6: Generalized Additive Models

John H Maindonald

April 3, 2018

Ideas and issues illustrated by the graphs in this vignette

Generalized Additive Models (GAMs) extend linear and generalized linear models to include smooth functions of explanatory variables, where the smoothness may be determined automatically. The graphs shown here illustrate some of the possibilities.

Note: Figure 6.9 shows the results of simulations. The version of this figure that is shown in Section 2 is, in order to keep to a minimum the time taken to process the vignette, for 25 simulations only. This is useful mainly as a check that the code does what is expected of it. More realistically, specify 500 or 1000 (as in the text) simulations.

1 Code for the Figures

```
fig6.1 <- function(plotit=TRUE){
    matohms <- data.frame(model.matrix(with(fruitohms, ~ poly(juice, 4))))
    names(matohms) <- c("Intercept", paste("poly4",1:4, sep=""))
    form <- formula(paste(paste(names(matohms), collapse="+"), "~ juice"))
    matohms$juice <- fruitohms$juice
    gph1 <- xyplot(form, data=matohms, layout=c(1,5), scales=list(tck=0.5),
                   ylab="Basis terms",
                   strip=strip.custom(strip.names=TRUE,
                   var.name="",
                   sep=expression(""),
                   factor.levels=c("Constant","Linear","Quadratic",
                                   "Cubic","Quartic")),
                   panel=function(x,y,...){
                       llines(smooth.spline(x,y))},
                   outer=TRUE,
                   legend=list(top=list(fun=grid::textGrob,
                               args=list(label="A: Basis functions",
                               just="left", x=0))))
    b <- coef(lm(I(ohms/1000) ~ poly(juice,4), data=fruitohms))
    matohms <- sweep(model.matrix(with(fruitohms, ~ poly(juice, 4))),
                     2, b, "*")
    matohms <- data.frame(matohms)
    names(matohms) <- c("Intercept", paste("poly4",1:4, sep=""))
    form <- formula(paste(paste(names(matohms), collapse="+"), "~ juice"))
    matohms$juice <- fruitohms$juice
    matohms$Kohms <- fruitohms$ohms/1000
    nam <- lapply(1:5, function(x)substitute(A %*% B,
                                         list(A=round(b[x],2),
                                              B=c("Constant","Linear",
                                              "Quadratic","Cubic",
                                              "Quartic")[x])))
    gph2 <- xyplot(form, data=matohms, layout=c(1,5),, scales=list(tck=0.5),
                   ylab="Add the contributions from these curves",
                   strip=strip.custom(strip.names=TRUE,
                   var.name="",
                   sep=expression(""),
                   factor.levels=as.expression(nam)),
                   panel=function(x,y,...){
                       llines(smooth.spline(x,y))},
                   outer=TRUE,
                   legend=list(top=list(fun=grid::textGrob,
                         args=list(label="B: Contribution to fitted curve",
                                   just="left", x=0))))
    if(plotit){
        print(gph1, position=c(0,0,.5,1))
        print(gph2, position=c(.5,0,1,1), newpage=FALSE)
}
    invisible(list(gph1, gph2))
}
```

```
fig6.2 <- function(){
    plot(ohms ~ juice, data=fruitohms, ylim=c(0, max(ohms)*1.02))
    ## 3 (=2+1) degrees of freedom natural spline
    fitns2 <- fitted(lm(ohms ~ splines::ns(juice, df=2), data=fruitohms))
    lines(fitns2 ~ juice, data=fruitohms, col="gray40")
    ## 4 (=3+1) degrees of freedom natural spline
    fitns3 <- fitted(lm(ohms ~ splines::ns(juice, df=3), data=fruitohms))
    lines(fitns3 ~ juice, data=fruitohms, lty=2, lwd=2, col="gray40")
    legend("topright", title="D.f. for cubic regression natural spline",
           legend=c("3  [ns(juice, 2)]",
           "4  [ns(juice, 3)]"),
           lty=c(1,2), lwd=c(1,2), cex=0.8)
    plot(ohms ~ juice, data=fruitohms, ylim=c(0, max(ohms)*1.02))
    ## 3 (=2+1) degrees of freedom natural spline
    fitns2 <- fitted(lm(ohms ~ splines::ns(juice, df=2), data=fruitohms))
    lines(fitns2 ~ juice, data=fruitohms, col="gray40")
    ## 4 (=3+1) degrees of freedom natural spline
    fitns3 <- fitted(lm(ohms ~ splines::ns(juice, df=3), data=fruitohms))
    lines(fitns3 ~ juice, data=fruitohms, lty=2, lwd=2, col="gray40")
    legend("topright", title="D.f. for cubic regression natural spline",
           legend=c("3  [ns(juice, 2)]",
           "4  [ns(juice, 3)]"),
           lty=c(1,2), lwd=c(1,2), cex=0.8)
}
```

```
fig6.3 <- function(){
    ohms.lm <- lm(ohms ~ ns(juice, df=3), data=fruitohms)
    termplot(ohms.lm, partial=TRUE, se=TRUE)
}
```

```
fig6.4 <- function(plotit=TRUE){
    matohms2 <- model.matrix(with(fruitohms, ~ splines::ns(juice, 2)))
    matohms3 <- model.matrix(with(fruitohms, ~ splines::ns(juice, 3)))
    m <- dim(matohms3)[1]
    longdf1 <- data.frame(juice=rep(fruitohms$juice,4),
                          basis2 = c(as.vector(matohms2),rep(NA,m)),
                          basis3 = as.vector(matohms3),
                          gp = factor(rep(c("Intercept",
                          paste("spline",1:3, sep="")),
                          rep(m,4))))
    gph1 <- xyplot(basis3 ~ juice | gp, data=longdf1, layout=c(1,4),
                   scales=list(tck=0.5),
                   ylab="Basis terms", strip=FALSE,
                   strip.left=strip.custom(strip.names=TRUE,
                   var.name="",
                   sep=expression(""),
                   factor.levels=c("Constant","Basis 1","Basis 2",
                   "Basis 3")),
                   par.settings=simpleTheme(lty=c(2,2,1,1)),
                   panel=function(x,y,subscripts){
                       llines(smooth.spline(x,y))
                       y2 <- longdf1$basis2[subscripts]
                       if(!any(is.na(y2))) llines(smooth.spline(x,y2),lty=1)},
                   outer=TRUE,
                   legend=list(top=list(fun=grid::textGrob,
                              args=list(label="A:Basis functions",
                              just="left", x=0))))
    b2 <- coef(lm(I(ohms/1000) ~ splines::ns(juice,2), data=fruitohms))
    b3 <- coef(lm(I(ohms/1000) ~ splines::ns(juice,3), data=fruitohms))
    spline2 <- as.vector(sweep(matohms2, 2, b2, "*"))
    spline3 <- as.vector(sweep(matohms3, 2, b3, "*"))
    longdf2 <- data.frame(juice=rep(fruitohms$juice,4),
                         spline2 = c(spline2, rep(NA,m)), spline3=spline3,
                         gp = factor(rep(c("Intercept",
                                     paste("spline",1:3, sep="")),
                         rep(m,4))))
    yran <- range(c(spline2, spline3))
    yran <- c(-6,8.5)
    gph2 <- xyplot(spline3 ~ juice | gp, data=longdf2, layout=c(1,4),
                  scales=list(tck=0.5, y=list(at=c(-4, 0, 4,8))), ylim=yran,
                  ylab="Add these contributions (ohms x 1000)", strip=FALSE,
                  strip.left=strip.custom(strip.names=TRUE,
                  var.name="",
                  sep=expression(""),
                  factor.levels=c("Const","Add 1","Add 2","Add 3")),
                  par.settings=simpleTheme(lty=c(2,2,1,1)),
                  panel=function(x,y,subscripts){
                      llines(smooth.spline(x,y))
                      y2 <- longdf2$spline2[subscripts]
                      if(!any(is.na(y2))) llines(smooth.spline(x,y2),lty=1)},
                  outer=TRUE,
                  legend=list(top=list(fun=grid::textGrob,
                              args=list(label="B: Contribution fo fitted curve",
                              just="left", x=0))))
    if(plotit){
        print(gph1, position=c(0,0,.5,1))
        print(gph2, position=c(.5,0,1,1), newpage=FALSE)
    }
    invisible(list(gph1, gph2))
}
```

```
fig6.5 <- function(){
    res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
    wr.gam <- gam(res ~ s(log(Distance)), data=worldRecords)
    plot(wr.gam, residuals=TRUE, pch=1, las=1, ylab="Fitted smooth")
}
```

```r
fig6.6 <-
function () {
    res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
    wr.gam <- gam(res ~ s(log(Distance)), data=worldRecords)
    gam.check(wr.gam)
}
```

```r
fig6.7 <-
function (mf=3,nf=2)
{
  opar <- par(mfrow=c(mf,nf), mar=c(0.25, 4.1, 0.25, 1.1))
    set.seed(29)         # Ensure exact result is reproducible
    res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
    npanels <- mf*nf
    for(i in 1:npanels){
        permres <- sample(res)  # Random permutation
                                # 0 for left-handers;  1 for right
        perm.gam <- gam(permres ~ s(log(Distance)), data=worldRecords)
        plot(perm.gam, las=1, rug=if(i<5) FALSE else TRUE, ylab="Fit")
    }
  par(opar)
}
```

```r
fig6.8 <- function(){
    meuse.gam <- gam(log(lead) ~ s(elev) + s(dist) + ffreq + soil,
                       data=meuse)
    plot(meuse.gam, residuals=TRUE, se=TRUE, pch=1)
    termplot(meuse.gam, terms="ffreq", se=TRUE)
    termplot(meuse.gam, terms="soil", se=TRUE)
}
```

```r
fig6.9 <- function(nsim=1000, caption=NULL){
  opar <- par(mfrow=c(1,2), oma=c(0,0,1.6,0.6))
  if(missing(caption))captCol <- "black" else captCol <- "blue"
  if(is.null(caption))caption <- paste("Graphs are from", nsim, "simulations")
    if(!exists("meuseML.gam"))
    meuseML.gam <- gam(log(lead) ~ s(elev) + s(dist) + ffreq + soil,
                       data=meuse, method="ML")
    if(!exists("meusexML.gam"))
    meusexML.gam <- gam(log(lead) ~ s(elev, dist) + ffreq + soil,
                        data=meuse, method="ML")
    ## Now simulate from meuseML.gam
    simY <- simulate(meuseML.gam, nsim=nsim)
    simResults <- matrix(0, nrow=nsim, ncol=3)
    colnames(simResults) <- c( "deltaDf", "fsim", "psim")
    for(i in 1:nsim){
        mML.gam <- gam(simY[,i] ~ s(elev) + s(dist) + ffreq + soil,
                       data=meuse, method="ML")
        mxML.gam <- gam(simY[,i] ~ s(elev, dist) + ffreq + soil,
                        data=meuse, method="ML")
        aovcomp <- anova(mML.gam, mxML.gam, test="F")
        simResults[i,] <- unlist(aovcomp[2, c("Df", "F", "Pr(>F)")])
    }
    ## Now plot the £p£-statistics and £F£-statistics
    ## against the change in degrees of freedom:
    colcode <- c("gray", "black")[1+(simResults[,"deltaDf"]>=1)]
    simResults <- as.data.frame(simResults)
    plot(psim ~ deltaDf, log="y", xlab="Change in degrees of freedom",
         ylab=expression(italic(p)*"-value"), col=colcode, data=simResults)
    abline(v=1, lty=2, col="gray")
    mtext("A", side=3, line=0.25, adj=0)
    mtext("1", side=1, at=1, line=0, cex=0.75)
    plot(fsim ~ deltaDf, log="y", xlab="Change in degrees of freedom",
         ylab=expression(italic(F)*"-statistic"),  col=colcode,
         data=simResults)
    abline(v=1, lty=2, col="gray")
    mtext("1", side=1, at=1, line=0, cex=0.75)
    mtext("B", side=3, line=0.25, adj=0)
  mtext(side=3, line=0.5, adj=0.052, outer=TRUE, caption, col=captCol)
    invisible(simResults)
  par(opar)
}
```

```
fig6.10A <- function(){
    if(!exists("meuseML.gam"))
    meuseML.gam <- gam(log(lead) ~ s(elev) + s(dist) + ffreq + soil,
                       data=meuse, method="ML")
    plot(meuseML.gam)
    termplot(meuseML.gam, terms="ffreq", se=TRUE)
    termplot(meuseML.gam, terms="soil", se=TRUE)
    mtext(side=3, line=0.65, "A: Add effects of dist and elev", outer=TRUE,
          cex=0.8, adj=0)
}
```

```
fig6.10B <- function(){
    if(!exists("meusexML.gam"))
    meusexML.gam <- gam(log(lead) ~ s(elev, dist) + ffreq + soil,
                        data=meuse, method="ML")
    plot(meusexML.gam)
    termplot(meusexML.gam, terms="ffreq", se=TRUE)
    termplot(meusexML.gam, terms="soil", se=TRUE)
    mtext(side=3, line=0.65, "B: Fit surface to dist and elev", outer=TRUE,
      cex=0.8, adj=0)
}
```

```
fig6.10 <- function()
print("Run fig6.10A() and fig6.10B() separately")
```

```
fig6.11 <- function(){
    hand <- with(cricketer, as.vector(as.vector(unclass(left)-1)))
                                    # 0 for left-handers
                                    # 1 for right
    hand.gam <- gam(hand ~ s(year), data=cricketer, family=binomial)
    plot(hand.gam, las=1, xlab="", ylab="Pr(left-handed)",
         trans=function(x)exp(x)/(1+exp(x)),
         shift=mean(predict(hand.gam)))
}
```

```
fig6.12 <- function(mf=3,nf=2){
    opar <- par(mfrow=c(mf,nf), mar=c(0.25, 4.1, 0.25, 1.1))
    npanel <- mf*nf
    for(i in 1:npanel){
        hand <- sample(c(0,1), size=nrow(cricketer), replace=TRUE,
                       prob=c(0.185, 0.815))
                                    # 0 for left-handers
                                    # 1 for right
        hand.gam <- gam(hand ~ s(year), data=cricketer, family=binomial)
        plot(hand.gam, las=1, xlab="",
             rug=if(i<4)FALSE else TRUE,
             trans=function(x)exp(x)/(1+exp(x)),
             shift=mean(predict(hand.gam)))
        }
        par(opar)
}
```

```
fig6.13 <- function(){
    rtlef <- data.frame(with(cricketer, as(table(year, left), "matrix")))
    rtlef <- within(rtlef, year <- as.numeric(rownames(rtlef)))
    right.gam <-  gam(right ~ s(year), data=rtlef, family=poisson)
    left.gam <-  gam(left ~ s(year), data=rtlef, family=poisson)
    rtlef <- within(rtlef,
                {fitright <- predict(right.gam, type="response")
                 fitleft <- predict(left.gam, type="response")})
    key.list <- list(text=expression("Right-handers", "Left-handers",
        "Left-handers "%*%" 4"),
                     corner=c(0,1), x=0, y=0.985,
                     points=FALSE, lines=TRUE)
    parset <- simpleTheme(col=c("blue", "purple", "purple"),
                          lty=c(1,1,2), lwd=c(2,2, 1))
    gph <- xyplot(fitright+fitleft+I(fitleft*4) ~ year, data=rtlef,
                  auto.key=key.list, par.settings=parset,tck=-0.05,
                  xlab="",
                  ylab="Number of cricketers\nborn in given year",
                  type="l", ylim=c(0,70))
    print(gph)
}
```

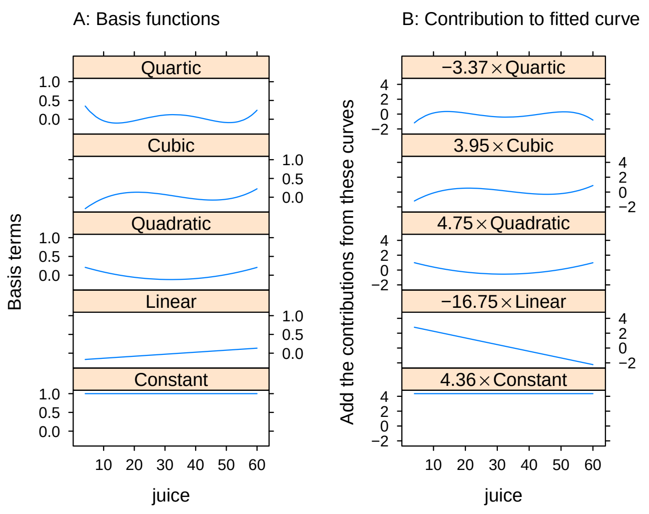
```
countAccs <- function(data=airAccs, dateCol="Date",
                      fromDate = as.Date("2006-01-01"),
                      by="1 day", prefix="num"){
  dfCount <- eventCounts(data, dateCol=dateCol,
                       from= fromDate, by=by,
                       prefix=prefix)
  dfCount[, "day"] <- julian(dfCount[,"Date"], origin=fromDate)
  dfCount
}
##
fig6.14 <- function()
    print("Run the separate functions fig6.14A() and fig6.14B()")
fig6.14A <- function(fromDate=as.Date("2006-01-01"), basis.df=50){
  if(!exists('dfDay06'))dfDay06 <- countAccs(fromDate=fromDate)
year <- seq(from=fromDate, to=max(dfDay06$Date), by="1 year")
atyear=julian(year, origin=fromDate)
dfDay06.gam <-
  gam(formula = num ~ s(day, k=basis.df), family = quasipoisson,
      data = dfDay06)
av <- mean(predict(dfDay06.gam))
plot(dfDay06.gam, xaxt="n", shift=av, trans=exp, rug=FALSE, xlab="",
     ylab="Estimated rate per day")
axis(1, at=atyear, labels=format(year, "%Y"))
mtext(side=3, line=0.75, "A: Events per day, vs date", adj=0)
}
fig6.14B <- function(fromDate=as.Date("2006-01-01"), basis.df=50){
  if(!exists('dfWeek06'))dfWeek06 <- countAccs(fromDate=fromDate,
                                               by="1 week")
dfWeek06.gam <- gam(num~s(day, k=basis.df), data=dfWeek06,
                    family=quasipoisson)
av <- mean(predict(dfWeek06.gam))
year <- seq(from=fromDate, to=max(dfWeek06$Date), by="1 year")
atyear=julian(year, origin=fromDate)
plot(dfWeek06.gam, xaxt="n", shift=av, trans=exp, rug=FALSE, xlab="",
      ylab="Estimated rate per week")
axis(1, at=atyear, labels=format(year, "%Y"))
mtext(side=3, line=0.75, "B: Events per week, vs date", adj=0)
}
```

2 Show the Figures

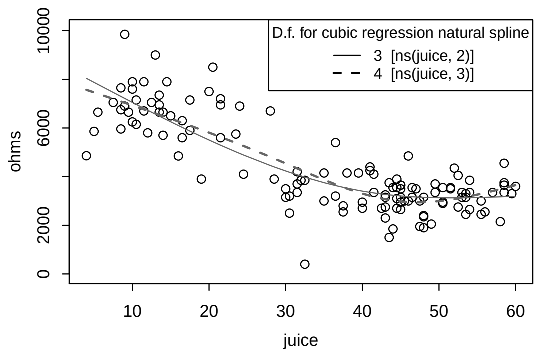
```
pkgs <- c("DAAG","mgcv","splines","gamclass")
z <- sapply(pkgs, require, character.only=TRUE, warn.conflicts=FALSE)
if(any(!z)){
  notAvail <- paste(names(z)[!z], collapse=", ")
  print(paste("The following packages require to be installed:", notAvail))
}
```

```
if(!exists("meuse")){
    msg <- "Cannot find package 'sp',"
        if(!require("sp"))
          return(paste(msg, "cannot do graph."))
    data("meuse", package="sp", envir = environment())
    }
```

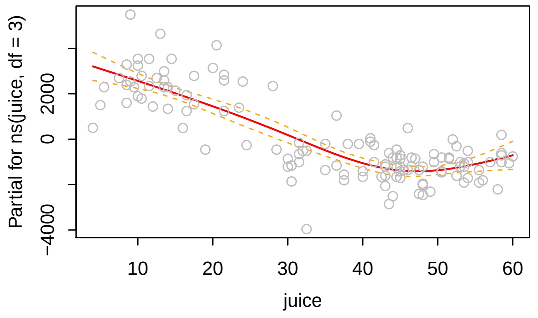
```
fig6.1()
```


A: Basis functions
B: Contribution to fitted curve
Basis terms
Add the contributions from these curves
Quartic
Cubic
Quadratic
Linear
Constant
−3.37 × Quartic
3.95 × Cubic
4.75 × Quadratic
−16.75 × Linear
4.36 × Constant
juice
juice

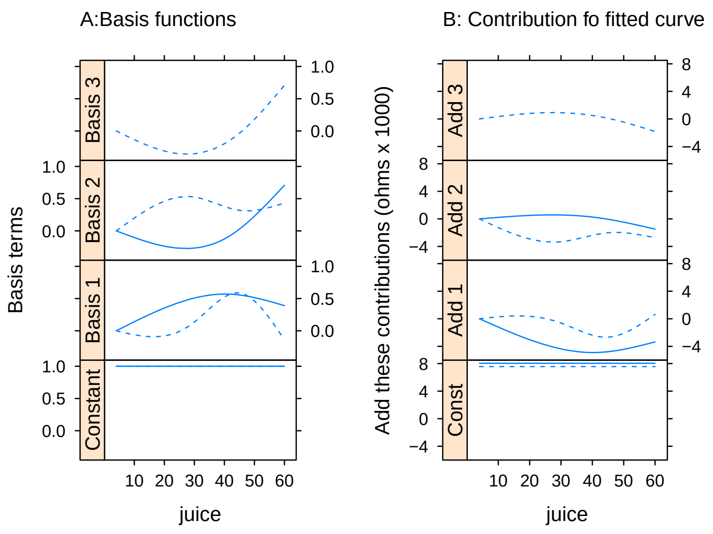
```
fig6.2()
```


D.f. for cubic regression natural spline
3 [ns(juice, 2)]
4 [ns(juice, 3)]
ohms
juice

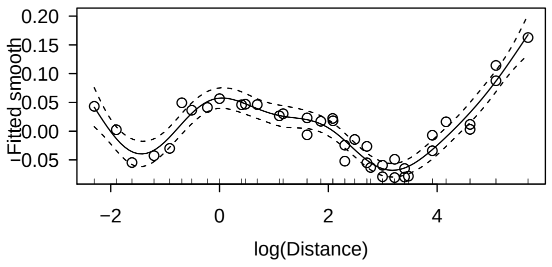
```
fig6.3()
```


Partial for ns(juice, df = 3)
juice

```
fig6.4()
```

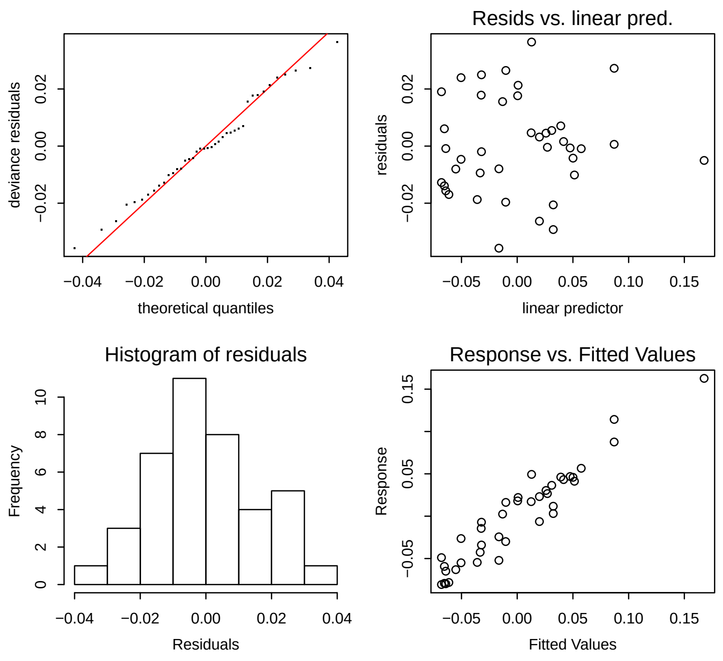


```
fig6.5()
```

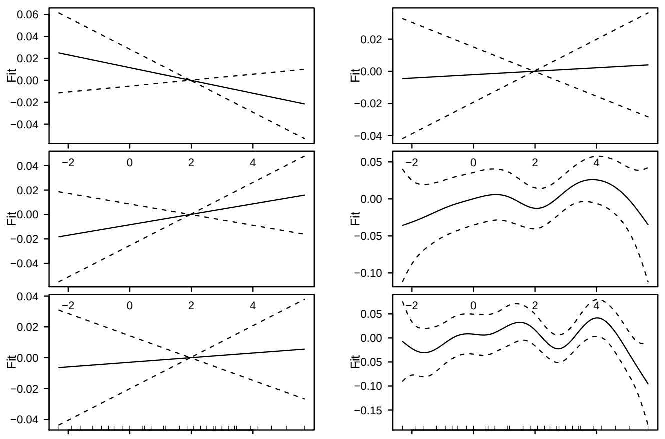


```
fig6.6()


Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 6 iterations.
The RMS GCV score gradient at convergence was 7.277e-07 .
The Hessian was positive definite.
Model rank =  10 / 10

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'.
```

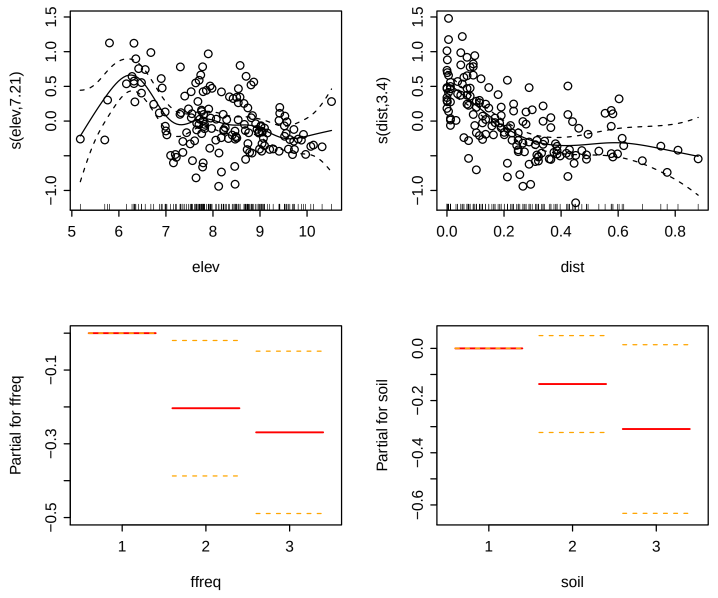
```
                     k'   edf k-index p-value
s(log(Distance)) 9.00 8.32     1.16     0.77
```



```
fig6.7()
```

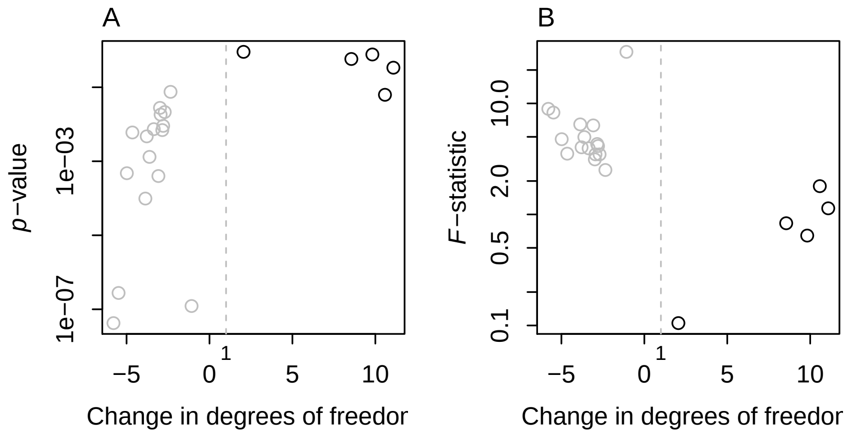


```
if(exists("meuse")) fig6.8() else
  print("Cannot locate data set 'meuse', hence cannot plot")
```



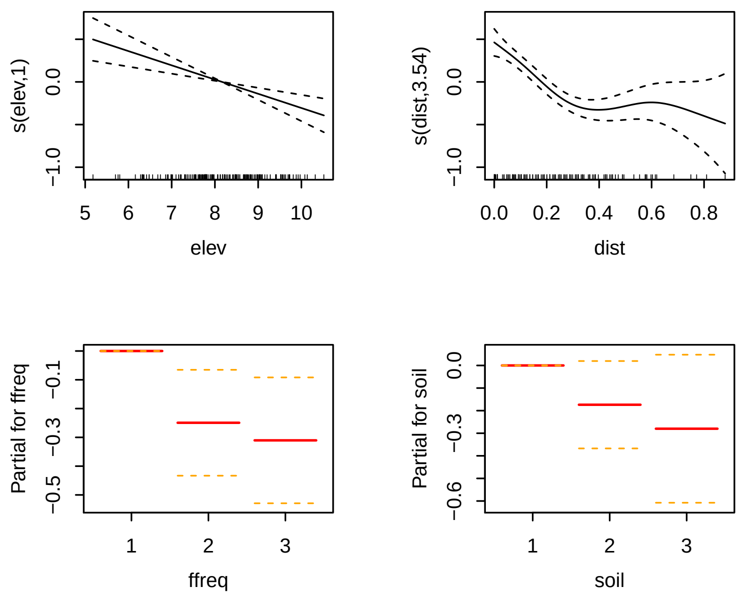
```
{
caption <- paste("These are from 25 simulations.",
                 "More usefully, try, eg: fig6.9(nsim=500)")
if(exists("meuse")) fig6.9(nsim=25, caption=caption) else
  print("Cannot locate data set 'meuse', hence cannot plot")
}
```

These are from 25 simulations. More usefully, try, eg: fig6.9(nsim=500)



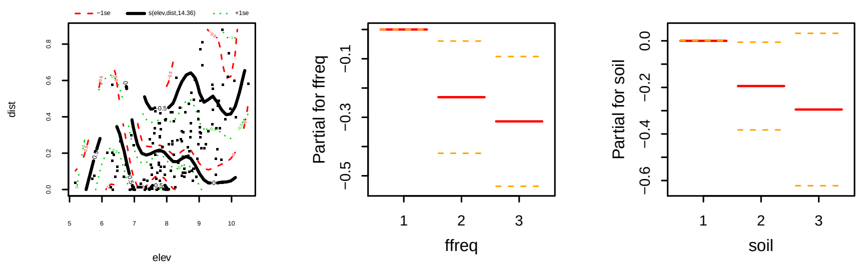
```
if(exists("meuse")) fig6.10A() else
  print("Cannot locate data set 'meuse', hence cannot plot")
```

A: Add effects of dist and elev

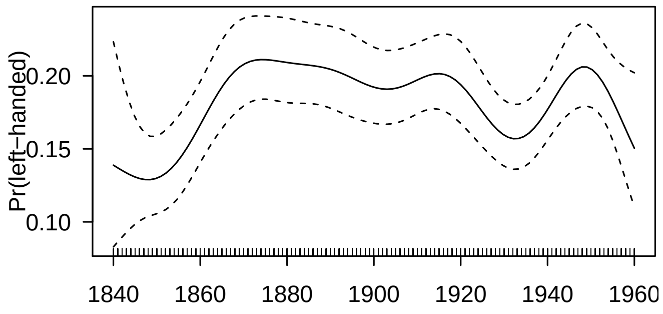


```
if(exists("meuse")) fig6.10B() else
  print("Cannot locate data set 'meuse', hence cannot plot")
```

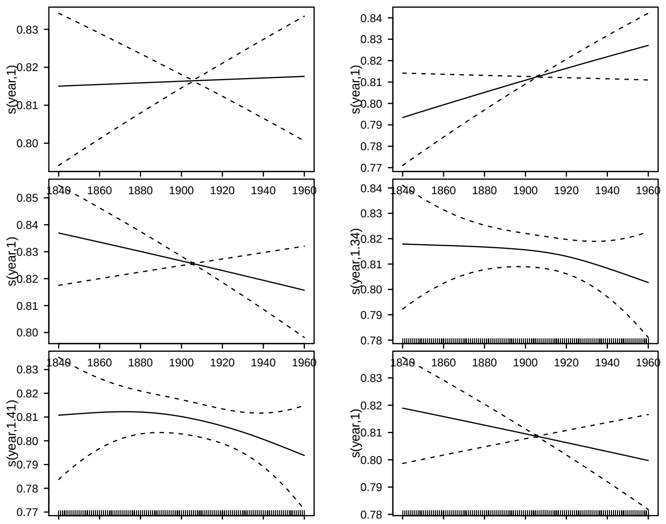
B: Fit surface to dist and elev



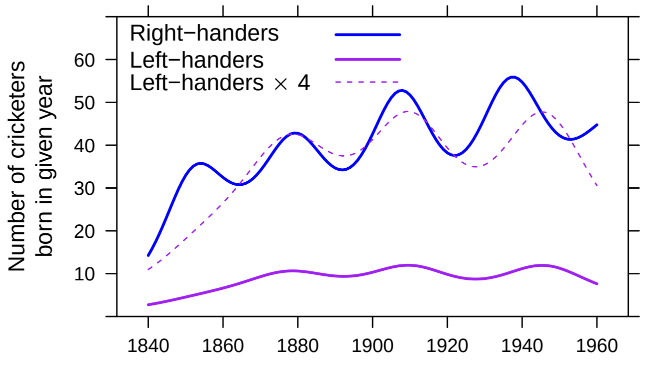
```
fig6.11()
```



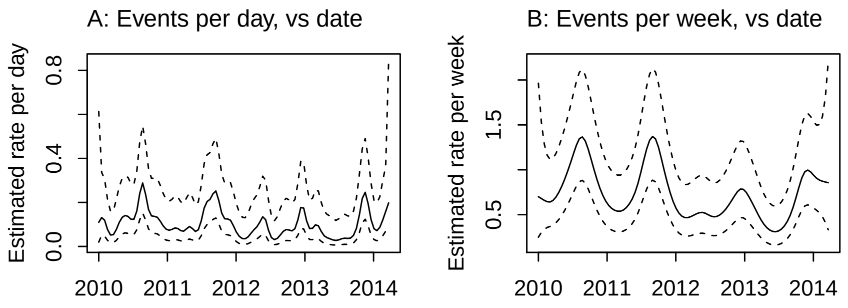
```
fig6.12()
```



```
fig6.13()
```



```
{
  fig6.14A(fromDate=as.Date("2010-01-01"), basis.df=50)
  fig6.14B(fromDate=as.Date("2010-01-01"), basis.df=50)
}
```


A: Events per day, vs date
Estimated rate per day
B: Events per week, vs date
Estimated rate per week