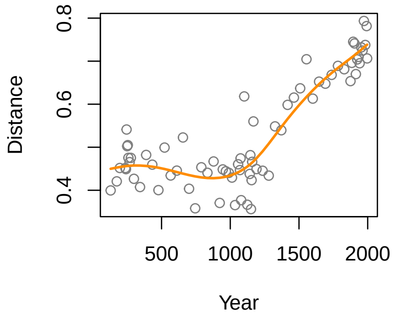

Figure 12: Distance from initial sample - modelled and raw

4.2 Identifying periods of rapid change

Here, we identify periods of rapid change, that is, where the first derivative (i.e. slope) of the fitted relationship from the GAMM is significantly different to zero. This doesn't mean that there is a difference from baseline - it is the rate of change in the distance that is significant.

To calculate where the first derivative differs from zero, we use the gratia package (Simpson, 2020). We can plot the derivative with confidence intervals (see Fig. 13). You can see that it only differs to zero at a time period centred around 1500. This correlates with the increase in distance-from-baseline beginning just prior and increasing rapidly before plateauing.

```
newDat <- data.frame(YEAR = seq(
  from = ceiling(min(pkStart[-1, "YEAR"])),
  to = floor(max(pkStart[-1, "YEAR"])),
  by = 1))

# calculate the derivatives of the model
# from gratia package
derivs <- fderiv(m2, newdata = newDat)
# calculate confidence intervals on the derivs
derivsCont <- data.frame(newDat,
                         confint(derivs,
                                 type = "simultaneous"))
```

```
ggplot(derivsCont, aes(x = YEAR, y = est)) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_line(aes(y = lower),linetype = "dashed") +
  geom_line(aes(y = upper),linetype = "dashed") +

  geom_point() +
  geom_point(data = derivsCont  %>%
               filter(!(upper > 0 & lower < 0)),
             aes(colour = "Slope different to zero")) +
  theme_bw() +
  scale_colour_manual(
    values = c("red2", "black"),
    limits = c("Slope different to zero",
               "Slope not different to zero")) +
  theme(legend.position = "bottom",
        panel.grid = element_blank(),
        legend.title = element_blank(),
        legend.direction = "vertical")
```

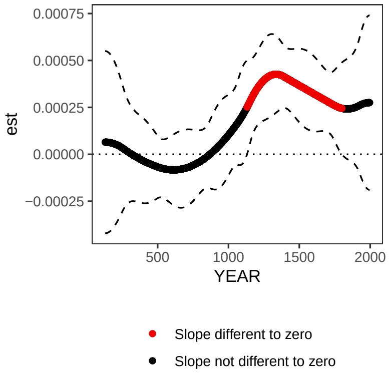

Figure 13: First derivative with confidence intervals

This period of rapid change can be overlain on the previous figure, which indicated change over time (Fig. 14, below).

```
preds <- data.frame(newDat,
                    predicted = predict(m2$gam,
                                        newdata = newDat))


# take the derivatives and
# from: https://gist.github.com/gavinsimpson/ca18c9c789ef5237dbc6
signifD <- function(x, d, upper, lower, eval = 0) {
  miss <- upper > eval & lower < eval
  incr <- decr <- x
  want <- d > eval
  incr[!want | miss] <- NA
  want <- d < eval
  decr[!want | miss] <- NA
  list(incr = incr, decr = decr)
}

sizes <- signifD(derivsCont$YEAR,
                 d = derivsCont$est,
                 upper = derivsCont$upper,
                 lower = derivsCont$lower)
```

```
predSigs <- data.frame(
  year = preds$YEAR,
  predicted = preds$predicted,
  increasing = sizes[["incr"]],
  decreasing = sizes[["decr"]]
  )

ggplot(preds, aes(x = YEAR, y = predicted)) +
  geom_vline(xintercept = 1280, linetype = "dashed",
             colour = "grey") +
  geom_vline(xintercept = 1840, linetype = "dashed",
             colour = "grey") +
  geom_line() +
  theme_bw() +
  geom_path(data= predSigs,
            aes(x = decreasing,
                colour = "Period of rapid change"),
            size = 2) +
  geom_path(data= predSigs,
            aes(x = increasing,
                colour = "Period of rapid change"),
            size = 2) +
  labs(x = "Year", y = "Distance from start") +
  scale_colour_manual(values = "red2") +
  geom_point(data = pkStart[ -1, ],
             aes(x = YEAR, y = dist_jaccard),
             shape = 1) +
  theme(legend.title = element_blank(),
        legend.position = c(0.02, 0.98),
        legend.justification = c(0, 1),
        panel.grid = element_blank())
```

5 Principal curves

Principal curves are an ordination technique which are a one-dimensional curve fitted through multidimensional datasets[1]. Here we fit one to the Rotonuiaha site. We transform the data using a Hellinger transformation (see `?decostand` for details).

```
prc1 <- prcurve(decostand(rotonuiaha[!metaCols],
                          "hellinger"),
                trace = FALSE, plotit = FALSE,
                maxit = 500)
```

[1] see excellent blog post by Gavin Simpson here

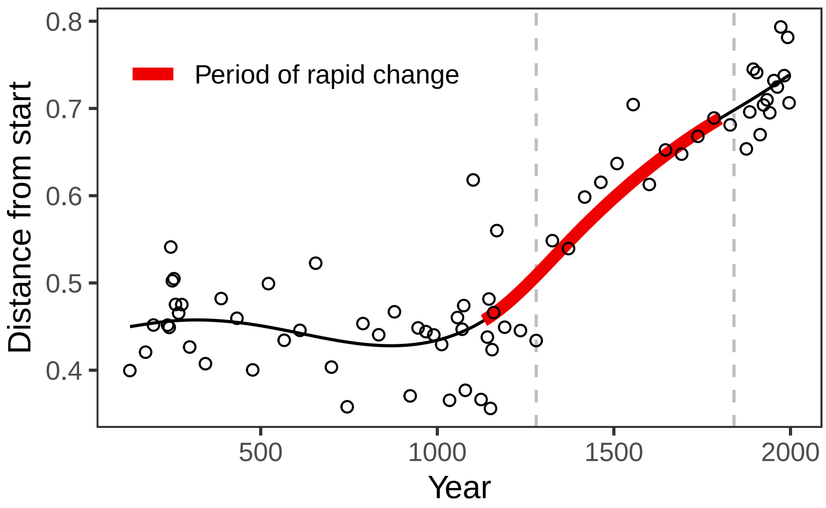

Figure 14: GAMM model with period of rapid change highlighted in red

```
prc1
```

```
##
##  Principal Curve Fitting
##
## Call: prcurve(X = decostand(rotonuiaha[!metaCols], "hellinger"), maxit
## = 500, trace = FALSE, plotit = FALSE)
##
## Algorithm converged after 5 iterations
##
##          SumSq Proportion
## Total     20.25       1.00
## Explained 15.19       0.75
## Residual   5.06       0.25
##
## Fitted curve uses 541.194 degrees of freedom.
```

```
prcurveScores <- data.frame(imps, scores(prc1))
```

The principal curve scores in raw form indicate that there was periods of rapid change after Polynesian and European settlement - and substantial, but short-lived, change at the time of the Taupō eruption (Fig. 15).

```
ggplot(prcurveScores,
       aes(x = YEAR, y = PrC)) +
  geom_vline(xintercept = 232, linetype = "dotdash",
             colour = "darkorange") +
  geom_vline(xintercept = 1280, linetype = "dashed",
             colour = "grey50") +
  geom_vline(xintercept = 1840, linetype = "dashed",
             colour = "grey50") +
  geom_point(shape = 16) +
  geom_line() +
  theme_bw()  +
  theme(panel.grid = element_blank())
```

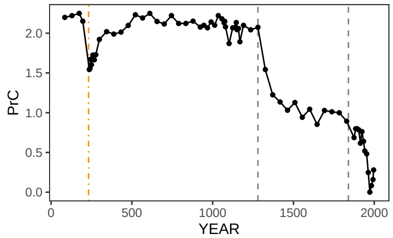

Figure 15: Principal curve scores over time. Orange line indicates the Taupō eruption.

We now use a gam to test for where periods of rapid change occur. This follows the same process as above for distance from start.

```
prcurveScores$year <- round(prcurveScores$YEAR, 0)

prGamm <- gamm(PrC ~  s(YEAR, bs = "cr", k = 20),
               correlation = corCAR1(form = ~ year),
               # method = "REML",
               data = prcurveScores)

par(mfrow = c(2,2))
gam.check(prGamm$gam)
```

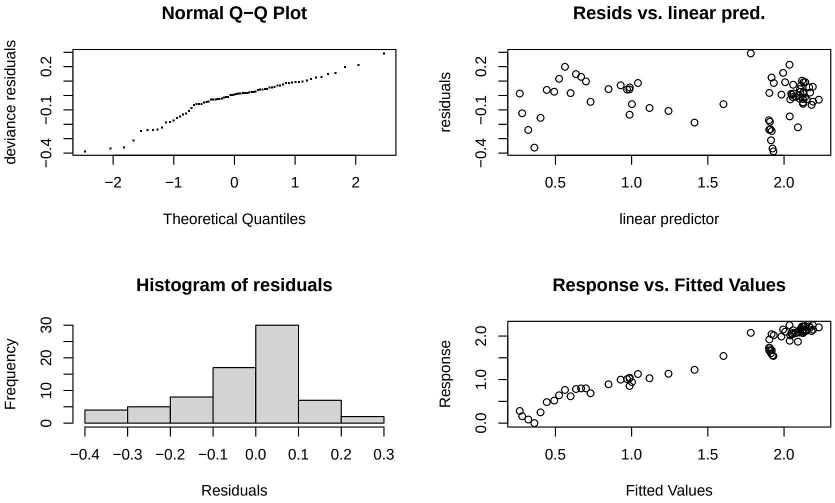

Figure 16: PrC GAMM diagnostics

```
##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## 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(YEAR) 19.0  8.9    0.45  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow = c(1,1))
```

```
GAMMsterPlot <- ggplot(prcurveScores, aes(x = year, y = PrC)) +
  geom_point() +
  theme_bw() +
  geom_line(y = fitted(prGamm$gam), colour = "red2")
```

We see that the GAMM model is a poor fit for the period of rapid change around the Taupō eruption; we can try a GAM model with adaptive splines to address this.

```
prGam <- gam(PrC ~  s(year, bs = "ad", k = 20),
               # correlation = corCAR1(form = ~ year),
               method = "REML",
               data = prcurveScores)



GAMsterPlot <- ggplot(prcurveScores, aes(x = year, y = PrC)) +
  geom_point() +
  theme_bw() +
  geom_line(y = fitted(prGam), colour = "royalblue")

grid.arrange(GAMMsterPlot + ggtitle("(A) GAMM plot"),
             GAMsterPlot + ggtitle("(B) GAM plot"),
             nrow = 1)
```

```
mod <- prGamm$gam

pcData <- data.frame(
  YEAR = seq(from = ceiling(min(imps$YEAR)),
             to = floor(max(imps$YEAR)),
             by = 1)
)
predPC <- data.frame(pcData,
                     predicted = predict(mod,
                                         newdata = pcData))
```

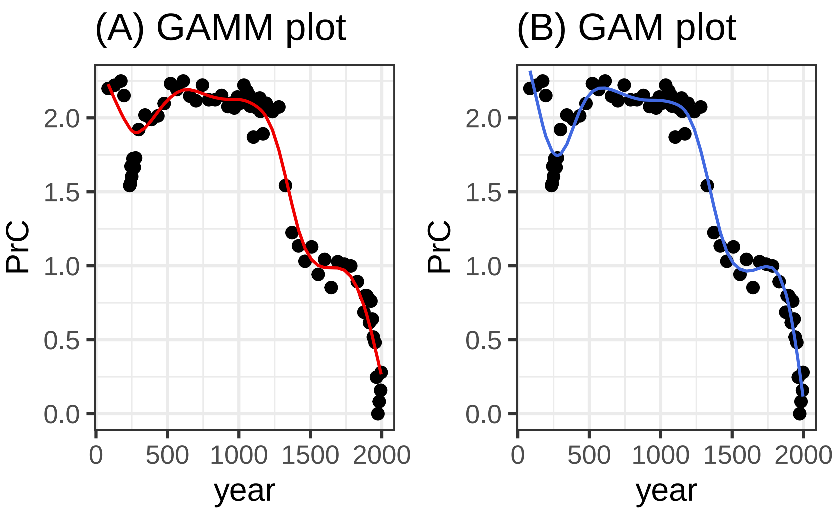

Figure 17: PrC (a) GAMM and (b) GAM with adaptive spline - raw scores and fitted relationships.

```r
derivsPC <- fderiv(mod, newdata = pcData)
derivsContPC <- data.frame(pcData,
                           confint(derivsPC,
                                   type = "simultaneous"))


sizesPC <- signifD(derivsContPC$YEAR,
                 d = derivsContPC$est,
        upper = derivsContPC$upper,
        lower = derivsContPC$lower)


predSigsPC <- data.frame(
  YEAR = predPC$YEAR,
  predicted = predPC$predicted,
  increasing = sizesPC[[1]],
  decreasing = sizesPC[[2]]
  )

ggplot(predPC, aes(x= YEAR, y = predicted)) +
  geom_vline(xintercept = 232,
             linetype = "dotdash",
             col = "darkorange") +
    geom_vline(xintercept = 1280,
             linetype = "dashed",
             col = "grey50") +
    geom_vline(xintercept = 1840,
             linetype = "dashed",
             col = "grey50") +
  geom_point(data = prcurveScores, aes(y = PrC),
             shape = 1) +
  geom_path() +
  geom_path(data= predSigsPC, aes(x = decreasing),
            size = 2, colour = "red2") +
  geom_path(data= predSigsPC, aes(x = increasing),
            size = 2, colour = "red2") +
  theme_bw() +
  theme(panel.grid = element_blank())
```

You'll note that the Taupō volcanic eruption eruption of 232 AD (orange dashed lines) is reflected to an extent by the fitted gam relationship; while the change following human settlement is prolonged and substantial and reflected in the rapid rates of change found following the model (Fig. 18). A gam and a thin plate spline was able to derive a better fit to the data (code not shown).

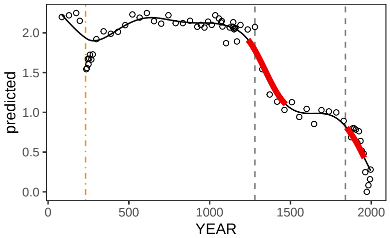

Figure 18: Periods of rapid change modelled using GAMM and PrC scores

6 Conclusions

This supplement has set out the methods by which one can calculate distance from baseline and change over time, and testing (a) whether a priori defined periods differ in their distance from baseline and (b) whether (and where) there are rapid periods of change.

References

Burge, Olivia Rata. (2022). Baselines: A package to calculate baselines from multivariate data. doi:10.5281/zenodo.6420020

Burge, Olivia R., Richardson, S. J., Wood, J. R., & Wilmshurst, J. M. (2023). A guide to assess distance from ecological baselines and change over time in palaeoecological records. *The Holocene*, *33*(8), 905–917. doi:10.1177/09596836231169986

Hui, F. K. C. (2016). Boral – bayesian ordination and regression analysis of multivariate abundance data in R. *Methods in Ecology and Evolution*, *7*(6), 744–750. doi:10.1111/2041-210X.12514

Magurran, A. E., Dornelas, M., Moyes, F., Gotelli, N. J., & McGill, B. (2015). Rapid biotic homogenization of marine fish assemblages. *Nature Communications*, *6*, ncomms9405. doi:10.1038/ncomms9405

Oksanen, J., Blanchet, F. G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P. R., O'Hara, R. B., … Wagner, H. (2019). Vegan: Community ecology package. R package version 2.5-6.

Simpson, G. L. (2020). Gratia: Graceful 'ggplot'-Based graphics and other functions for GAMs fitted using 'mgcv'. R package version 0.3.0.

Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis*. New York: Springer-Verlag.

Wilmshurst, J. M., & McGlone, M. S. (1996). Forest disturbance in the central North Island, New Zealand, following the 1850 BP Taupo eruption. *The Holocene*, *6*(4), 399–411. doi:10.1177/095968369600600402