Figures for Chapter 6

John H Maindonald

October 28, 2012

```
fig6.1 <- function(plotit=TRUE){
    library(lattice); library(grid)
    library(DAAG)
    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=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=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(){
    library(splines)
    library(DAAG)
    plot(ohms ~ juice, data=fruitohms, ylim=c(0, max(ohms)*1.02))
    ## 3 (=2+1) degrees of freedom natural spline
    fitns2 <- fitted(lm(ohms ~ 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 ~ 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)
    library(splines)
    library(DAAG)
    plot(ohms ~ juice, data=fruitohms, ylim=c(0, max(ohms)*1.02))
    ## 3 (=2+1) degrees of freedom natural spline
    fitns2 <- fitted(lm(ohms ~ 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 ~ 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(plotit=TRUE){
```

```
library(lattice)
library(grid)        # Supplies the function textGrob
matohms2 <- model.matrix(with(fruitohms, ~ ns(juice, 2)))
matohms3 <- model.matrix(with(fruitohms, ~ 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=textGrob,
                          args=list(label="A: Basis functions",
                          just="left", x=0))))
b2 <- coef(lm(I(ohms/1000) ~ ns(juice,2), data=fruitohms))
b3 <- coef(lm(I(ohms/1000) ~ 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=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))
}

fig6.4 <- function(){
    library(mgcv)
    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")
}

fig6.5 <-
function () {
    library(mgcv)
    res <- resid(lm(log(Time) ~ log(Distance), data=worldRecords))
    wr.gam <- gam(res ~ s(log(Distance)), data=worldRecords)
    gam.check(wr.gam)
}

fig6.6 <-
function ()
{
    opar <- par(mfrow=c(3,2), 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))
    for(i in 1:6){
        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)
}

fig6.7 <- function(){
```

```
    cat('Run the separate functions fig6.7A() and fig6.7B()\n')
    }

fig6.7A <- function(){
    library(sp)
    data(meuse)
    meuse$ffreq <- factor(meuse$ffreq)
    meuse$soil <- factor(meuse$soil)
    ## Model 2ML: s(elev, dist) (smooth surface), plus factors
    formxML <- log(lead) ~ s(elev, dist) + ffreq + soil
    meusexML.gam <- gam(formxML, method="ML", data=meuse)
    opar <- par(mar=c(4.1,3.6,2.1, 1.6), mex=0.8,
                oma=c(0,0,2.1,0), mfrow=c(3,1))
    plot(meusexML.gam)
    termplot(meusexML.gam, terms="ffreq", se=TRUE)
    mtext(side=3, line=0.65, "A: One pattern of change", outer=TRUE,
          cex=0.8, adj=0)
    termplot(meusexML.gam, terms="soil", se=TRUE)
    par(opar)
}

fig6.7B <- function(){
    opar <- par(mar=c(4.1,3.6,2.1, 1.6), mex=0.8,
                oma=c(0,0,2.1,0), mfrow=c(3,1))
    formxxML <- log(lead) ~ s(elev, dist, by=ffreq) + ffreq + soil
    meusexxML.gam <- gam(formxxML, method="ML", data=meuse)
    plot(meusexxML.gam)
    mtext(side=3, line=0.65, "B: Contours change with ffreq", outer=TRUE,
          cex=0.8, adj=0)
    par(opar)
}

fig6.8 <- function(){
    library(sp)
    data(meuse)
    meuse$ffreq <- factor(meuse$ffreq)
    meuse$soil <- factor(meuse$soil)
    meuse.gam <- gam(log(lead) ~ s(elev) + s(dist) + ffreq + soil,
                     data=meuse)
    opar <- par(mar=c(3.6,3.1,2.1, 1.6), mgp=c(2.25, 0.5, 0),
                oma=c(0,0,2.1,0), mfrow=c(2,2))
    plot(meuse.gam, residuals=TRUE, se=TRUE, pch=1)
    termplot(meuse.gam, terms="ffreq", se=TRUE)
    termplot(meuse.gam, terms="soil", se=TRUE)
    par(opar)
}
```

```
fig6.9 <- function(){
    meuseML.gam <- gam(log(lead) ~ s(elev) + s(dist) + ffreq + soil,
                       data=meuse, method="ML")
    meusexML.gam <- gam(log(lead) ~ s(elev, dist) + ffreq + soil,
                        data=meuse, method="ML")
    pval <- anova(meuseML.gam, meusexML.gam, test="F")["Pr(>F)"][2,]
    ## Now simulate from meuseML.gam
    hat <- predict(meuseML.gam)
    simY <- simulate(meuseML.gam, nsim=1000)
    f1000 <- p1000 <- deltaDf <- numeric(1000)
    for(i in 1:1000){
        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")
        f1000[i] <- aovcomp["F"][2,]
        p1000[i] <- aovcomp["Pr(>F)"][2,]
        deltaDf[i] <- aovcomp[2, "Df"]
    }
    ## Now plot the $p$-statistics and $F$-statistics
    ## against the change in degrees of freedom:
    par(mfrow=c(1,2))
    colcode <- c("gray", "black")[1+(deltaDf>=1)]
    plot(p1000 ~ deltaDf, log="y", xlab="Change in degrees of freedom",
         ylab=expression(italic(p)*"-value"), col=colcode)
    abline(v=1, lty=2, col="gray")
    mtext("A", side=3, line=0.75, adj=0)
    mtext("1", side=1, at=1, line=0, cex=0.75)
    plot(f1000 ~ deltaDf, log="y", xlab="Change in degrees of freedom",
         ylab=expression(italic(F)*"-statistic"),  col=colcode)
    abline(v=1, lty=2, col="gray")
    mtext("1", side=1, at=1, line=0, cex=0.75)
    mtext("B", side=3, line=0.75, adj=0)
    par(mfrow=c(1,1))
}

fig6.10 <- function(){
    par(mfrow=c(1,2))
    mdbRain.gam <- gam(mdbRain ~ s(Year) + s(SOI), data=bomregions)
    plot(mdbRain.gam, residuals=TRUE, se=2, pch=1, cex=0.5, select=1)
    plot(mdbRain.gam, residuals=TRUE, se=2, pch=1, cex=0.5, select=2)
    par(mfrow=c(1,1))
}

fig6.11 <- function(){
library(DAAG)
```

```
Erie <- greatLakes[,"Erie"]
plot(Erie, xlab="",
     ylab="Level (m)")
}

fig6.12 <- function(){
    library(DAAG)
    Erie <- greatLakes[,"Erie"]
    opar <- par(oma=c(0,0,4,0))
    lag.plot(Erie, lags=3,
             do.lines=FALSE,
             layout=c(2,3), main="")
    mtext(side=3, line=3, adj=-0.155,
          "A: Lag plots, for lags 1, 2 and 3 respectively", cex=1)
    par(fig=c(0,1,0,0.6), new=TRUE)
    par(mar=c(2.75, 3.1, 3.6, 1.6))
    acf(Erie, main="", xlab="")
    mtext(side=3, line=0.5, "B: Autocorrelation estimates at successive lags",
          adj=-0.35, cex=1)
    mtext(side=1, line=1.75, "Lag", cex=1)
    par(opar)
}

fig6.13 <- function(){
    library(DAAG)
    Erie <- greatLakes[,"Erie"]
    df <-  data.frame(height=as.vector(Erie), year=time(Erie))
    obj <- gam(height ~ s(year), data=df)
    plot(obj, shift=mean(df$height), residuals=T, pch=1, xlab="")
}

fig6.14 <- function(){
    library(DAAG)
    Erie <- greatLakes[,"Erie"]
    erie.ar <- ar(Erie)
    library(forecast)
    plot(forecast(erie.ar, h=15), ylab="Lake level (m)")
}

fig6.15 <- function(){
    opar <- par(mfrow=c(3,2), mar=c(0.25, 4.1, 0.25, 1.1))
    for(i in 1:6){
        df <- data.frame(x=1:200, y=arima.sim(list(ar=0.7), n=200))
        df.gam <- gam(y ~ s(x), data=df)
        plot(df.gam, residuals=TRUE)
    }
    par(opar)
}
```

```
fig6.16 <- function(){
    library(mgcv)
    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)))
}

fig6.17 <- function(){
    opar <- par(mfrow=c(3,2), mar=c(0.25, 4.1, 0.25, 1.1))
    for(i in 1:6){
        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)
}

fig6.18 <- 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)
}

fig6.1()
fig6.2()
fig6.3()
fig6.4()
fig6.5()
fig6.6()
fig6.7()
fig6.7A()
fig6.7B()
fig6.8()
fig6.9()
fig6.10()
fig6.11()
fig6.12()
fig6.13()
fig6.14()
fig6.15()
fig6.16()
fig6.17()
fig6.18()
```