3: Data-Based Generalization

John H Maindonald

June 18, 2018

Ideas and issues illustrated by the graphs in this vignette

A central concern of data analysis is to generalize from results from the one data set that is available for analysis to some wider relevance. Ideas of sample and population are crucial for such generalization. A further important notion is that of a *sampling distribution*,

Mechanisms for assessing predictive accuracy include the use of theory, simulation (which extends the use of theory into areas where the calculations may be intractable), the training/test approach, cross-validation, and bootstrap methods.

Note: Figures 3.15 and 3.16 show results from repeated sampling – simulation or bootstrap sampling. The versions of these figures that are shown in Section 2 are for a substantially reduced number of repeat samples, relative to the text *Statistically Informed Data Mining*.

```
# To include the figures, change `showFigs <- FALSE`
# to `showFigs <- TRUE` in the source `.Rnw` file,
# and regenerate the PDF.
#
showFigs <- FALSE
```

1 R Functions for Creating Chapter 3 Figures

```
fig3.1 <-
function (x=fCatBwt){
lattice::stripplot(jitter(x), pch='|', xlab="Weight (kg)",
                   aspect=0.25, col="black", border="gray")
}
```

```
fig3.2 <-
function(x=fCatBwt){
opar <- par(mfrow=c(2,2), xpd=TRUE,
            mar=c(3.6,3.1,3.6,1.1), mgp=c(2.25, 0.5, 0))
hist(x, labels=TRUE, xlim=c(2, 3),
     xlab="Height (cm)", main="", xpd=TRUE)
title(main="A: Frequency histogram", adj=0, line=1.5, cex.main=1.05)
hist(x, labels=TRUE, xlim=c(2,3),
     xlab="Weight (kg)", main="", freq=FALSE, xpd=TRUE)
title(main="B: Density histogram", adj=0, line=1.5, cex.main=1.05)
par(xpd=FALSE)
hist(x, xlim=c(2,3), xlab="Weight (kg)",
     main="", freq=FALSE)
axis(1)
lines(density(x), xlab="Weight (kg)", col="gray40")
title(main="C: Histogram, density is overlaid", adj=0, line=1.5,
      cex.main=1.05)
plot(density(x), xlim=c(2,3), xlab="Height (cm)",
     main="", sub="", bty="l", type="l")
av <- mean(x)
sdev <- sd(x)
xval <- pretty(c(2,3), n=40)
lines(xval, dnorm(xval, mean=av, sd=sdev), lty=2, col="gray40")
title(main="D: Density curve estimate", adj=0, line=1.5, cex.main=1.05)
par(opar)
par(mfrow=c(1,1))
}
```

```
fig3.3 <-
function (x=fCatBwt, plotit=TRUE){
    av <- mean(x); sdev <- sd(x); sampsize <- length(x)
    simmat <- cbind(x, matrix(rnorm(sampsize*5, mean=av, sd=sdev),
                              ncol=5))
    simdf <- as.data.frame(simmat)
    names(simdf) <- c("Source", paste("normal", 1:5, sep=""))
    simdf <- stack(simdf)
    names(simdf) <- c("height", "Sample")
    denplotSimple <- densityplot(~height, groups=Sample, data=simdf,
                                 xlab="Body weight (kg)")
    denplotn <- update(denplotSimple, scales=list(tck=0.5),
                       main=list(expression(plain("A: Simulation (Densities)")),
                       cex.title=0.9, x=0.05, just="left"),
                       par.settings=simpleTheme(lty=1:6))
    bwpltBasic <- bwplot(Sample ~ height, data=simdf,
```

```
                          xlab="Body weight (kg)",
                          auto.key=list(columns=3))
    bwplotn <- update(bwpltBasic, scales=list(tck=0.5),
                      main=list(expression(plain("B: Simulation (Boxplots)")),
                                cex.title=0.9, x=0.05, just="left"))
    if(plotit){
        print(denplotn, position=c(0,0,0.5,1))
        print(bwplotn, position=c(0.5,0,1,1),newpage=FALSE)
    }
    invisible(list(denplotn, bwplotn))
}
```

```
fig3.4 <-
function (x=fCatBwt, plotit=TRUE)
{
    sampsize <- length(x)
    bootmat <- cbind(x, matrix(0, ncol=5, nrow=sampsize))
    for(i in 2:6) bootmat[,i] <- sample(x, replace=TRUE)
    colnames(bootmat) <- c("Source", paste("normal", 1:5, sep=""))
    bootdf <- stack(as.data.frame(bootmat))
    names(bootdf) <- c("height", "Sample")
    denplotSimple <- densityplot(~ height, groups=Sample, data=bootdf,
                                 xlab="Body weight (kg)")
    legendA <- expression(plain("A: Bootstrap (Densities)"))
    denplot <- update(denplotSimple, scales=list(tck=0.5),
                      main=list(legendA, x=0.05, just="left"), cex.title=0.9,
                      par.settings=simpleTheme(lty=1:6))
    bwpltBasic <- bwplot(Sample ~ height, data=bootdf,
                         xlab="Body weight (kg)",
                         auto.key=list(columns=3))
    legendB <- expression(plain("B: Bootstrap (Boxplots)"))
    bwplot <- update(bwpltBasic, scales=list(tck=0.5),
                     main=list(legendB, x=0.05, just="left"), cex.title=0.9)
    if(plotit){
        print(denplot, position=c(0,0,0.5,1))
        print(bwplot, position=c(0.5,0,1,1),newpage=FALSE)
    }
    invisible(list(denplot, bwplot))
}
```

```
fig3.5 <-
function ()
{
```

```
    opar <- par(mgp=c(2,.75,0), mfrow=c(1,2))
    curve(dnorm(x), from = -3, to = 3,
          ylab=expression("dnorm("*italic(x)*")"),
          xlab=expression("Normal deviate "*italic(x)))
    curve(pnorm(x), from = -3, to = 3,
          ylab=expression("pnorm("*italic(x)*")"),
          xlab=expression("Normal deviate "*italic(x)))
    par(opar)
}
```

```
fig3.6 <-
function (){
    heights <- na.omit(subset(survey, Sex=="Female")$Height)
    plot(density(heights), bty="l", main="",
         cex.axis=1.15, cex.lab=1.15)
    av <- mean(heights); sdev <- sd(heights)
    abline(v=c(av-sdev, av, av+sdev), col="gray", lty=c(2,1,2))
    ## Show fitted normal curve
    xval <- pretty(heights, n=40)
    normal_den <- dnorm(xval, mean=av, sd=sdev)
    lines(xval, normal_den, col="gray40", lty=2)
    ytop <- par()$usr[4]-0.25*par()$cxy[2]
    text(c(av-sdev, av+sdev), ytop,
         labels=c("mean-SD","mean+SD"), col="gray40", xpd=TRUE)
}
```

```
fig3.7 <-
function (wts=fCatBwt){
    opar <- par(pty="s")
    qqnorm(wts)
    par(opar)
}
```

```
fig3.8 <-
function (wts=fCatBwt)
{
    opar <- par(mfrow=c(1,2), mar=c(2.1, 3.6, 3.6,2.6),
                mgp=c(2.25, 0.5,0))
    av <- numeric(1000)
    for (i in 1:1000)
        av[i] <- mean(rnorm(47, mean=2.36, sd=0.27))
    avdens <- density(av)
```

```
    xval <- pretty(c(2.36-3*0.27, 2.36+3*0.27), 50)
    den <- dnorm(xval, mean=2.36, sd=0.27)
    plot(xval, den, type="l", xlab="",  xlim=c(1.5, 3.75),
         ylab="Density", ylim=c(0,max(avdens$y)),
         col="gray", lwd=2, lty=2)
    lines(avdens)
    mtext(side=3, line=0.75, "A: Simulation (from a normal distribution)",
          adj=0)
    legend("bottomright",
           legend=c("Source", "Sampling\ndistribution\nof mean"),
           col=c("gray", "black"), lty=c(2,1), lwd=c(2,1), bty="n",
           y.intersp=0.75, inset=c(0,0.2),
           cex=0.8)
    av <- numeric(1000)
    for (i in 1:1000)
        av[i] <- mean(sample(wts, size=length(wts), replace=TRUE))
    avdens <- density(av)
    plot(density(wts), ylim=c(0, max(avdens$y)),
         xlab="", ylab="Density", xlim=c(1.5, 3.75),
         col="gray", lwd=2, lty=2, main="")
    lines(avdens)
    mtext(side=3, line=0.75,
          "B: Bootstrap samples (resample sample)", adj=0)
    legend("bottomright",
           legend=c("Source",
                    "Sampling\ndistribution\nof mean"),
           col=c("gray", "black"), lty=c(2,1), lwd=c(2,1), bty="n",
           y.intersp=0.75, inset=c(0,0.2),
           cex=0.8)
    par(opar)
    par(mfrow=c(1,1))
}
```

```
fig3.9 <-
function ()
{
    xleft <- 0:3; xrt <- 1:4
    ybot <- rep(0,4); ytop <- rep(1,4) - 0.05
    opar <- par(mar=rep(0.1,4))
    plot(c(0,5), c(-1,4), xlab="", ylab="", axes=F, type="n")
    for(i in 0:3){
        i1 <- i+1
        rect(xleft, ybot+i, xrt, ytop+i)
        xli <- xleft[i+1]; xri <- xrt[i+1];
```

```
        yboti <- (ybot+i)[i+1]; ytopi <- (ytop+i)[i+1]
        rect(xli, yboti, xri, ytopi, col="gray80")
        text(0.5*(xli+xri), 0.5*(yboti+ytopi), "TEST")
        text(0.5*(xleft[-i1]+xrt[-i1]), 0.5*(ybot[-i1]+ytop[-i1])+i, "Training")
        text(4+strwidth("TE"), i+0.475, paste("Fold", i1), adj=0)
}
}
```

```
fig3.10 <-
function (dset=cuckoos, plotit=TRUE)
{
    parset1 <- lattice::simpleTheme(pch=1:6, alpha=0.8)
    plt1 <- lattice::xyplot(length ~ breadth, groups=species, data=dset,
                    par.settings=parset1, aspect=1,
                    scales=list(tck=0.5),
                    auto.key=list(columns=2, alpha=1),
                    main=grid::textGrob("A:", x=unit(.025, "npc"),
                    y = unit(.25, "npc"), just="left",
                    gp=gpar(cex=1))
                    )
    Species <- factor(c(rep("other", 5), "wren")[unclass(cuckoos$species)])
    parset2 <- lattice::simpleTheme(pch=c(0,6), alpha=0.8,
                            col=trellis.par.get()$superpose.symbol$col[c(7,6)])
    plt2 <- lattice::xyplot(length ~ breadth, groups=Species, data=dset,
                    par.settings=parset2,
                    aspect=1, ylab="", scales=list(tck=0.25),
                    auto.key=list(columns=1, alpha=1),
                    main=grid::textGrob("B:", x=unit(.05, "npc"),
                    y = unit(.25, "npc"), just="left",
                    gp=grid::gpar(cex=1))
                    )
    plt2 <- update(plt2,
                    par.settings=list(layout.heights=list(key.top=1.5)))
    if(plotit){
        print(plt1, position=c(0,0,0.515,1))
        print(plt2, position=c(0.485,0,1,1), newpage=FALSE)
    }
    invisible(list(plt1, plt2))
}
```

```
fig3.11 <-
function (dset=cuckoos)
{
```

```
    parset <- list(dot.symbol=list(pch=1, alpha=0.6))
    dotwren <- dotplot(species %in% "wren" ~ length, data=dset,
                       scales=list(y=list(labels=c("Other", "Wren"))),
                       par.settings=parset, xlab="Length (mm)")
    dotwren
}
```

```
fig3.12 <-
function(dset=cuckoos)
{
    avdiff <- numeric(100)
    for(i in 1:100){
        avs <- with(dset, sapply(split(length, species %in% "wren"),
                                 function(x)mean(sample(x, replace=TRUE))))
        avdiff[i] <- avs[1] - avs[2]  # FALSE (non-wren) minus TRUE (wren)
    }
    xtxt <- paste("Means of bootstrap samples of length difference,\n",
                  "non-wren - wren (mm)")
    dotdiff <- dotplot(~ avdiff, xlab=xtxt,
                       par.settings=list(dot.symbol=list(pch=1, alpha=0.6)))
    dotdiff
}
```

```
fig3.13 <-
function (dset=mcats)
{
    xyplot(Hwt ~ Bwt, data=dset,
           type=c("p","r"))
}
```

```
fig3.14 <-
function(dset=mcats)
{
    mcats.lm <- lm(Hwt ~ Bwt, data=dset)
    res <- resid(mcats.lm)
    plot(density(res), main="")
    rug(res, col="gray")
}
```

```
fig3.15 <-
function(dset=mcats, nrepeats=100)
```

```
{
    bootmat <- bootreg(formula = Hwt ~ Bwt,
                       data = dset,
                       nboot = nrepeats)
    bootdf <- as.data.frame(bootmat)
    names(bootdf) <- c("Intercept","Slope")
    colr <- adjustcolor(rep("black",3),
                        alpha.f=0.25)
    if(packageVersion('car') < '3.0.0'){
    scatterplot(Slope ~ Intercept, col=colr,
                data=bootdf, boxplots="xy",
                reg.line=NA, smooth=FALSE)} else
    scatterplot(Slope ~ Intercept, col=colr,
                data=bootdf, boxplots="xy",
                regLine=FALSE, smooth=FALSE)
}
```

```
fig3.16 <-
function (dset=mcats, plotit=TRUE, nrepeats=100)
{
    bootmat <- bootreg(formula = Hwt ~ Bwt,
                       data = dset[-97, ],
                       nboot = nrepeats)
    bootdf0 <- as.data.frame(bootmat)
    names(bootdf0) <- c("Intercept","Slope")
    gphA <- xyplot(Slope ~ Intercept, data=bootdf0, alpha=0.25,
                   main=paste("A:", nrepeats, "bootstrap samples"),
                   cex.title=1.1)
        simmat <- simreg(formula = Hwt ~ Bwt,
                         data=dset[-97, ], nsim=nrepeats)
    simdf <- as.data.frame(simmat)
    names(simdf) <- c("Intercept","Slope")
    gphB <- xyplot(Slope ~ Intercept, data=simdf, alpha=0.25,
                   main=paste("B:", nrepeats, "simulations"),
                   cex.title=1.1)
    if(plotit){
        print(gphA, position=c(0,0,0.515,1))
        print(gphB, position=c(0.485,0,1,1), newpage=FALSE)
    }
    invisible(list(gphA, gphB))
}
```

2 Show the Figures

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

```
msg <- "Data object 'cats', from 'MASS', may not be available"
if(!requireNamespace("MASS"))print(msg) else {
  mcats <- subset(MASS::cats, Sex=="M")
  fcats <- subset(MASS::cats, Sex=="F")
  fCatBwt <- na.omit(fcats[, "Bwt"])
}
if(!exists("cuckoos")){
        msg <- "Cannot find either 'cuckoos' or 'DAAG::cuckoos',"
if(requireNamespace("DAAG"))cuckoos <- DAAG::cuckoos else
          print(msg)
      }
```

```
if(exists('fCatBwt'))fig3.1() else
    print("Object 'fCatBwt' is not available; get from 'MASS::cats'")
```

```
if(exists('fCatBwt'))fig3.2() else
    print("Object 'fCatBwt' is not available; get from 'MASS::cats'")
```

```
if(exists('fCatBwt'))fig3.3() else
    print("Object 'fCatBwt' is not available; get from 'MASS::cats'")
```

```
if(exists('fCatBwt'))fig3.4() else
    print("Object 'fCatBwt' is not available; get from 'MASS::cats'")
```

```
fig3.5()
```

```
fig3.6()
```

```
if(exists("fCatBwt"))fig3.7() else
    print("Object 'fCatBwt' was not found; get from 'MASS::cats'")
```

```
if(exists("fCatBwt"))fig3.8() else
    print("Object 'fCatBwt' was not found; get from 'MASS::cats'")
```

```
fig3.9()
```

```
if(exists("cuckoos"))fig3.10() else
    print("Object 'cuckoos' was not found; get 'DAAG::cuckoos'")
```

```
if(exists("cuckoos"))fig3.11() else
    print("Object 'cuckoos' was not found; get 'DAAG::cuckoos'")
```

```
if(exists("cuckoos"))fig3.12() else
    print("Object 'cuckoos' was not found; get 'DAAG::cuckoos'")
```

```
if(exists('mcats'))fig3.13() else
    print("Object 'mcats' was not found; subset from 'MASS::cats'")
```

```
if(exists('mcats'))fig3.14() else
    print("Object 'mcats' was not found; subset from 'MASS::cats'")
```

```
if(!require(car)){
    print("Figure 3.15 requires the 'car' package")
    return("The 'car' package needs to be installed.")
}
if(exists("mcats"))fig3.15(nrepeats=100) else
    print("Object 'mcats' was not found; subset from 'MASS::cats'")
```

```
if(exists('mcats'))fig3.16() else
    print("Object 'mcats' was not found; subset from 'MASS::cats'")
```