**R CODE\_PeerJ The importance of local natural variability (Cardenas et al.,)**

**Within-year Generalized Additive model (GAM) for PY1**

require(nlme)

require(mgcv)

require(TSA)

require(lubridate)

***##Data***

PY1\_Jday <-read.csv(file="EXPORT YELCHO/ peerj-19157-PY1\_Jday.csv",stringsAsFactors = FALSE)

class(PY1\_Jday)

names(PY1\_Jday)

str(PY1\_Jday)

PY1\_Jday

***###Decimal day date transformation***

PY1\_Jday $Datetime <- as.POSIXct(PY1\_Jday $Datetime,format = "%m/%d/%Y %H:%M")

PY1\_Jday <- transform(PY1\_Jday, decitime = decimal\_date(Datetime))

str(PY1\_Jday)

plot(PY1\_Jday $Datetime, PY1\_Jday $mean\_TEMP, pch=16)

***###:::YP1***

YP1gam1<-gam(mean\_TEMP ~ s(decitime),data = PY1\_Jday)

summary(YP1gam1)

gam.check(YP1gam1)

par(mfrow=c(1,1))

plot(YP1gam1, scheme=1)

plot(YP1gam1,all.terms = T, shift = -0.33240 , ylab= "Mean Day Water Tempearture (C)", xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1 YELCHOP1 /n seasonality within a year", scheme = 1, pages=1)

points(PY1\_Jday$decitime, PY1\_Jday$mean\_TEMP, pch=16, cex=0.6)

***###\_\_\_GAM\_\_\_\_\_\_GAM\_checks for autocorrelation***

par(mfrow=c(1,2))

acf(residuals(YP1gam1))

pacf(residuals(YP1gam1))

par(mfrow=c(1,1))

***###\_\_\_GAM\_\_\_\_\_\_helps to find ARMA structure***

eacf(residuals(YP1gam1))

***##::::GAMM::::::::::Autoregression structure GAM1 final***

YP1\_AR1MA1 <- (gamm(mean\_TEMP ~ s(decitime), data = PY1\_Jday, correlation = corARMA(p=1, q = 1)))

anova(YP1gamm1$lme,YP1\_AR1$lme,YP1\_AR1MA1$lme,YP1\_AR1MA2$lme)#to choose

plot(YP1\_AR1MA1$gam,all.terms = T, shift = -0.31245 , ylab= "Mean Day Water Tempearture (C)", xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1AR1MA2 YELCHOP1 /n seasonality within a year", scheme = 1, pages=1)

points(PY1\_Jday$decitime, PY1\_Jday$mean\_TEMP, pch=16, cex=0.6)

***###FOR GAMM***

summary(YP1\_AR1MA1$gam)

layout(matrix(1:1, ncol = 1))

plot(YP1\_AR1MA1$gam, scale = 0)

layout(1)

reeeei<-resid(YP1\_AR1MA1$lme, type = "normalized")

par(mfrow=c(1,2))

acf(reeeei, lag.max = 300, main = "ACF - ARMA errors/YP1\_AR1MA1")

pacf(reeeei, lag.max = 300, main = "pACF- ARMA errors/YP1\_AR1MA1")

eacf(reeeei)

acf(reeeei, plot=FALSE)$acf[2]

**PY2**

***###Data***

PY2\_Jday <-read.csv(file="EXPORT YELCHO/ peerj-19157-PY2\_Jday.csv",stringsAsFactors = FALSE)

class(PY2\_Jday)

names(PY2\_Jday)

str(PY2\_Jday)

PY2\_Jday

***###Decimal day date transformation***

PY2\_Jday$Datetime20m <- as.POSIXct(PY2\_Jday $Datetime20m,format = "%m/%d/%Y %H:%M")

PY2\_Jday <- transform(PY2\_Jday, decitimeP20 = decimal\_date(Datetime20m))

str(PY2\_Jday)

par(mfrow=c(1,1))

plot(PY2\_Jday$Datetime20m, PY2\_Jday$mean\_TEMP20m, pch=16)

***###:::YP2***

names (PY2\_Jday)

YP20gam1<-gam(mean\_TEMP20m ~ s(decitimeP20),data = PY2\_Jday)###final

***###\_\_\_GAM\_\_\_\_\_\_FOR GAM***

summary(YP20gam1)

***###\_\_\_GAM\_\_\_\_\_\_GAM\_check residuals***

gam.check(YP20gam1)

***###\_\_\_GAM\_\_\_\_\_\_GAM\_check effect plot***

par(mfrow=c(1,2))

plot(YP20gam1, scheme=1)

***###\_\_\_GAM\_\_\_\_\_\_GAM check response plots***

plot(YP20gam1,all.terms = T, shift = -0.51117 , ylab= "Mean Day Water Tempearture (C)",

xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1 YELCHO P20m /n variability pattern within the period", scheme = 1, pages=1)

points(PY2\_Jday $decitimeP20, PY2\_Jday $mean\_TEMP20m, pch=16, cex=0.6)

***###\_\_\_GAM\_\_\_\_\_\_GAM\_check for autocorrelation***

par(mfrow=c(1,2))

acf(residuals(YP20gam1))

pacf(residuals(YP20gam1))

***###\_\_\_GAM\_\_\_\_\_\_helps to find ARMA structure***

eacf(residuals(YP20gam1))

***###::::GAMM::::::::::Autoregression structure GAM1***

YP20\_AR1 <- (gamm(mean\_TEMP20m ~ s(decitimeP20), data = PY2\_Jday, correlation = corARMA(p=1)))

##anova(YP20gamm1$lme,YP20\_AR1$lme,YP20\_AR1MA1$lme, YP20\_AR1MA2$lme)

names(PY2\_Jday)

par(mfrow=c(1,1))

plot(YP20\_AR1$gam,all.terms = T, shift = -0.50662 , ylab= "Mean Day Water Tempearture (C)", xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1AR1 YELCHOP4 /n seasonality within a year", scheme = 1, pages=1)

points(PY2\_Jday $decitimeP20, PY2\_Jday $mean\_TEMP20m, pch=16, cex=0.6)

***###FOR GAMM***

summary(YP20\_AR1$gam)

layout(matrix(1:1, ncol = 1))

plot(YP20\_AR1$gam, scale = 0)

layout(1)

RREESS<-resid(YP20\_AR1$lme, type = "normalized")

par(mfrow=c(1,2))

acf(RREESS, lag.max = 300, main = "ACF - ARMA errors/YP20\_AR1")

pacf(RREESS, lag.max = 300, main = "pACF- ARMA errors/YP20\_AR1")

eacf(RREESS)

acf(RREESS, plot=FALSE)$acf[2]

**PY4**

***###Data***

PY4\_Jday <-read.csv(file="EXPORT YELCHO/ peerj-19157-PY4\_Jday.csv",stringsAsFactors = FALSE)

class(PY4\_Jday)

names(PY4\_Jday)

str(PY4\_Jday)

PY4\_Jday

***###Decimal day date transformation***

PY4\_Jday $Datetimep4 <- as.POSIXct(PY4\_Jday $Datetimep4,format = "%m/%d/%Y %H:%M")

PY4\_Jday <- transform(PY4\_Jday, decitimeP4 = decimal\_date(Datetimep4))

str(PY4\_Jday)

par(mfrow=c(1,1))

plot(PY4\_Jday $Datetimep4, PY4\_Jday $mean\_TEMp4, pch=16)

***###:::YP4\_final***

names (PY4\_Jday)

YP4gam1<-gam(mean\_TEMp4 ~ s(decitimeP4),data = PY4\_Jday)### final

***###\_\_\_GAM\_\_\_\_\_\_FOR GAM***

summary(YP4gam1)

***#\_\_\_GAM\_\_\_\_\_\_GAM\_check residuals***

gam.check(YP4gam1)

***#\_\_\_GAM\_\_\_\_\_\_GAM\_check effect plot***

par(mfrow=c(1,2))

plot(YP4gam2, scheme=1)

par(mfrow=c(1,1))

plot(YP4gam2, scheme=1)

***###\_\_\_GAM\_\_\_\_\_\_GAM check response plots***

plot(YP4gam1, all.terms = T, shift = -0.598156 , ylab= "Mean Day Water Tempearture (C)",

xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1 YELCHO P4 /n variability pattern within the period", scheme = 1, pages=1)

points(PY4\_Jday $decitimeP4, PY4\_Jday $mean\_TEMp4, pch=16, cex=0.6)

***###\_\_\_GAM\_\_\_\_\_\_GAM\_check for autocorrelation***

par(mfrow=c(1,2))

acf(residuals(YP4gam1))

pacf(residuals(YP4gam1))

***###\_\_\_GAM\_\_\_\_\_\_helps to find ARMA structure***

eacf(residuals(YP4gam1))

***###::::GAMM::::::::::Autoregression structure GAM1***

YP4\_AR1 <- (gamm(mean\_TEMp4 ~ s(decitimeP4), data = PY4\_Jday, correlation = corARMA(p=1)))

##anova(YP4gamm1$lme,YP4\_AR1$lme,YP4\_AR1MA1$lme,YP4\_AR2MA1$lme)

plot(YP4\_AR1$gam,all.terms = T, shift = -0.60589, ylab= "Mean Day Water Tempearture (C)", xlab = "Time (days)", ylim=c(min(-1.5),max(3.2)),main="gam1AR1 YELCHOP4 /n seasonality within a year", scheme = 1, pages=1)

points(PY4\_Jday $decitimeP4, PY4\_Jday $mean\_TEMp4, pch=16, cex=0.6)

***###FOR GAMM\_results summary check autocorrelation***

summary(YP4\_AR1$gam)

layout(matrix(1:1, ncol = 1))

plot(YP1\_AR1MA1$gam, scale = 0)

layout(1)

RESS<-resid(YP4\_AR1$lme, type = "normalized")

par(mfrow=c(1,2))

acf(RESS, lag.max = 300, main = "ACF - ARMA errors/ YP4\_AR1")

pacf(RESS, lag.max = 300, main = "pACF- ARMA errors/ YP4\_AR1")

eacf(RESS)

acf(RESS, plot=FALSE)$acf[2]

**PY1\_ Method of finite differences (Curtis and Simpson, 2014: Monteith et al., 2014)**

require(nlme)

require(mgcv)

require(TSA)

require(lubridate)

YP1gam2<-gam(mean\_TEMP ~ s(decitime, k=58),data = PY1\_Jday)

***##################################################################***

***## Functions for derivatives of GAM(M) models from Curtis and Simpson, 2014##***

***##################################################################***

Deriv <- function(mod, n = 400, eps = 1e-3, newdata, term) {

if(inherits(mod, "gamm"))

mod <- mod$gam

m.terms <- attr(terms(mod), "term.labels")

if(missing(newdata)) {

newD <- sapply(model.frame(mod)[, m.terms, drop = FALSE],

function(x) seq(min(x), max(x), length = n))

names(newD) <- m.terms

} else {

newD <- newdata

}

newDF <- data.frame(newD) ## needs to be a data frame for predict

X0 <- predict(mod, newDF, type = "lpmatrix")

newDF <- newDF + eps

X1 <- predict(mod, newDF, type = "lpmatrix")

Xp <- (X1 - X0) / eps

Xp.r <- NROW(Xp)

Xp.c <- NCOL(Xp)

## dims of bs

bs.dims <- sapply(mod$smooth, "[[", "bs.dim") - 1

## number of smooth terms

t.labs <- attr(mod$terms, "term.labels")

## match the term with the the terms in the model

if(!missing(term)) {

want <- grep(term, t.labs)

if(!identical(length(want), length(term)))

stop("One or more 'term's not found in model!")

t.labs <- t.labs[want]

}

nt <- length(t.labs)

## list to hold the derivatives

lD <- vector(mode = "list", length = nt)

names(lD) <- t.labs

for(i in seq\_len(nt)) {

Xi <- Xp \* 0

want <- grep(t.labs[i], colnames(X1))

Xi[, want] <- Xp[, want]

df <- Xi %\*% coef(mod)

df.sd <- rowSums(Xi %\*% mod$Vp \* Xi)^.5

lD[[i]] <- list(deriv = df, se.deriv = df.sd)

}

class(lD) <- "Deriv"

lD$gamModel <- mod

lD$eps <- eps

lD$eval <- newD - eps

lD ##return

}

confint.Deriv <- function(object, term, alpha = 0.05, ...) {

l <- length(object) - 3

term.labs <- names(object[seq\_len(l)])

if(missing(term)) {

term <- term.labs

} else { ## how many attempts to get this right!?!?

##term <- match(term, term.labs)

##term <- term[match(term, term.labs)]

term <- term.labs[match(term, term.labs)]

}

if(any(miss <- is.na(term)))

stop(paste("'term'", term[miss], "not a valid model term."))

res <- vector(mode = "list", length = length(term))

names(res) <- term

residual.df <- df.residual(object$gamModel)

tVal <- qt(1 - (alpha/2), residual.df)

##for(i in term.labs[term]) {

for(i in term) {

upr <- object[[i]]$deriv + tVal \* object[[i]]$se.deriv

lwr <- object[[i]]$deriv - tVal \* object[[i]]$se.deriv

res[[i]] <- list(upper = drop(upr), lower = drop(lwr))

}

res$alpha = alpha

res

}

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)

}

plot.Deriv <- function(x, alpha = 0.05, polygon = TRUE,

sizer = FALSE, term,

eval = 0, lwd = 3,

col = "lightgrey", border = col,

ylab, xlab, main, ...) {

l <- length(x) - 3

## get terms and check specified (if any) are in model

term.labs <- names(x[seq\_len(l)])

if(missing(term)) {

term <- term.labs

} else {

term <- term.labs[match(term, term.labs)]

}

if(any(miss <- is.na(term)))

stop(paste("'term'", term[miss], "not a valid model term."))

if(all(miss))

stop("All terms in 'term' not found in model.")

l <- sum(!miss)

nplt <- n2mfrow(l)

tVal <- qt(1 - (alpha/2), df.residual(x$gamModel))

if(missing(ylab))

ylab <- expression(italic(hat(f)\*"'"\*(x)))

if(missing(xlab)) {

xlab <- attr(terms(x$gamModel), "term.labels")

names(xlab) <- xlab

}

if (missing(main)) {

main <- term

names(main) <- term

}

## compute confidence interval

CI <- confint(x, term = term)

## plots

layout(matrix(seq\_len(l), nrow = nplt[1], ncol = nplt[2]))

for(i in term) {

upr <- CI[[i]]$upper

lwr <- CI[[i]]$lower

ylim <- range(upr, lwr)

plot(x$eval[,i], x[[i]]$deriv, type = "n",

ylim = ylim, ylab = ylab, xlab = xlab[i], main = main[i], ...)

if(isTRUE(polygon)) {

polygon(c(x$eval[,i], rev(x$eval[,i])),

c(upr, rev(lwr)), col = col, border = border)

} else {

lines(x$eval[,i], upr, lty = "dashed")

lines(x$eval[,i], lwr, lty = "dashed")

}

abline(h = 0, ...)

if(isTRUE(sizer)) {

lines(x$eval[,i], x[[i]]$deriv, lwd = 1)

S <- signifD(x[[i]]$deriv, x[[i]]$deriv, upr, lwr,

eval = eval)

lines(x$eval[,i], S$incr, lwd = lwd, col = "blue")

lines(x$eval[,i], S$decr, lwd = lwd, col = "red")

} else {

lines(x$eval[,i], x[[i]]$deriv, lwd = 2)

}

}

layout(1)

invisible(x)

}

***###Derivative code***

YEL1.deriv1=Deriv(YP1gam2, n=404)

plot(YEL1.deriv1, sizer=TRUE, alpha=0.01)##first derivative plot

***###Plot***

plot(mean\_TEMP ~ decitime, data = PY1\_Jday, type = "p", ylab = "Water Temperature")

pdat <- with(PY1\_Jday, data.frame(decitime = seq(min(decitime), max(decitime), length = 404)))

p2 <- predict(YP1gam2, newdata = pdat)

lines(p2 ~ decitime, data = pdat)

CI <- confint(YEL1.deriv1, alpha = 0.01)

S <- signifD(p2, YEL1.deriv1$decitime$deriv, CI$decitime$upper, CI$decitime$lower,

eval = 0)

lines(S$incr ~ decitime, data = pdat, lwd = 3, col = "blue")

lines(S$decr ~ decitime, data = pdat, lwd = 3, col = "red")

***##Export data from method of finite differences to csv file***

blueSIGPLOT<-S <- signifD(p2, YEL1.deriv1$decitime$deriv, CI$decitime$upper, CI$decitime$lower, eval = 0)

write.csv( blueSIGPLOT, file = "blueSIGPLOT.csv")