9: Further Methods

John H Maindonald

March 16, 2018

```
doFigs <- FALSE
```

```
fig9.1 <- function(){
## ---- smooth-ohms ----
## Plot points
plot(ohms ~ juice, data=fruitohms)
## Add smooth curve, using default
## smoothing window
with(fruitohms,
     lines(lowess(ohms ~ juice), col="gray", lwd=2))
}
```

```
fig9.3 <- function(){
## ---- ant111b ----
# ant111b is in DAAG
Site <- with(ant111b, reorder(site, harvwt,
                              FUN=mean))
stripplot(Site ~ harvwt, data=ant111b,
          scales=list(tck=0.5),
          xlab="Harvest weight of corn")
}
```

```
fig9.4 <- function(){
## ---- Erie1 ----
plot(Erie, xlab="",
     ylab="Level (m)")
}
```

```
fig9.2A <- function(){
    if(!exists("eyeAmpM.gam"))
eyeAmpM.gam <- mgcv::gam(amp ~ s(x,y), data=subset(eyeAmp, Sex=="m"))
        if(!exists("eyeAmpF.gam"))
eyeAmpF.gam <- mgcv::gam(amp ~ s(x,y), data=subset(eyeAmp, Sex=="f"))
lims <- range(c(predict(eyeAmpF.gam), predict(eyeAmpM.gam)))
mgcv::vis.gam(eyeAmpM.gam, plot.type='contour', color="cm", zlim=lims, main="")
}
fig9.2B <- function(){
    if(!exists("eyeAmpM.gam"))
    eyeAmpM.gam <- mgcv::gam(amp ~ s(x,y), data=subset(eyeAmp, Sex=="m"))
    if(!exists("eyeAmpF.gam"))
    eyeAmpF.gam <- mgcv::gam(amp ~ s(x,y), data=subset(eyeAmp, Sex=="f"))
lims <- range(c(predict(eyeAmpF.gam), predict(eyeAmpM.gam)))
mgcv::vis.gam(eyeAmpF.gam, plot.type='contour', color="cm", zlim=lims, main="")
}
fig9.2 <- function(){
print("Run the separate figures fig9.2A() and fig9.2B()")
}
```

```
fig9.5A <- function(){
## ---- lagErie ----
## Panel A
lag.plot(Erie, lags=3,
         do.lines=FALSE,
         layout=c(1,3),
         main="  ")
mtext(side=3, line=2, adj=0,
      "A: Lag plots, at lags of 1, 2 and 3")
}
fig9.5B <- function(){
## ---- acfErie ----
## Panel B
acf(Erie, main="")
mtext(side=3, line=1, adj=0, cex=1.25,
      "B: Autocorrelation function")
}
fig9.5 <- function(){
"Run the separate functions fig9.5A() and fig9.5B()."
}
```

```
fig9.6 <- function(){
## ---- gamErie ----
df <-  data.frame(
```

```
  height=as.vector(Erie),
  year=time(Erie))
obj <- mgcv::gam(height ~ s(year), data=df)
plot(obj,
     shift=mean(df$height),
     residuals=TRUE, pch=1,
     xlab="",
     ylab="Height of lake")
}
```

```
fig9.7 <- function(){
## ---- arima-sim ----
for (i in 1:6){
ysim <- arima.sim(list(ar=0.85),
                  n=200)
df <- data.frame(x=1:200,
                 y=ysim)
df.gam <- mgcv::gam(y ~ s(x),
               data=df)
plot(df.gam,
     ylab=paste("Sim", i),
     residuals=TRUE)
}
}
```

```
fig9.8 <- function(){
## ---- Erie-fcast ----
erie.ar <- ar(Erie, order.max=1)
fc <- forecast::forecast(erie.ar, h=15)
plot(fc, main="",
     ylab="Lake level (m)")
  # 15 time points ahead
}
```

```
fig9.9 <- function(){
## ---- mdb-gam ----
mdbRain.gam <- mgcv::gam(mdbRain ~ s(Year) + s(SOI),
                   data=bomregions)
plot(mdbRain.gam, residuals=TRUE, se=2, pch=1,
     cex=0.5)
}
```

```
fig9.10 <- function(){
## ---- ar1sims ----
opar <- par(mar=c(2.1, 3.1, 0.25, 1.1), mgp=c(2.25,0.5,0))
mdbRain.gam <- mgcv::gam(mdbRain ~ s(Year) + s(SOI),
                         data=bomregions)
n <-  dim(bomregions)[1]
acf(resid(mdbRain.gam), ylab="MDB series")
for(i in 1:5)acf(rnorm(n), ylab=paste("Sim",i),
                 col="gray40")
par(opar)
}
```

```
library(DAAGviz, quietly = TRUE)
```

```
if(!requireNamespace('gamclass', quietly=TRUE))print('gamclass must be installed')
fromDate <- as.Date("2006-01-01")
dfDay06 <- gamclass::eventCounts(data=gamclass::airAccs, dateCol="Date",
                       from= fromDate, by='1 day', prefix="num")
dfDay06$day <- julian(dfDay06$Date, origin=fromDate)
dfWeek06 <- gamclass::eventCounts(data=gamclass::airAccs, dateCol="Date",
                        from=fromDate,
                        by='1 week', prefix="num")
dfWeek06$day <- julian(dfWeek06$Date, origin=fromDate)
year <- seq(from=fromDate, to=max(dfDay06$Date), by="1 year")
atyear=julian(year, origin=fromDate)
```

```
fig9.11A <- function(){
  if(!exists('dfday06')){
    print(paste("Unable to plot graph.",
                "Ensure package 'gamclass' is installed."))
    return()
  }
dfDay06.gam <-
  mgcv::gam(formula = num ~ s(day, k=200), 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)
}
fig9.11B <- function(){
dfWeek06.gam <- mgcv::gam(num~s(day, k=200), data=dfWeek06, family=quasipoisson)
```

```
av <- mean(predict(dfWeek06.gam))
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)
}
fig9.11 <- function(){
    "Run the separate functions fig9.10A() & fig9.10B()"
}
```

```
fig9.12 <- function(){
## ---- fgl-scores2D ----
scores <- predict(fgl.lda)$x
xyplot(scores[,2] ~ scores[,1], groups=fgl$type,
       xlab="Discriminant 1",
       ylab="Discriminant 2",
       aspect=1, scales=list(tck=0.4),
       auto.key=list(space="right"),
       par.settings=simpleTheme(alpha=0.6, pch=1:6))
}
```

```
## ---- bronchit-first3 ----
## ---- bronchit-rp ----
set.seed(47)   # Reproduce tree shown
fig9.13 <- function(){
## ---- treefig ----
opar <- par(mar=rep(1.1,4))
if(!exists('b.rpart'))b.rpart <- rpart(rfac ~ cig+poll, data=bronchit)
plot(b.rpart)
text(b.rpart, xpd=TRUE)
par(opar)
}
```

```
fig9.14 <- function(){
## ---- rf-x-bronchit ----
opar <- par(mar=rep(2.1,4))
set.seed(31)   # Reproduce the trees shown
num <- 1:nrow(bronchit)
for(i in 1:12){
  useobs <- sample(num, replace=TRUE)
  dset <- bronchit[useobs, ]
  b.rpart <- rpart(rfac ~ cig+poll, data=dset,
                   control=rpart.control(maxdepth=2))
```

```
  plot(b.rpart, uniform=TRUE)
  text(b.rpart, xpd=TRUE, cex=1.2)
}
par(opar)
}
```

```
fig9.15 <- function(){
## ---- proximity-plot ----
parset <- simpleTheme(pch=1:2)
if(!requireNamespace('randomForest'))
  return('Fig 9.15 requires randomForest package')
bronchit.rf <- randomForest::randomForest(rfac ~ cig+poll,
                            proximity=TRUE,
                            data=bronchit)
points <- cmdscale(1-bronchit.rf$proximity)
xyplot(points[,2] ~ points[,1],
       groups=bronchit$rfac,
       xlab="Axis 1", ylab="Axis 2",
       par.settings=parset, aspect=1,
       auto.key=list(columns=2))
}
```

```
fig9.16 <- function(){
opar <- par(mar=c(2,3,2,5))
plot(aupts, axes=FALSE, ann=FALSE, fg="gray",
     frame.plot=TRUE)
city <- rownames(aupts)
pos <- rep(1,length(city))
pos[city=="Melbourne"]<- 3
pos[city=="Canberra"] <- 4
pos[city=="Sydney"] <- 4
par(xpd=TRUE)
text(aupts, labels=city, pos=pos)
par(opar)
}
```

Figure 1: Resistance in ohms is plotted against apparent juice content. A smooth curve (in gray) has been added, using the `lowess` smoother. The width of the smoothing window was the default fraction $f = \frac{2}{3}$ of the range of values of the x-variable.

Figure 2: Estimated contours of left eye responses to visual stimulae, projected onto the plane.

```
figset9 <- function(){
    library(MASS)
    library(lattice)
    library(DAAGviz)
    if(!require('DAAG', quietly=TRUE))stop('DAAG must be installed')
    if(!requireNamespace('mgcv', quietly=TRUE))stop('mgcv must be installed')
    if(!requireNamespace('oz', quietly=TRUE))stop('oz must be installed')
  if(!requireNamespace('ggplot2', quietly=TRUE))
    stop('ggplot2 must be installed')
}
```

```
figset9()


Attaching package:  'DAAG'
The following object is masked from 'package:MASS':
      hills

  ## ---- bronchit-rfac ----
## Now make the outcome variable a factor
bronchit <-
  within(bronchit,
         rfac <- factor(r, labels=c("abs","pres")))
```

```
## ---- getErie ----
Erie <- greatLakes[,"Erie"]
```

```
library(rpart, quietly=TRUE)
```

Figure 3: Yields from 4 packages of land on each of eight sites on the Caribbean island of Antigua. Data are a summarized version of a subset of data given in Andrews and Herzberg 1985, pp.$\tilde{3}$39-353.

Figure 4: Level of Lake Erie, in meters.

Figure 5: Panel A plots Lake Erie levels vs levels at lags 1, 2 and 3 respectively. Panel B shows a consistent pattern of decreasing autocorrelation at successive lags.

Figure 6: GAM smoothing term, fitted to the Lake Erie Data. Most of the autocorrelation structure has been removed, leaving residuals that are very nearly independent.

Figure 7: The plots are from repeated simulations of an AR1 process with a lag 1 correlation of 0.85. Smooth curves, assuming independent errors, have been fitted.

Figure 8: Predictions, 15 years into the future, of lake levels (m). The shaded areas give 80% and 95% confidence bounds.

Figure 9: Estimated contributions of the model terms to `mdbRain`, in a GAM model that is the sum of smooth terms in `Year` and `Rain`. The dashed curves show pointwise 2-SE limits, for the fitted curve. Note the downturn in the trend of `mdbRain` after about 1985,

Figure 10: The top left panel shows the autocorrelation plot of the residuals from the GAM model `mdbRain.gam`. The five remaining panels show autocorrelation plots for a series of independent random normal numbers.

Figure 11: Estimated number of events (aircraft crashes) per time interval versus time. In Panel A, the outcome variable was events per day, while in Panel B it was events per week.

Figure 12: Visual representation of a classification rule, derived using *linear discriminant analysis*, for the forensic glass data. A five-dimensional pattern of separation between the categories has been collapsed down to two dimensions. Some categories may therefore be better distinguished than is evident from this figure.

```
## ---- aupoints ----
aupts <- cmdscale(audists)
```

Figure 13: Decision tree for predicting whether a miner has bronchitis. Where the condition at a node is satisfied, the left branch is taken. Thus, at the initial node, `cig<4.385` takes the branch to the left. In general, unless a random number seed is specified, the tree may be different for each different run of the calculations.

Figure 14: Each tree is for a different bootstrap sample of observations. The final classification is determined by a random vote over all trees. Where there are > 2 explanatory variables (but not here) a different random sample of variables is typically used for each different split. The final classification is determined by a random vote over all trees.

Figure 15: The plot is designed to represent, in two dimensions, the random forest result. It aims to reflect probabilities of group membership given by the analysis. It is not derived by a 'scaling' of the feature space.

Figure 16: Relative locations of Australian cities, derived from road map distances, using metric scaling.

Figure 17: In Panel A, Figure 16 has been rotated and scaled, to give a best fit to a map of Australia. Each city moves as shown by the line that radiates out from it. Panel B is the equivalent plot for the Sammon scaling ordination.