Supplementary text: R code to reproduce the analyses presented in main text and supplementary material

The accompanying data set

The data file (“Vaughan_Gotelli_data.csv”) contains both raw data and the results for some parts of the analysis. It comprises:

- 27378 rows x 115 columns
- 3067 sampling locations
- 19915 ‘core’ samples used for classification of macroinvertebrate communities and subsequent Markov model analysis. Samples taken in consecutive years from the same locations, allowing annual transitions between biological classes to be calculated.
- 7463 ‘extra’ samples. Additional samples from the 3067 locations, excluded from the biological classification and Markov chain analyses as they were not sampled in neighbouring years, but used for the credit and debt calculations.

Guide to the columns:

- 1-5: basic metadata - site codes, year, location and whether the row is a core or extra sample.
- 6-83: invertebrate data (78 taxa). The values are mid-points of the abundance scales (1-9 individuals, 10-99, etc), $\log_{10}$ transformed.
- 84-87: water chemistry and temperature, and mean air temperatures.
- 88-89: river classification into upland v. lowland (‘up.2.cls’), and urban v. rural locations (urb.2.cls’).
- 90-93: additional geographic variables (altitude, slope, distance from source and proportion of catchment with urban land cover).
- 94-105: classifications from the 12 different cluster analyses (results from Section 1 below). Column names are explained following the Section 1 code.
- 106-110: reconstructed environmental conditions (water and air temperatures, BOD and PC1 of nitrate, orthophosphate and discharge).

Read in the data and load the requisite R packages

```r
# Read in the data set and R packages
df1 <- read.csv("Vaughan_Gotelli_data.csv")

library(vegan)
library(cluster)
library(mgcv)
library(reshape2)
library(nlme)
library(dplyr)
library(rioja)
```

The following code is divided into seven sections:

1. Cluster analysis of macroinvertebrate data
2. Analysis of temporal trends in the prevalence of the biological classes generated in Section 1
3. Markov model 1: loglinear analysis to select an appropriate model structure
4. Markov model 2: trends in annual transition probabilities
5. Markov model 3: calculate transition matrix statistics
6. Calibrate transfer functions and reconstruct environmental conditions
7. Calculate environmental credits, debts and lags

Sections 1-5 relate to Part 1 in the study workflow figure (Fig S3), whilst 6-7 relate to Part 2. For brevity, where a similar analysis is run repeatedly (e.g. the nine regressions fitted for Fig 2/Table S4), just one is reproduced here, but results for all permutations are provided in accompanying .csv file.

SECTION 1: Cluster analysis

```
# Prepare the invertebrate data: presence-absence and log abundance versions
# of the site x family matrix
abun <- df1
abun <- abun[abun$Data.set == "core", ]
prev <- abun
prev[, 6:83][prev[, 6:83] != 0] <- 1


# Calculate Jaccard and Bray-Curtis distances among invertebrate samples
Jac.d <- vegdist(prev[, 6:83], method = "jaccard", binary = TRUE)
BC.d <- vegdist(abun[, 6:83], method = "bray", binary = FALSE)


# Hierarchical cluster analysis using Ward's method
Jac.clus <- hclust(Jac.d, method = "ward.D2")
BC.clus <- hclust(BC.d, method = "ward.D2")


# Partitioning Around Medoids (PAM) clustering
Jac.pam.3 <- pam(Jac.d, diss = TRUE, k = 3)
BC.pam.3 <- pam(BC.d, diss = TRUE, k = 3)
```

The results of the cluster analyses for the 12 different approaches (Jaccard and Bray-Curtis distances, Ward's and PAM clustering, and 3, 5 and 7 clusters) are saved in columns 94-105 of 'Vaughan_Gotelli_data.csv'. The column names are built from the following abbreviations: 'hc' = hierarchical (Ward's) clustering; 'pam' = partitioning around medoids clustering; 'Jac' = classification based on presence-absence data (Jaccard dissimilarities); 'BC' = classification based on abundance data (Bray-Curtis dissimilarities); and the number at the end = number of biological classes.

SECTION 2: Temporal trends in biological class prevalence

```
# Read in the classifications from Section 1
cl.df <- df1[df1$Data.set == "core", -c(6:83, 106:110)]
```

```r
# Create a data set for making model predictions: all 3067 locations and
# every year (1991-2011)
site.data <- unique(cl.df[, c("Site", "Easting",  "Northing", "Altitude",
                              "Slope", "D.source", "LCM.urban")])
pr.data <- expand.grid(Site = unique(cl.df$Site), Year = seq(1991, 2011))
pr.data <- merge(pr.data, site.data)


# Create a separate presence-absence variable for each biological class.
# Illustrated here with 3-class, Jaccard distances, Ward's clustering.
hc.Jac.3.1 <- ifelse(cl.df$hc.Jac.3.clus == 1, 1, 0)
hc.Jac.3.2 <- ifelse(cl.df$hc.Jac.3.clus == 2, 1, 0)
hc.Jac.3.3 <- ifelse(cl.df$hc.Jac.3.clus == 3, 1, 0)


# Model the observed changes in class prevalence through time
hc.Jac.3.1.gam <- gam(hc.Jac.3.1 ~ as.factor(Year) + log(Altitude + 1) +
                        log(Slope + 1) + log(D.source + 1) +
                        log(LCM.urban + 1) + te(Easting, Northing),
                      data = cl.df, family = binomial)
hc.Jac.3.2.gam <- gam(hc.Jac.3.2 ~ as.factor(Year) + log(Altitude + 1) +
                        log(Slope + 1) + log(D.source + 1) +
                        log(LCM.urban + 1) + te(Easting, Northing),
                      data = cl.df, family = binomial)
hc.Jac.3.3.gam <- gam(hc.Jac.3.3 ~ as.factor(Year) + log(Altitude + 1) +
                        log(Slope + 1) + log(D.source + 1) +
                        log(LCM.urban + 1) +te(Easting, Northing),
                      data = cl.df, family = binomial)

# Generate predictions
hc.Jac.3.1.pr <- predict(hc.Jac.3.1.gam, pr.data, type = "response")
hc.Jac.3.2.pr <- predict(hc.Jac.3.2.gam, pr.data, type = "response")
hc.Jac.3.3.pr <- predict(hc.Jac.3.3.gam, pr.data, type = "response")


# Calculate annual mean prevalence
hc.Jac.3.1.pr <- tapply(hc.Jac.3.1.pr, pr.data$Year, mean)
hc.Jac.3.2.pr <- tapply(hc.Jac.3.2.pr, pr.data$Year, mean)
hc.Jac.3.3.pr <- tapply(hc.Jac.3.3.pr, pr.data$Year, mean)


# Compile the results
props <- data.frame(Year = seq(1991, 2011), Jac.3.Cl1 = hc.Jac.3.1.pr,
                    Jac.3.Cl2 = hc.Jac.3.2.pr, Jac.3.Cl3 = hc.Jac.3.3.pr)


# ------------------------------------
# Bootstrapped 95% confidence intervals
sites <- unique(cl.df$Site)

for (h in 1:50){

  boot.s <- data.frame(Site = sample(sites, 3067, replace = TRUE))
  boot.data <- merge(boot.s, cl.df)

  # Create data set for predictions, preserving duplicate site-years from
  # the bootstrap sampling
  for (k in 1991:2011) {ifelse(k == 1991, yrs <- rep(k, 3067),
                               yrs <- c(yrs, rep(k, 3067)))}
  pr.data <- data.frame(Site = rep(boot.s$Site, 21), Year = yrs)
  pr.data <- merge(pr.data, site.data)

  hc.Jac.3.1 <- ifelse(boot.data$hc.Jac.3.clus == 1, 1, 0)
  hc.Jac.3.2 <- ifelse(boot.data$hc.Jac.3.clus == 2, 1, 0)
  hc.Jac.3.3 <- ifelse(boot.data$hc.Jac.3.clus == 3, 1, 0)

  hc.Jac.3.1.gam <- gam(hc.Jac.3.1 ~ as.factor(Year) + log(Altitude + 1) +
                          log(Slope + 1) + log(D.source + 1) +
                          log(LCM.urban + 1) + te(Easting, Northing, k = 4),
                        data = boot.data, family = binomial)
  hc.Jac.3.1.pr <- predict(hc.Jac.3.1.gam, pr.data, type = "response")
  hc.Jac.3.1.pr <- tapply(hc.Jac.3.1.pr, pr.data$Year, mean)

  hc.Jac.3.2.gam <- gam(hc.Jac.3.2 ~ as.factor(Year) + log(Altitude + 1) +
                          log(Slope + 1) + log(D.source + 1) +
                          log(LCM.urban + 1) + te(Easting, Northing, k = 4),
                        data = boot.data, family = binomial)
  hc.Jac.3.2.pr <- predict(hc.Jac.3.2.gam, pr.data, type = "response")
  hc.Jac.3.2.pr <- tapply(hc.Jac.3.2.pr, pr.data$Year, mean)

  hc.Jac.3.3.gam <-gam(hc.Jac.3.3 ~ as.factor(Year) + log(Altitude + 1) +
                        log(Slope + 1) + log(D.source + 1) +
                        log(LCM.urban + 1) + te(Easting, Northing, k = 4),
                        data = boot.data, family = binomial)
  hc.Jac.3.3.pr <- predict(hc.Jac.3.3.gam, pr.data, type = "response")
  hc.Jac.3.3.pr <- tapply(hc.Jac.3.3.pr, pr.data$Year, mean)

  props <- data.frame(Year = seq(1991, 2011), Jac.3.Cl1 = hc.Jac.3.1.pr,
                      Jac.3.Cl2 = hc.Jac.3.2.pr, Jac.3.Cl3 = hc.Jac.3.3.pr)

  ifelse(h == 1, {bs.trends <- props}, {bs.trends <- rbind(bs.trends, props)})

  ## Progress count
  Sys.sleep(0.02); print(h); flush.console()
}
# ------------------------------------


# Calculate the non-parametric 95% confidence limits and combine with annual
# estimates.
low.cl <- aggregate(bs.trends[, 2:4], by = list(bs.trends$Year),
                      quantile, probs = .025)
colnames(low.cl) <- paste(colnames(low.cl), ".low", sep = "")
upp.cl <- aggregate(bs.trends[, 2:4], by = list(bs.trends$Year),
                      quantile, probs = .975)
colnames(upp.cl) <- paste(colnames(upp.cl), ".upp", sep = "")
summ.bst <- data.frame(props, low.cl[, -1], upp.cl[, -1])
```

SECTION 3: Markov model part 1: loglinear analysis

Two custom functions are created (‘loglin.prep’ and ‘log.lin.tests’) to prepare the data and run the analysis to test for heterogeneity in transition probabilities through time and/or among river types following Caswell (2001) and Hill *et al*. (2002).

```r
# Custom function to prepare data for loglinear analysis
# --------------------------------------
loglin.prep <- function(x, classif, class.scheme, n.class) {
  # x = annual data frame
  # classif = site classification variable e.g. 2-class upland-lowland
  # class.scheme = scheme used for classifying the biological samples
  # n.class = number of biological classes

  bb <- unique(x[, classif])
  for (k in 1:length(bb)) {
    focal.cat <- bb[k]
    cc <- x[x[, classif] == focal.cat, ]
    yr.class.1 <- reshape2::dcast(cc[, c("Site", "Year", class.scheme)],
                                  Site ~ Year, dplyr::first)

    # Check that the columns are in ascending order
    if(!identical(colnames(yr.class.1)[2:22],
                  as.character(seq(1991, 2011)))) {
      stop("Error - problem with column order")}

    # Update the names when using years
    colnames(yr.class.1)[2:22] <- paste("T.", colnames(yr.class.1)[2:22],
                                        sep = "")
    yr <- list()

    for (n in 1:20) {
      cross.class <- matrix(nrow = n.class, ncol = n.class)
      colnames(cross.class) <- seq(1, n.class)
      rownames(cross.class) <- seq(1, n.class)
      for (i in 1:n.class) {
        for (j in 1:n.class) {
          time.pair <- yr.class.1[, (n + 1):(n + 2)]
          select.data <- time.pair[(time.pair[, 1] == j &
                                      time.pair[, 2] == i), ]
          select.data <- na.omit(select.data)
          cross.class[i, j] <- nrow(select.data)
        }
      }
      if(nrow(na.omit(time.pair)) != sum(cross.class)) {stop(
        "Error - numbers do not add up")}
      yr[[n]] <- cross.class
      cross.class <- sweep(cross.class, 2, colSums(cross.class), "/")
      cross.class[is.nan(cross.class)] <- 0
      names(yr)[[n]] <- paste(colnames(time.pair[1]), ".",
                              colnames(time.pair[2]), sep = "")
    }

    for (i in 1:20) {
      melted <- reshape2::melt(yr[[i]])
      melted <- cbind.data.frame(melted, rep(names(yr)[[i]], nrow(melted)))
      colnames(melted) <- c("Fate", "State", "Count", "Timing")
      melted$Fate <- as.factor(melted$Fate)
      melted$State <- as.factor(melted$State)
      if(i == 1) {yr.long <- melted} else {yr.long <- rbind(yr.long, melted)}
    }
    yr.long$loc <- focal.cat

    ifelse(k == 1, {yr.out <- yr.long}, {yr.out <- rbind(yr.out, yr.long)})
  }
  return(yr.out)
}
# ---------------------------------------


# ---------------------------------------
log.lin.tests <- function(x) {
  # Stage 1: fit the set of models
  # Null hypothesis (FS, STL)
  FS.STL <- glm(Count ~ Fate + State + Timing + loc +
                Fate:State + State:Timing + State:loc + Timing:loc +
                State:Timing:loc, data = x, family = poisson)

  # Add time (terms FT & FST) to the null hypothesis
  FST.STL <- glm(Count ~ Fate + State + Timing + loc +
                 Fate:State + State:Timing + State:loc + Timing:loc +
                 Fate:Timing +  State:Timing:loc + Fate:State:Timing,
               data = x, family = poisson)

  # Add location (terms FL & FSL) to the null hypothesis
  FSL.STL <- glm(Count ~ Fate + State + Timing + loc +
                 Fate:State + State:Timing + State:loc + Timing:loc +
                 Fate:loc + State:Timing:loc + Fate:State:loc,
               data = x, family = poisson)

  # Add time & space (terms FT, FL, FST and FSL) to the null hypothesis
  FST.FSL.STL <- glm(Count ~ Fate + State + Timing + loc +
                     Fate:State + State:Timing + State:loc + Timing:loc +
                     Fate:Timing + Fate:loc +
                     State:Timing:loc + Fate:State:Timing + Fate:State:loc,
                   data = x, family = poisson)

  # Saturated model (FSTL)
  FSTL <- glm(Count ~ Fate * State * Timing * loc,
              data = x, family = poisson)


  # Stage 2: create the output tables
  GoF.tab <- data.frame(Model = c("FS.STL", "FST.STL", "FSL.STL",
                                  "FST.FSL.STL", "FSTL"), G.sq = NA,
                        GoF.df = NA, AIC = NA, delta.AIC = NA)

  # Goodness-of-fit statistics for the models (log-likelihood ratio versus
  # the saturated model)
  GoF.tab$G.sq[1] <- round(2 * (logLik(FSTL) - logLik(FS.STL)), 2)
  GoF.tab$G.sq[2] <- round(2 * (logLik(FSTL) - logLik(FST.STL)), 2)
  GoF.tab$G.sq[3] <- round(2 * (logLik(FSTL) - logLik(FSL.STL)), 2)
  GoF.tab$G.sq[4] <- round(2 * (logLik(FSTL) - logLik(FST.FSL.STL)), 2)
  GoF.tab$G.sq[5] <- round(2 * (logLik(FSTL) - logLik(FSTL)), 2)
  GoF.tab$GoF.df[1] <- df.residual(FS.STL) - df.residual(FSTL)
  GoF.tab$GoF.df[2] <- df.residual(FST.STL) - df.residual(FSTL)
  GoF.tab$GoF.df[3] <- df.residual(FSL.STL) - df.residual(FSTL)
  GoF.tab$GoF.df[4] <- df.residual(FST.FSL.STL) - df.residual(FSTL)
  GoF.tab$GoF.df[5] <- df.residual(FSTL) - df.residual(FSTL)

  GoF.tab$AIC[1] <- round(AIC(FS.STL) - AIC(FSTL), 2)
  GoF.tab$AIC[2] <- round(AIC(FST.STL) - AIC(FSTL), 2)
  GoF.tab$AIC[3] <- round(AIC(FSL.STL) - AIC(FSTL), 2)
  GoF.tab$AIC[4] <- round(AIC(FST.FSL.STL) - AIC(FSTL), 2)
  GoF.tab$AIC[5] <- round(AIC(FSTL) - AIC(FSTL), 2)

  GoF.tab$delta.AIC[1] <- GoF.tab$AIC[1] - min(GoF.tab$AIC)
  GoF.tab$delta.AIC[2] <- GoF.tab$AIC[2] - min(GoF.tab$AIC)
  GoF.tab$delta.AIC[3] <- GoF.tab$AIC[3] - min(GoF.tab$AIC)
  GoF.tab$delta.AIC[4] <- GoF.tab$AIC[4] - min(GoF.tab$AIC)
  GoF.tab$delta.AIC[5] <- GoF.tab$AIC[5] - min(GoF.tab$AIC)

  t.res <- data.frame(Test = c("Time.1", "Time.2", "Space.1", "Space.2",
                               "Time.x.Space"),
                      Term = c("FT.FST", "FT.FST", "FL.FSL", "FL.FSL",
                               "FTL.FTSL"),
                      Model.1 = c("FS.STL", "FSL.STL", "FS.STL",
                                  "FST.STL", "FST.FSL.STL"),
                      Model.2 = c("FST.STL", "FST.FSL.STL", "FSL.STL",
                                  "FST.FSL.STL", "FSTL"), G.sq = NA,
                      delta.df = NA, p.val = NA)

  t.res$G.sq[1] <- round(2 * (logLik(FST.STL) - logLik(FS.STL)), 2)
  t.res$G.sq[2] <- round(2 * (logLik(FST.FSL.STL) - logLik(FSL.STL)), 2)
  t.res$G.sq[3] <- round(2 * (logLik(FSL.STL) - logLik(FS.STL)), 2)
  t.res$G.sq[4] <- round(2 * (logLik(FST.FSL.STL) - logLik(FST.STL)), 2)
  t.res$G.sq[5] <- round(2 * (logLik(FSTL) - logLik(FST.FSL.STL)), 2)

  # More complex model placed second so that the difference > 0
  t.res$delta.df[1] <- round(df.residual(FS.STL) - df.residual(FST.STL), 2)
  t.res$delta.df[2] <- round(df.residual(FSL.STL) -
                               df.residual(FST.FSL.STL), 2)
  t.res$delta.df[3] <- round(df.residual(FS.STL) - df.residual(FSL.STL), 2)
  t.res$delta.df[4] <- round(df.residual(FST.STL) -
                               df.residual(FST.FSL.STL), 2)
  t.res$delta.df[5] <- round(df.residual(FST.FSL.STL) - df.residual(FSTL), 2)

  t.res$p.val[1] <- 1 - pchisq((2 * (logLik(FST.STL) - logLik(FS.STL))),
                               t.res$delta.df[1])
  t.res$p.val[2] <- 1 - pchisq((2 * (logLik(FST.FSL.STL) - logLik(FSL.STL))),
                               t.res$delta.df[2])
  t.res$p.val[3] <- 1 - pchisq((2 * (logLik(FSL.STL) - logLik(FS.STL))),
                               t.res$delta.df[3])
  t.res$p.val[4] <- 1 - pchisq((2 * (logLik(FST.FSL.STL) - logLik(FST.STL))),
                               t.res$delta.df[4])
  t.res$p.val[5] <- 1 - pchisq((2 * (logLik(FSTL) - logLik(FST.FSL.STL))),
                               t.res$delta.df[5])

  res.tables <- list(GoF = GoF.tab, Tests = t.res)
  return(res.tables)
}
# -----------------------------------------
# Run the loglinear analyses.
# 1. Rivers split into upland and lowland
log.lin.tests(loglin.prep(cl.df, "up.2.cls", "hc.Jac.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "up.2.cls", "hc.BC.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "up.2.cls", "pam.Jac.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "up.2.cls", "pam.BC.3.clus", 3))

# 2. Rivers split into urban and rural
log.lin.tests(loglin.prep(cl.df, "urb.2.cls", "hc.Jac.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "urb.2.cls", "hc.BC.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "urb.2.cls", "pam.Jac.3.clus", 3))
log.lin.tests(loglin.prep(cl.df, "urb.2.cls", "pam.BC.3.clus", 3))
```

SECTION 4: Markov part 2: trends in annual transition probabilities

Create a custom function (‘gen.tmat’) to create matrices of annual transition probabilities and then use linear regressions, fitted by generalized least squares, to test for trends in transition probabilities through time. This relates to Fig 2 and Table S4.

```r
#  Create function 'gen.tmat'
# --------------------------------------
gen.tmat <- function(sites, Time, clus, n.class) {
  #  sites = vector of sampling sites from which data were collected
  #  Time = sampling year
  #  clus = clustering results (which class each sample belongs to)
  #  n.class = number of classes into which to divide the data (3)
  class.1 <- cbind.data.frame(sites, Time, clus)
  if (is.numeric(Time)) {
    class.1 <- reshape2::dcast(class.1, sites ~ Time, dplyr::first)
    class.1a <- class.1[, 2:ncol(class.1)]
    class.1a <- class.1a[, order(names(class.1a))]
    class.1 <- cbind.data.frame(class.1[, 1], class.1a)
    colnames(class.1)[1] <- "sites"
    colnames(class.1)[2:ncol(class.1)] <- paste("T.",
                                                colnames(class.1)[2:ncol(class.1)]
, sep = "")
  } else {
    class.1 <- reshape2::dcast(class.1, sites ~ Time, first)
  }
  count.list <- list()
  props.list <- list()
  for (n in 1:(ncol(class.1) - 2)) {
    cross.class <- matrix(nrow = n.class, ncol = n.class)
    colnames(cross.class) <- seq(1, n.class)
    rownames(cross.class) <- seq(1, n.class)
    for (i in 1:n.class) {
      for (j in 1:n.class) {
        time.pair <- class.1[, (n + 1):(n + 2)]
        select.data <- time.pair[(time.pair[, 1] == j &
                                    time.pair[, 2] == i), ]
        select.data <- na.omit(select.data)
        cross.class[i, j] <- nrow(select.data)
      }
    }
    if(nrow(na.omit(time.pair)) != sum(cross.class)) {stop(
      "Error - numbers do not add up")}
    count.list[[n]] <- cross.class
    cross.class <- sweep(cross.class, 2, colSums(cross.class), "/")
    cross.class[is.nan(cross.class)] <- 0
    props.list[[n]] <- cross.class
    names(count.list)[[n]] <- paste(colnames(time.pair[1]), ".",
                                    colnames(time.pair[2]), sep = "")
    names(props.list)[[n]] <- paste(colnames(time.pair[1]), ".",
                                    colnames(time.pair[2]), sep = "")
  }

  mat.list <- list(counts = count.list, props = props.list)
  mat.list

}
# --------------------------------------


# Generate data frames of annual transition probabilities.
tr.p <- data.frame(State = c(rep("Cl.1", 3), rep("Cl.2", 3), rep("Cl.3", 3)),
                   Fate  = rep(c("Cl.1", "Cl.2", "Cl.3"), 3),
                   Trans = NA)
hc.Jac3.yr <- gen.tmat(cl.df$Site, cl.df$Year,  cl.df$hc.Jac.3.clus, 3)$props
hc.BC3.yr <- gen.tmat(cl.df$Site, cl.df$Year, cl.df$hc.BC.3.clus, 3)$props
pam.Jac3.yr <- gen.tmat(cl.df$Site, cl.df$Year, cl.df$pam.Jac.3.clus, 3)$props
pam.BC3.yr <- gen.tmat(cl.df$Site, cl.df$Year, cl.df$pam.BC.3.clus, 3)$props

for (i in 1:20) {
  aa <- cbind(tr.p, as.vector(hc.Jac3.yr[[i]]),
              as.vector(hc.BC3.yr[[i]]), as.vector(pam.Jac3.yr[[i]]),
              as.vector(pam.BC3.yr[[i]]))
  aa$Trans <- i
  ifelse(i == 1, tr.probs <- aa, tr.probs <- rbind(tr.probs, aa))
}
colnames(tr.probs)[4:7] <- c("hc.Jac", "hc.BC", "pam.Jac", "pam.BC")
tr.probs <- tr.probs[order(tr.probs$State, tr.probs$Fate), ]


# Fit gls models: with and without first order autocorrelation functions.
# Example here = persisting in Class 1 (i.e. State = 1, Fate = 1)
m1.1 <- gls(hc.Jac ~ Trans, method = "REML",
             data = tr.probs[tr.probs$State == "Cl.1" &
                               tr.probs$Fate == "Cl.1", ])
m1.2 <- update(m1.1, correlation = corARMA(p = 1, q = 0))
m1.3 <- update(m1.1, correlation = corARMA(p = 0, q = 1))
m1.4 <- update(m1.1, correlation = corARMA(p = 1, q = 1))


# Select optimal correlation structure by comparing AIC
AIC(m1.1, m1.2, m1.3, m1.4)
```

SECTION 5: Markov part 3: calculate transition matrix statistics

```r
# Create function 'tr.mat'
# --------------------------------------
tr.mat <- function(p.list) {
  # p.list = 'gen.tmat' object (list of matrices using proportions)
  require(popbio)
  require(Matrix)

  for (i in 1:length(p.list$props)) {
    trans.p <- p.list$props[[i]]
    trans.c <- p.list$counts[[i]]

    # Observed prevalence of each class (second time point of the transition)
    obs.pr <-  data.frame(t(rowSums(trans.c) / sum(trans.c)))
    colnames(obs.pr) <- paste("Obs.", seq(1, nrow(trans.c)), sep = "")

    # Persistance
    persist <- sum(diag(trans.c)) / sum(trans.c)

    # Predicted proportions of the different classes at equilibrium
    eqm.pr <- data.frame(t(popbio::eigen.analysis(trans.p)$stable.stage))
    colnames(eqm.pr) <- paste("Eqm.", seq(1, nrow(trans.p)), sep = "")

    # Mean absolute difference between obs and eqm class prevalences
    oe.diff <- rowMeans(abs(obs.pr - eqm.pr))

    # Damping ratio
    damp.ratio <- damping.ratio(trans.p)
    damp.conv.rate <- log(damp.ratio)

    # Dobrushin coefficient
    dob.1 <- as.matrix(rep(NA, nrow(trans.p)))
    for (k in 1:nrow(trans.p)) {
      # Calculates the 1-norm = maximum absolute column value
      dob.1[k] <- norm(trans.p - trans.p[, k], type = "1")
    }
    dob.1 <- .5 * max(dob.1)
    dob.conv.rate <- -log(dob.1)

    df1 <- data.frame(Matrix = names(p.list$props)[i], obs.pr, eqm.pr,
                      OE.diff = oe.diff, Persist = persist,
                      Damp.r = damp.ratio, Damp.CR = damp.conv.rate,
                      Dobrushin = dob.1, Dobrushin.CR = dob.conv.rate)

    ifelse(i == 1, MC.res <- df1, MC.res <- rbind(MC.res, df1))
  }
  MC.res
}
# --------------------------------------


# Typical call to generate transition matrix statistics for the 3-class
# (Jaccard distances/Ward's method) results
HCJ3 <- tr.mat(gen.tmat(cl.df$Site, cl.df$Year, cl.df$hc.Jac.3.clus, 3))


# GLS analysis of transition matrix properties through time follows the same
# protocol as for transition probabilities (Section 2). The example below is for t
he
# mean difference between the observed and predicted (equilibrium) class
# frequencies. Annual transitions are labelled 1-20, with 0.5 added to
# reflect that each transition spans two years.
m.1 <- gls(OE.diff ~ I(seq(1, 20) + .5), method = "REML", data = HCJ3)
m.2 <- update(m.1, correlation = corARMA(p = 1, q = 0))
```

```r
m.3 <- update(m.1, correlation = corARMA(p = 0, q = 1))
m.4 <- update(m.1, correlation = corARMA(p = 1, q = 1))
AIC(m.1, m.2, m.3, m.4)
```

SECTION 6: Calibrate transfer functions and reconstruct environmental conditions

```r
# Extract training data (1991-2) and data for reconstructing temperature
# or water quality 1991-2011 (= 'recon.d').
# Rows removed where water chemistry is missing.
train.d <- df1[df1$Year == 1991 | df1$Year == 1992, 1:87]
train.d <- na.omit(train.d)
recon.d <- na.omit(df1[, 6:87])


# Fit WA-PLS models. Variable 'Nut.disch.PC1' = PC1 from nitrate,
# orthophosphate and discharge.
BOD.m <- WAPLS(train.d[, 6:83], train.d$log.BOD.25km)
Temp.m <- WAPLS(train.d[, 6:83], train.d$Water.T.25km)
Air.T.m <- WAPLS(train.d[, 6:83], train.d$Mean.air.temp)
NutD.m <- WAPLS(train.d[, 6:83], train.d$Nut.disch.PC1)


# Cross validate the models
BOD.m.cv <- crossval(BOD.m, cv.method = "loo")
Temp.m.cv <- crossval(Temp.m, cv.method = "loo")
Air.T.m.cv <- crossval(Air.T.m, cv.method = "loo")
NutD.m.cv <- crossval(NutD.m, cv.method = "loo")


# Select number of components to retain
rand.t.test(BOD.m.cv)
rand.t.test(Temp.m.cv)
rand.t.test(Air.T.m.cv)
rand.t.test(NutD.m.cv)


# Generate reconstructed conditions (= predictions using 'recon.d'). Illustrated h
ere for BOD.
BOD.1 <- data.frame(predict(BOD.m, recon.d[1:15000, 1:78]))
BOD.2 <- data.frame(predict(BOD.m, recon.d[15001:26103,]))
BOD <- rbind(BOD.1, BOD.2)


# Reconstructed temperature and water chemistry variables are saved in
# columns 107-110. Column 106 ('Env.recon') identifies the 26103 samples
# are used to compare observed and reconstructed conditions (i.e. the 27378
# samples, less those for which observed chemistry was not available.
```

SECTION 7: Calculate environmental credits, debts and lags

```r
# Annual mean observed and re-constructed conditions
a.m <- df1[!is.na(df1$Env.recon), c(4, 84:87, 107:110)]
a.m <- aggregate(a.m, by = list(a.m$Year), mean)
colnames(a.m)[1] <- "Year"
```

```r
# Data frame for calculating credits/debts. Each column = reconstructed
# followed by observed conditions. Add in 'Yr.0': time series
# starting at zero for use with fixed y-intercepts
gls.data <- data.frame(Year = rep(seq(1991, 2011), 2),
                       Type = c(rep("Recon", 21), rep("Obs", 21)),
                       BOD = c(a.m$recon.BOD, a.m$log.BOD.25km),
                       Water.T = c(a.m$recon.Water.T, a.m$Water.T.25km),
                       Air.T = c(a.m$recon.Air.T, a.m$Mean.air.temp),
                       NutD = c(a.m$recon.NutD.PC1, a.m$Nut.disch.PC1))
gls.data$Yr.0 <- gls.data$Year - 1991


# Cross calibrate BOD (and nitrate/orthophopshate/discharge PC1) against
#  water and air temperatures.
cc.gam <- gam(recon.Water.T ~ s(recon.BOD), data = df1)
cc.AT.gam <- gam(recon.Air.T ~ s(recon.BOD), data = df1)
cc.NutD.gam <- gam(recon.Water.T ~ s(recon.NutD.PC1), data = df1)


# Estimate temperature equivalents for BOD and PC1
aa <- data.frame(recon.BOD = gls.data$BOD)
gls.data$BOD.T.eqv <- predict(cc.gam, newdata = aa, type = "response")
gls.data$BOD.Air.T.eqv <- predict(cc.AT.gam, newdata = aa, type = "response")

aa <- data.frame(recon.NutD.PC1 = gls.data$NutD)
gls.data$NutD.T.eqv <- predict(cc.NutD.gam, newdata = aa, type = "response")
rm(aa)


# Calculate the environmental lags
lags <- data.frame(Year = seq(1991, 2011), Yr.0 = seq(0, 20))
lags$Water.T.raw <- gls.data[gls.data$Type == "Obs", "Water.T"] -
  gls.data[gls.data$Type == "Recon", "Water.T"]
lags$BOD.raw <- gls.data[gls.data$Type == "Obs", "BOD"] -
  gls.data[gls.data$Type == "Recon", "BOD"]
lags$Air.T.raw <- gls.data[gls.data$Type == "Obs", "Air.T"] -
  gls.data[gls.data$Type == "Recon", "Air.T"]
lags$t.eqv.BOD <- gls.data[gls.data$Type == "Obs", "BOD.T.eqv"] -
  gls.data[gls.data$Type == "Recon", "BOD.T.eqv"]
lags$Air.T.eqv.BOD <- gls.data[gls.data$Type == "Obs", "BOD.Air.T.eqv"] -
  gls.data[gls.data$Type == "Recon", "BOD.Air.T.eqv"]
lags$NutD.raw <- gls.data[gls.data$Type == "Obs", "NutD"] -
  gls.data[gls.data$Type == "Recon", "NutD"]
lags$t.eqv.NutD <- gls.data[gls.data$Type == "Obs", "NutD.T.eqv"] -
  gls.data[gls.data$Type == "Recon", "NutD.T.eqv"]


# Calculate the net lags
lags$Net.Water.T <- lags$Water.T.raw + lags$t.eqv.BOD
lags$Net.Air.T <- lags$Air.T.raw + lags$Air.T.eqv.BOD
lags$Net.NutD <- lags$Water.T.raw + lags$t.eqv.NutD


# Environmental credit and debt calculations. Example shown here:
#  = water temperature and BOD
# Credit/debt models using GLS. Compare first order models for autocorrelated
# residuals and a variance
# 1. Comparison of observed and reconstructed temperatures
m1 <- gls(Water.T ~ Yr.0 + Yr.0:Type, data = gls.data, method = "REML")
m2 <- update(m1, correlation = corARMA(p = 1, q = 0))
m3 <- update(m1, correlation = corARMA(p = 0, q = 1))
m4 <- update(m1, weights = varIdent(form = ~ 1 | Type))
m5 <- update(m1, correlation = corARMA(p = 1, q = 0),
             weights = varIdent(form = ~ 1 | Type))
m6 <- update(m1, correlation = corARMA(p = 0, q = 1),
             weights = varIdent(form = ~ 1 | Type))
AIC(m1, m2, m3, m4, m5, m6)


# 2. Comparison of observed and reconstructed BOD
m1 <- gls(BOD ~ Yr.0 + Yr.0:Type, data = gls.data, method = "REML")
m2 <- update(m1, correlation = corARMA(p = 1, q = 0))
m3 <- update(m1, correlation = corARMA(p = 0, q = 1))
m4 <- update(m1, weights = varIdent(form = ~ 1 | Type))
m5 <- update(m1, correlation = corARMA(p = 1, q = 0),
              weights = varIdent(form = ~ 1 | Type))
m6 <- update(m1, correlation = corARMA(p = 0, q = 1),
              weights = varIdent(form = ~ 1 | Type))
AIC(m1, m2, m3, m4, m5, m6)


# 3. Climatic (water temperature) lag v. Year
m1 <- gls(Water.T.raw ~ Yr.0 - 1, data = lags, method = "REML")
m2 <- update(m1, correlation = corARMA(p = 1, q = 0))
m3 <- update(m1, correlation = corARMA(p = 0, q = 1))
AIC(m1, m2, m3)


# 4. Water quality lag (in degrees Celsius equivalent) v. Year
m1 <- gls(t.eqv.BOD ~ Yr.0 - 1, data = lags, method = "REML")
m2 <- update(m1, correlation = corARMA(p = 1, q = 0))
m3 <- update(m2, correlation = corARMA(p = 0, q = 1))
AIC(m1, m2, m3)


# 5. Net environmental lag
m1 <- gls(Net.Water.T ~ Yr.0 - 1, data = lags, method = "REML")
m2 <- update(m1, correlation = corARMA(p = 1, q = 0))
m3 <- update(m1, correlation = corARMA(p = 0, q = 1))
AIC(m1, m2, m3)
```