eAppendix 1. R code to find SE and CI for the MMT of the temperature-mortality spline.

eAppendix 2. Temperature-mortality associations for the 52 provincial capital cities in Spain (with 95% CI shaded grey). Dashed vertical lines are unconstrained minimum mortality temperatures (MMT). Average daily mortality count is indicated in parentheses after the city name. RR=relative risk.

eAppendix 3. Sensitivity analysis for the cities where unconstrained minimum mortality temperatures (MMT) are at or close to one of the imprecisely estimated tails of the curve. RR=relative risk. I2=index of heterogeneity.

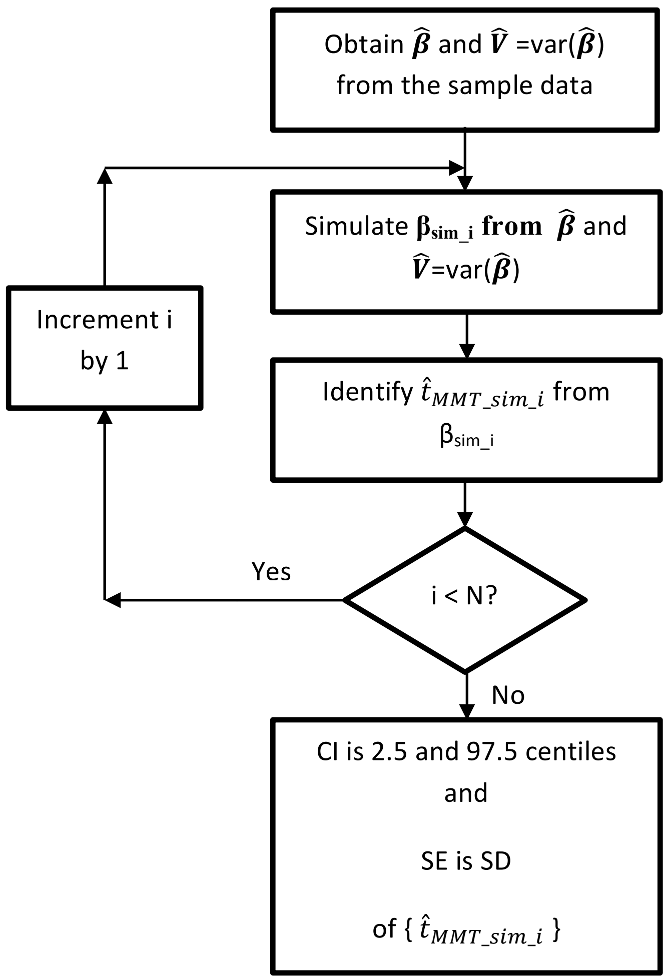
eAppendix 4. Sensitivity analysis for DLNM choice of knots and lag period. I2=index of heterogeneity. Res. I2=residual heterogeneity.

Algorithm to find SE and CI for the minimum mortality temperature of the temperature-mortality spline.

Suppose we have a parameter vector **β** of a function of **t** (typically a spline of temperature, but the algorithm proposed is more general):

$y = a + f(t, \boldsymbol{\theta})$ (1)

Denote the value of t at which y is minimum as t_{MMT}, and the minimum y as y_{MMT}. Using the ^ notation to denote a sample estimate of a parameter, we know $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{V}}$=var($\widehat{\boldsymbol{\beta}}$). We seek a CI and SE for $\hat{t}_{MMT}$.



CIs and SEs obtained from the above algorithm may be termed an approximate parametric bootstrap. ***Parametric*** because we generate resampled data from parameters rather than actually resampling data, and ***approximate*** because we generate resampled values of the estimated spline rather than of the data themselves.

```
##################################################################################
# FUNCTION TO ESTIMATE MINIMUM OF A EXPOSURE-RESPONSE FUNCTION FROM A FITTED MODEL
##################################################################################
#
# DISCLAIMER:
#   THE CODE COMPOSING THIS FUNCTION HAS NOT BEEN SYSTEMATICALLY TESTED. THE
#   PRESENCE OF BUGS CANNOT BE RULED OUT. ALSO, ALTHOUGH WRITTEN GENERICALLY
#   FOR WORKING IN DIFFERENT SCENARIOS AND DATA, THE FUNCTION HAS NOT BEEN
#   TESTED IN CONTEXTS DIFFERENT THAN THE EXAMPLE INCLUDED IN THE PAPER.
#   IT IS RESPONSIBILITY OF THE USER TO CHECK THE RELIABILITY OF THE RESULTS IN
#   DIFFERENT APPLICATIONS.
#
# UPDATE: 18 October 2016
# Future updates of this function can be found at http://www.ag-myresearch.com/
#
##################################################################################
#
findmin <- function(basis,model=NULL,coef=NULL,vcov=NULL,at=NULL,from=NULL,
  to=NULL,by=NULL,sim=FALSE,nsim=5000) {
#
################################################################################
#   ARGUMENTS:
#   - basis: A SPLINE OR OTHER BASIS FOR AN EXPOSURE x CREATED BY DLNM FUNCTION
#            CROSSBASIS OR ONEBASIS
#   - model: THE FITTED MODEL
#   - coef AND vcov: COEF AND VCOV FOR basis IF model IS NOT PROVIDED
#
#   - at: A NUMERIC VECTOR OF x VALUES OVER WHICH THE MINIMUM IS SOUGHT
#   OR
#   - from, to: RANGE OF x VALUES OVER WHICH THE MINIMUM IS SOUGHT.
#   - by: INCREMENT OF THE SEQUENCES x VALUES OVER WHICH THE MINIMUM IS SOUGHT
#
#   - sim: IF BOOTSTRAP SIMULATION SAMPLES SHOULD BE RETURNED
#   - nsim: NUMBER OF SIMULATION SAMPLES
################################################################################


################################################################################
# CREATE THE BASIS AND EXTRACT COEF-VCOV
#
  # CHECK AND DEFINE BASIS
  if(!any(class(basis)%in%c("crossbasis","onebasis")))
    stop("the first argument must be an object of class 'crossbasis' or 'onebasis'")
#
  # INFO
  one <- any(class(basis)%in%c("onebasis"))
  attr <- attributes(basis)
  range <- attr(basis,"range")
  if(is.null(by)) by <- 0.1
  lag <- if(one) c(0,0) else cb=attr(basis,"lag")
  if(is.null(model)&&(is.null(coef)||is.null(vcov)))
    stop("At least 'model' or 'coef'-'vcov' must be provided")
  name <- deparse(substitute(basis))
  cond <- if(one) paste(name,"[[:print:]]*b[0-9]{1,2}",sep="") else
    paste(name,"[[:print:]]*v[0-9]{1,2}\\.l[0-9]{1,2}",sep="")
#
  # SET COEF, VCOV CLASS AND LINK
  if(!is.null(model)) {
    model.class <- class(model)
    coef <- dlnm:::getcoef(model,model.class)
    ind <- grep(cond,names(coef))
    coef <- coef[ind]
    vcov <- dlnm:::getvcov(model,model.class)[ind,ind,drop=FALSE]
    model.link <- dlnm:::getlink(model,model.class)
  } else model.class <- NA
```

```
#
  # CHECK
  if(length(coef)!=ncol(basis) || length(coef)!=dim(vcov)[1] ||
      any(is.na(coef)) || any(is.na(vcov)))
    stop("model or coef/vcov not consistent with basis")
#
  # DEFINE at
  at <- dlnm:::mkat(at,from,to,by,range,lag,bylag=1)
  predvar <- if(is.matrix(at)) rownames(at) else at
  predlag <- dlnm:::seqlag(lag,by=1)
#
  # CREATE THE MATRIX OF TRANSFORMED CENTRED VARIABLES (DEPENDENT ON TYPE)
  type <- if(one) "one" else "cb"
  Xpred <- dlnm:::mkXpred(type,basis,at,predvar,predlag,cen=NULL)
  Xpredall <- 0
  for(i in seq(length(predlag))) {
    ind <- seq(length(predvar))+length(predvar)*(i-1)
    Xpredall <- Xpredall + Xpred[ind,,drop=FALSE]
  }
#
################################################################################
# FIND THE MINIMUM
#
  pred <- drop(Xpredall%*%coef)
  ind <- which.min(pred)
  min <- predvar[ind]
#
################################################################################
# SIMULATIONS
  #
  if(sim) {
    # SIMULATE COEFFICIENTS
    k <- length(coef)
    eigen <- eigen(vcov)
    X <- matrix(rnorm(length(coef)*nsim),nsim)
    coefsim <- coef + eigen$vectors %*% diag(sqrt(eigen$values),k) %*% t(X)
    # COMPUTE MINIMUM
    minsim <- apply(coefsim,2,function(coefi) {
      pred <- drop(Xpredall%*%coefi)
      ind <- which.min(pred)
      return(predvar[ind])
    })
  }
#
################################################################################
#
  res <- if(sim) minsim else min
#
  return(res)
}

############################ END OF FINDMIN FUNCTION ############################
```

```
########### EXAMPLES OF ESTIMATION OF THE MMT USING PUBLIC DOMAIN DATA ###########
#
# The required data file london.csv is in the public domain, and available from
# http://www.ag-myresearch.com/bmcmrm2014.html
#
# UPDATE: 18 October 2016
# Future updates of this function can be found at http://www.ag-myresearch.com/
#
##################################################################################

library(dlnm) ; library(splines)
source("findmin.R")

################################################################################
# FIRST EXAMPLE: SUBSET OF DATA, SIMPLE MODEL, UNIDIMENSIONAL EXPOSURE-RESPONSE

# LOAD A SUBSET OF THE DATA: LONDON JULY-DEC 2005
# (CONVENIENCE SAMPLE, LARGISH ASIMMETRIC CI)
data <- subset(read.csv("london.csv"), year==2005 & month>6)
head(data)

# A UNIDIMENSIONAL 4DF SPLINE
b1 <- onebasis(data$tmean,df=4)

# SIMPLE MODEL WITH NO CONTROL FOR CONFOUNDING
m1 <- glm(death~b1,family=quasipoisson,data)

# ESTIMATE MMT, WITH CI AND STANDARD ERROR
(min1 <- findmin(b1,m1))
(min1ci <- quantile(findmin(b1,m1,sim=T),c(2.5,97.5)/100))
(min1se <- sd(findmin(b1,m1,sim=T)))

# ESTIMATE THE MINIMUM WITHIN A SPECIFIED RANGE (15-16: MEANINGLESS ILLUSTRATION)
(min1b <- findmin(b1,m1,from=15,to=16))

# PLOT
plot(crosspred(b1,m1),ylab="RR",xlab="Temperature",xlim=c(0,25),
  ylim=c(0.9,1.3),lwd=1.5)
abline(v=min1)
abline(v=min1ci,lty=2)

################################################################################
# SECOND EXAMPLE: WHOLE DATA, FULL MODEL, BI-MENSIONAL EXPOSURE-LAG-RESPONSE

# LOAD DATA: LONDON 1993-2006
data <- read.csv("london.csv")
head(data)

# BI-DIMENSIONAL EXPOSURE-LAG-RESPONSE SPLINE
vk <- equalknots(data$tmean,fun="bs",df=4,degree=2)
lk <- logknots(25,3)
cb2 <- crossbasis(data$tmean, lag=25, argvar=list(fun="bs",degree=2,knots=vk),
  arglag=list(knots=lk))

# FULL MODEL WITH CONTROL FOR CONFOUNDING
m2 <- glm(death~cb2+ns(time,10*14)+dow,family=quasipoisson(),data)

# ESTIMATE MMT, WITH CI AND STANDARD ERROR
(min2 <- findmin(cb2,m2))
(min2ci <- quantile(findmin(cb2,m2,sim=T),c(2.5,97.5)/100))
(min2se <- sd(findmin(cb2,m2,sim=T)))

# IN PERCENTILE SCALE
sum(data$tmean<min2)/nrow(data)*100

# PLOT
cb2 <- crossbasis(data$tmean, lag=25, argvar=list(fun="bs",degree=2,knots=vk),
  arglag=list(knots=lk))
```

```
plot(crosspred(cb2,m2,cen=min2),"overall",ylab="RR",xlab="Temperature",
  xlim=c(-5,35),ylim=c(0.5,3.5),lwd=1.5)
abline(v=min2)
abline(v=min2ci,lty=2)

################################ END OF EXAMPLES #################################
```

```
##### SIMULATION STUDY TO EVALUATE BIAS AND COVERAGE OF THE FUNCTION findmin() #####
#
# The required data file london.csv is in the public domain, and available from
# http://www.ag-myresearch.com/bmcmrm2014.html
#
# UPDATE: 18 October 2016
# Future updates of this function can be found at http://www.ag-myresearch.com/
#
####################################################################################

library(dlnm) ; library(splines)
source("findmin.R")

################################################################################
# DEFINE A KNOWN EXPOSURE-RESPONSE ASSOCIATION

# LOAD A SUBSET OF THE DATA: LONDON JULY-DEC 2005
# (CONVENIENCE SAMPLE, LARGISH ASIMMETRIC CI)
data <- subset(read.csv("london.csv"), year==2005 & month>6)
head(data)

# A UNIDIMENSIONAL 4DF SPLINE
b <- onebasis(data$tmean,df=4)

# SIMPLE MODEL, WITH PREDICTED OUTCOME
m <- glm(death~b,family=poisson,data)
deathpred <- predict(m,type="response")

# REAL MINIMUM
(min <- findmin(b,m))

################################################################################
# RUN SIMULATION

# NUMBER OF SIMULATIONS
nsim <- 100

# GRID
summary(data$tmean)
at <- seq(0,24.3,by=0.1)

# CREATE THE OBJECT TO STORE THE INFO
res <- matrix(NA,nsim,6,dimnames=list(seq(nsim),
  c("est_min","bias","covered","true_at_CI_edge","boundary","est_SE")))

# RUN THE LOOP
set.seed(13041975)
for(i in seq(nsim)) {

  cat(i,"")

  # GENERATE SIMULATED DATA
  deathsim <- rpois(length(deathpred),deathpred)

  # FIT THE MODEL
  msim <- glm(deathsim~b,family=poisson,data)

  # FIND MIN, SAMPLE OF BOOTSTRAP ESTIMATES, AND CI AND SE FROM THOSE
  minsim   <- findmin(b,msim,at=at)
  minsimbs <- findmin(b,msim,,at=at,sim=T)
  mincisim <- quantile(minsimbs,c(2.5,97.5)/100)
  minsesim <- sd(minsimbs)

  # STORE THE DATA
  res[i,1] <- minsim
  res[i,2] <- min-minsim
  res[i,3] <- mincisim[1]<=min & mincisim[2]>=min
  res[i,4] <- mincisim[1]==min | mincisim[2]==min
```

```
  res[i,5] <- any(mincisim == range(at))
  res[i,6] <- minsesim
}

################################################################################
# ASSESS RESULTS

# BIAS
mean(res[,2])

# COVERAGE
mean(res[,3])*100

# % OF CIS WITH TRUE MIN AT EDGE (DISCRETE SPACE ISSUE)
mean(res[,4])*100

# PERCENTAGE OF CI AT BOUNDARY OF X SPACE
mean(res[,5])*100

# TRUE SAMPLING SD OF ESTIMATED MINIMA (EMPIRICAL SE)
# AND DISTRIBUTION OF ESTIMATED SES
sd(res[,1])
summary(res[,6])
mean(res[,6])

################################# END OF SIMULATION #################################
```

Table of performance for proposed bootstrap CI and alternative point estimates for minimum mortality temperature in above 1,000 simulations

Statistic	Model distribution: Poisson	Model distribution: Negative Binomial (phi=1.2)
Coverage (%) of 95% CIs (% of CIs including or touching true MMT)	96.4%	96.8%
of which, % with MMT on CI lower or upper limit	1.2%	0.9%
Mean SE (MMT) estimated by bootstrap	1.35 ºC	1.72 ºC
Median SE (MMT) estimated by bootstrap	0.92 ºC	1.23 ºC
Actual empirical *'true'* SE (SD of MMTs over simulations)	0.96 ºC	1.13 ºC
Sampling distribution of usual MMT estimates and of alternative bootstrap estimates		
Usual estimates		
Bias (difference from true MMT=16.1)	-0.05 ºC	-0.10 ºC
SD	0.96 ºC	1.13 ºC
95% fractile interval	15.1-17.5 ºC	15.0-17.6 ºC
Alternative bootstrap estimates		
Bias (difference from true MMT=16.1)	-0.07 ºC	-0.10 ºC
SD	0.83 ºC	1.06 ºC
95% fractile interval	14.6-17.5 ºC	13.9-17.9 ºC

MMT = minimum mortality temperature; *CI* = confidence interval; *SE* = standard error; *SD* = standard deviation.