R Code for ‘A Practical Guide . . .’

John Maindonald

2024-06-02

Table of contents

# Preface

Code provided here is for the May 2024 text:

“A Practical Guide to Data Analysis Using R – An Example-Based Approach”, by John H Maindonald, W John Braun, and Jeffrey L Andrews.

Code that is shown in the text is filled out to include all code for graphs. In chapter 2 and later, the text includes code only for those graphs that are specifically targeted at the methodology under discussion.

The text has been available in digital format since early May 2024. It has been available in print in the UK from May 30, is due to be available in print in the USA in July, and in Australia and New Zealand in August 2024.

For details for the UK, see: [A Practical Guide to Data Analysis Using R – Cambridge University Press UK](https://www.cambridge.org/gb/universitypress/subjects/statistics-probability/computational-statistics-machine-learning-and-information-sc/practical-guide-data-analysis-using-r-example-based-approach)

# 1. Chapter 1: Learning from data

library(knitr)  
opts\_chunk[['set']](fig.width=6, fig.height=6, comment=" ",  
 out.width="80%", fig.align="center", fig.show='hold',  
 size="small", ps=10, strip.white = TRUE,   
 tidy.opts = list(replace.assign=FALSE))

## xtras=TRUE ## Set to TRUE to execute code 'extras'  
xtras <- FALSE  
library(knitr)  
## opts\_chunk[['set']](results="asis")  
## opts\_chunk[['set']](eval=FALSE) ## Set to TRUE to execute main part of code  
opts\_chunk[['set']](eval=FALSE)

##### Packages required (plus any dependencies)

latticeExtra (lattice is a dependency); DAAG; car; MASS; AICcmodavg; BayesFactor; boot; MPV; ggplot2; tidyr

Additionally, knitr and Hmisc are required in order to process the Rmd source file. The prettydoc package is by default used to format the html output.

##### Chapter summary

##### A note on terminology — variables, factors, and more!

### Section 1.1: Questions, and data that may point to answers

#### Subsection 1.1.1: A sample is a window into the wider population

## For the sequence below, precede with set.seed(3676)  
set.seed(3696)  
sample(1:9384, 12, replace=FALSE) # NB: `replace=FALSE` is the default

chosen1200 <- sample(1:19384, 1200, replace=FALSE)

## For the sequence below, precede with set.seed(366)  
set.seed(366)  
split(sample(seq(1:10)), rep(c("Control","Treatment"), 5))  
# sample(1:10) gives a random re-arrangement (permutation) of 1, 2, ..., 10

##### Cluster sampling

##### \*A note on with-replacement samples

sample(1:10, replace=TRUE)  
## sample(1:10, replace=FALSE) returns a random permutation of 1,2,...10

#### Subsection 1.1.2: Formulating the scientific question

##### Example: a question about cuckoo eggs

library(latticeExtra) # Lattice package will be loaded and attached also  
cuckoos <- DAAG::cuckoos  
## Panel A: Dotplot without species means added  
dotplot(species ~ length, data=cuckoos) ## `species ~ length` is a 'formula'  
## Panel B: Box and whisker plot  
bwplot(species ~ length, data=cuckoos)  
## The following shows Panel A, including species means & other tweaks  
av <- with(cuckoos, aggregate(length, list(species=species), FUN=mean))  
dotplot(species ~ length, data=cuckoos, alpha=0.4, xlab="Length of egg (mm)") +  
 as.layer(dotplot(species ~ x, pch=3, cex=1.4, col="black", data=av))  
 # Use `+` to indicate that more (another 'layer') is to be added.  
 # With `alpha=0.4`, 40% is the point color with 60% background color  
 # `pch=3`: Plot character 3 is '+'; `cex=1.4`: Default char size X 1.4

## Code  
suppressPackageStartupMessages(library(latticeExtra, quietly=TRUE))  
cuckoos <- DAAG::cuckoos  
## For tidier labels replace ".", in several of the names, by a space  
specnam <- with(cuckoos, sub(pattern=".", replacement=" ",   
 levels(species), fixed=TRUE))  
# fixed=TRUE: "interpret "." as ".", not as a 'any single character'"  
cuckoos <- within(cuckoos, levels(species) <- specnam)  
## Panel A: Dotplot: data frame cuckoos (DAAG)  
av <- with(cuckoos, aggregate(length, list(species=species), FUN=mean))  
gphA <- dotplot(species ~ length, data=cuckoos, alpha=0.4) +  
 as.layer(dotplot(species ~ x, pch=3, cex=1.4, col="black", data=av))  
# alpha sets opacity. With alpha=0.4, 60% of the background shows through  
# Enter `print(plt1)` or `plot(plt1)` or simply `plt1` to display the graph  
## Panel B: Box plot  
gphB <- bwplot(species ~ length, data=cuckoos)  
update(c("A: Dotplot"=gphA, "B: Boxplot"=gphB), between=list(x=0.4),  
 xlab="Length of egg (mm)")   
## latticeExtra::c() joins compatible plots together.   
## See `?latticeExtra::c`

#### Subsection 1.1.3: Planning for a statistical analysis

##### Understand the data

##### Causal inference

##### What was measured? Is it the relevant measure?

##### Use relevant prior information in the planning stages

##### Subject area knowledge and judgments

##### The importance of clear communication

##### Data-based selection of comparisons

##### Models must be fit for their intended use

#### Subsection 1.1.4: Results that withstand thorough and informed challenge

#### Subsection 1.1.5: Using graphs to make sense of data

##### Graphical comparisons

#### Subsection 1.1.6: Formal model-based comparison

options(width=70)  
cuckoos <- DAAG::cuckoos  
av <- with(cuckoos, aggregate(length, list(species=species), FUN=mean))  
setNames(round(av[["x"]],2), abbreviate(av[["species"]],10))

with(cuckoos, scale(length[species=="wren"], scale=FALSE))[,1]

### Section 1.2: Graphical tools for data exploration

#### Subsection 1.2.1: Displays of a single variable

library(latticeExtra, quietly=TRUE)  
fossum <- subset(DAAG::possum, sex=="f")  
femlen <- DAAG::bounce(fossum[["totlngth"]], d=0.1)  
## Panel A  
yaxpos <- c(0,5,10,15,20)/(5\*nrow(fossum))  
z <- boxplot(list(val = femlen), plot = FALSE)  
gph1 <- bwplot(~femlen, ylim=c(0.55,2.75), xlim=c(70,100),   
 scales=list(y=list(draw=FALSE)))+  
 latticeExtra::layer(panel.rug(x,pch="|"))  
legstat <- data.frame(x=c(z$out,z$stats), y=c(1.08, rep(1.3,5)),  
 tx=c("Outlier?", "Smallest value", "lower quartile", "median",   
 "upper quartile", "Largest value"),   
 tx2= c("", "(outliers excepted)",rep("",3), "(no outliers)"))  
gphA <- gph1+latticeExtra::layer(data=legstat,  
 panel.text(x=x,y=y,labels=tx,adj=c(0,0.4),srt=90, cex=0.85),  
 panel.text(x=x[c(2,6)]+0.75,y=c(1.125,1.38),labels=tx2[c(2,6)],  
 adj=c(0,0.4),srt=90, cex=0.85))  
## Panel B  
gph2 <- densityplot(~femlen, ylim=c(0,0.108), xlim=c(70,100),   
 plot.points=TRUE, pch="|",cex=1.75, ylab=c(""," Density"))  
gph3 <- histogram(~femlen, ylim=c(0,0.108), type="density",   
 scales=list(y=list(at=yaxpos, labels=c(0,5,10,15,20), col="gray40")),   
 alpha=0.5, ylab="", breaks=c(75,80,85,90,95,100),   
 col='transparent',border='gray40')  
gph4 <- doubleYScale(gph2, gph3, use.style=FALSE, add.ylab2=FALSE)  
gphB <- update(gph4, par.settings=list(fontsize = list(text=10, points=5)),  
 scales=list(tck=c(0.5,0.5)))  
update(c("B: Density curve, with histogram overlaid"=gphB,   
 "A: Boxplot, with annotation added"=gphA, layout=c(1,2), y.same=F),   
 as.table=TRUE, between=list(y=1.4),   
 xlab="Total length of female possums (cm)")

fossum <- subset(DAAG::possum, sex=="f")  
densityplot(~totlngth, plot.points=TRUE, pch="|", data=fossum) +  
 layer\_(panel.histogram(x, type="density", breaks=c(75,80,85,90,95,100)))

##### Comparing univariate displays across factor levels

library(latticeExtra, quietly=TRUE)  
fossum <- subset(DAAG::possum, sex=="f")  
femlen <- DAAG::bounce(fossum[["totlngth"]], d=0.1)  
## Panel A  
yaxpos <- c(0,5,10,15,20)/(5\*nrow(fossum))  
z <- boxplot(list(val = femlen), plot = FALSE)  
gph1 <- bwplot(~femlen, ylim=c(0.55,2.75), xlim=c(70,100),   
 scales=list(y=list(draw=FALSE)))+  
 latticeExtra::layer(panel.rug(x,pch="|"))  
legstat <- data.frame(x=c(z$out,z$stats), y=c(1.08, rep(1.3,5)),  
 tx=c("Outlier?", "Smallest value", "lower quartile", "median",   
 "upper quartile", "Largest value"),   
 tx2= c("", "(outliers excepted)",rep("",3), "(no outliers)"))  
gphA <- gph1+latticeExtra::layer(data=legstat,  
 panel.text(x=x,y=y,labels=tx,adj=c(0,0.4),srt=90, cex=0.85),  
 panel.text(x=x[c(2,6)]+0.75,y=c(1.125,1.38),labels=tx2[c(2,6)],  
 adj=c(0,0.4),srt=90, cex=0.85))  
## Panel B  
gph2 <- densityplot(~femlen, ylim=c(0,0.108), xlim=c(70,100),   
 plot.points=TRUE, pch="|",cex=1.75, ylab=c(""," Density"))  
gph3 <- histogram(~femlen, ylim=c(0,0.108), type="density",   
 scales=list(y=list(at=yaxpos, labels=c(0,5,10,15,20), col="gray40")),   
 alpha=0.5, ylab="", breaks=c(75,80,85,90,95,100),   
 col='transparent',border='gray40')  
gph4 <- doubleYScale(gph2, gph3, use.style=FALSE, add.ylab2=FALSE)  
gphB <- update(gph4, par.settings=list(fontsize = list(text=10, points=5)),  
 scales=list(tck=c(0.5,0.5)))  
update(c("B: Density curve, with histogram overlaid"=gphB,   
 "A: Boxplot, with annotation added"=gphA, layout=c(1,2), y.same=F),   
 as.table=TRUE, between=list(y=1.4),   
 xlab="Total length of female possums (cm)")

## Create boxplot graph object --- Simplified code  
gph <- bwplot(Pop~totlngth | sex, data=possum)   
## plot graph, with dotplot distribution of points below boxplots  
gph + latticeExtra::layer(panel.dotplot(x, unclass(y)-0.4))

#### Subsection 1.2.2: Patterns in univariate time series

layout(matrix(c(1,2)), heights=c(2.6,1.75))  
measles <- DAAG::measles  
## Panel A:  
par(mgp=c(2.0,0.5,0))  
plot(log10(measles), xlab="", ylim=log10 (c(1,5000\*540)),  
 ylab=" Deaths", yaxt="n", fg="gray", adj=0.16)  
londonpop <-ts(c(1088, 1258, 1504, 1778, 2073, 2491, 2921, 3336, 3881,  
 4266, 4563, 4541, 4498, 4408), start=1801, end=1931, deltat=10)  
points(log10(londonpop\*500), pch=16, cex=.5)  
ytiks1 <- c(1, 10, 100, 1000)  
axis(2, at=log10(ytiks1), labels=paste(ytiks1), lwd=0, lwd.ticks=1)  
abline(h=log10(ytiks1), col = "lightgray", lwd=2)  
par(mgp=c(-2,-0.5,0))  
ytiks2 <- c(1000000, 5000000) ## London population in thousands  
abline(h=log10(ytiks2\*0.5), col = "lightgray", lwd=1.5)  
abline(v=seq(from=1650,to=1950,by=50), col = "lightgray", lwd = 1.5)  
mtext(side=2, line=0.5, "Population", adj=1, cex=1.15, las=3)  
axis(2, at=log10(ytiks2\*0.6), labels=paste(ytiks2), tcl=0.3,  
 hadj=0, lwd=0, lwd.ticks=1)  
mtext(side=3, line=0.3, "A (1629-1939)", adj=0, cex=1.15)  
##  
## Panel B: window from 1840 to 1882  
par(mgp=c(2.0,0.5,0))  
plot(window(measles, start=1840, end=1882), xlab="",  
ylab="Deaths Pop (1000s)", ylim=c(0, 4200), fg="gray")  
points(window(londonpop, start=1840, end=1882), pch=16, cex=0.5)  
mtext(side=3, line=0.5, "B (1841-1881)", adj=0, cex=1.15)

#### Subsection 1.2.3: Visualizing relationships between pairs of variables

#### Subsection 1.2.4: Response lines (and/or curves)

par(pty="s")  
plot(distance.traveled ~ starting.point, data=DAAG::modelcars, fg="gray",  
xlim=c(0,12.5), xaxs="i", xlab = "Distance up ramp (cm)",  
ylab="Distance traveled (cm)")

#### Subsection 1.2.5: Multiple variables and times

## Apply function range to columns of data frame jobs (DAAG)  
sapply(DAAG::jobs, range) ## NB: `BC` = British Columbia

## Panel A: Basic plot; all series in a single panel; use log y-scale  
formRegions <- Ontario+Quebec+BC+Alberta+Prairies+Atlantic ~ Date  
basicGphA <-  
 xyplot(formRegions, outer=FALSE, data=DAAG::jobs, type="l", xlab="",   
 ylab="Number of workers", scales=list(y=list(log="e")),  
 auto.key=list(space="right", lines=TRUE, points=FALSE))  
 ## `outer=FALSE`: plot all columns in one panel  
## Panel B: Separate panels (`outer=TRUE`); sliced log scale  
basicGphB <-  
 xyplot(formRegions, data=DAAG::jobs, outer=TRUE, type="l", layout=c(3,2),   
 xlab="", ylab="Number of workers",  
 scales=list(y=list(relation="sliced", log=TRUE)))  
# Provinces are in order of number of workers in Dec96  
## Create improved x- and y-axis tick labels; will update to use  
datelabpos <- seq(from=95, by=0.5, length=5)  
datelabs <- format(seq(from=as.Date("1Jan1995", format="%d%b%Y"),  
 by="6 month", length=5), "%b%y")  
## Now create $y$-labels that have numbers, with log values underneath  
ylabposA <- exp(pretty(log(unlist(DAAG::jobs[,-7])), 5))  
ylabelsA <- paste(round(ylabposA),"\n(", log(ylabposA), ")", sep="")  
## Repeat, now with 100 ticks, to cover all 6 slices of the scale  
ylabposB <- exp(pretty(log(unlist(DAAG::jobs[,-7])), 100))  
ylabelsB <- paste(round(ylabposB),"\n(", log(ylabposB), ")", sep="")  
gphA <- update(basicGphA, scales=list(x=list(at=datelabpos, labels=datelabs),  
 y=list(at=ylabposA, labels=ylabelsA)))  
gphB <- update(basicGphB, xlab="", between=list(x=0.25, y=0.25),  
 scales=list(x=list(at=datelabpos, labels=datelabs),  
 y=list(at=ylabposB, labels=ylabelsB)))  
layout.list <- list(layout.heights=list(top.padding=0,  
 bottom.padding=0, sub=0, xlab=0),   
 fontsize=list(text=8, points=5))  
jobstheme <- modifyList(ggplot2like(pch=1, lty=c(4:6,1:3),  
 col.line='black', cex=0.75),layout.list)  
print(update(gphA, par.settings=jobstheme, axis=axis.grid,  
 main=list("A: Same vertical log scale",y=0)),  
 position=c(0.1,0.615,0.9,1), newpage=TRUE)  
print(update(gphB, par.settings=jobstheme, axis=axis.grid,  
 main=list("B: Sliced vertical log scale",y=0)),  
 position=c(0,0,1,0.625), newpage=FALSE)

plot(c(1230,1860), c(0, 10.5), axes=FALSE, bty="n",  
 xlab="", ylab="", type="n", log="x")  
xpoints <- c(1366, 1436, 1752, 1840)  
axis(1, at=xpoints, labels=FALSE, tck=0.01, lty=1, lwd=0, lwd.ticks=1)  
for(i in 1:4){  
 axis(1, at=xpoints[i],  
 labels=substitute(italic(a), list(a=paste(xpoints[i]))),  
 line=-2.25, lty=0, cex=0.8, lwd=0, lwd.ticks=1)  
 lines(rep(xpoints[i],2), c(0, 0.15\*par()[["cxy"]][2]), lty=1)  
}  
axpos <- 1250\*cumprod(c(1, rep(1.2,2)))  
axis(1, at=c(axpos,1840), labels=F, lwd.ticks=0)  
lab <- round(axpos)  
axis(1, at=axpos, labels=lab)  
lab2 <- lapply(round(log2(xpoints),3), function(x)substitute(2^a, list(a=x)))  
axis(1, at=xpoints, labels=as.expression(lab2), line=-3.5, lwd=0)  
labe <- lapply(format(round(log(xpoints),3)), function(x)substitute(e^a, list(a=x)))  
axis(1, at=xpoints, labels=as.expression(labe), line=-5, lwd=0)  
lab10 <- lapply(round(log10(xpoints),3), function(x)substitute(10^a, list(a=x)))  
axis(1, at=xpoints, labels=as.expression(lab10), line=-6.5, lwd=0)  
par(family="mono", xpd=TRUE)  
axis(1, at=1220, labels="log=2", line=-3.5, hadj=0, lwd=0)  
axis(1, at=1220, labels='log="e"', line=-5, hadj=0, lwd=0)  
axis(1, at=1220, labels="log=10", line=-6.5, hadj=0, lwd=0)   
wid2 <- strwidth("log=2")  
par(family="sans")

#### Subsection 1.2.6: \*Labeling technicalities

#### Subsection 1.2.7: Graphical displays for categorical data

stones <- array(c(81,6,234,36,192,71,55,25), dim=c(2,2,2),  
 dimnames=list(Success=c("yes","no"),  
 Method=c("open","ultrasound"), Size=c("<2cm", ">=2cm")))  
margin12 <- margin.table(stones, margin=1:2)

byMethod <- 100\*prop.table(margin12, margin=2)  
pcGood <- 100\*prop.table(stones, margin=2:3)["yes", , ]  
dimnam <- dimnames(stones)  
numOps <- margin.table(stones, margin=2:3)  
opStats <- data.frame(Good=c(pcGood[1,],pcGood[2,]),  
 numOps=c(numOps[1,], numOps[2,]),  
 opType=factor(rep(dimnam[["Method"]],c(2,2))),  
 Size=rep(dimnam[["Size"]],2))  
xlim <- range(opStats$Good)\*c(0.65,1.015)  
ylim <- c(0, max(opStats$numOps)\*1.15)  
plot(numOps~Good, data=opStats, type="h", lwd=4, xlim=xlim, ylim=ylim,  
 fg="gray",col=rep(c("blue","red"),rep(2,2)),  
 xlab="Success rate (%)", ylab="Number of operations")  
# with(opStats, text(numOps~Good, labels=Size,  
# col=rep(c('blue','red'),rep(2,2)),  
# offset=0.25,pos=3, cex=0.75))  
labpos <- lapply(split(opStats, opStats$Size),   
 function(x)apply(x[,1:2],2,function(z)c(z[1],mean(z),z[2])))  
sizeNam <- names(labpos)  
lapply(labpos, function(x)lines(x[,'Good'],x[,'numOps']+c(0,35,0),  
 type="c",col="gray"))  
txtmid <- sapply(labpos, function(x)c(x[2,'Good'],x[2,'numOps']+35))  
text(txtmid[1,]+c(-1.4,0.85),txtmid[2,],labels=sizeNam,col="gray40",  
 pos=c(4,2), offset=0)  
par(xpd=TRUE)  
text(byMethod[1,1:2],rep(par()$usr[4],2)+0.5\*strheight("^"), labels=c("^","^"),  
 col=c("blue","red"),cex=1.2,srt=180)  
text(byMethod[1,], par()$usr[4]+1.4\*strheight("A"),  
 labels=paste(round(byMethod[1,],1)),cex=0.85)  
text(byMethod[1,1:2]+c(3.5,-3.5), rep(par()$usr[4],2)+2.65\*strheight("A"),  
labels=c("All open","All ultrasound"), pos=c(2,4))  
par(xpd=FALSE)  
abline(h=100\*(0:2),col="lightgray",lwd=0.5)  
abline(v=10\*(5:9),col="lightgray",lwd=0.5)  
legend("topleft", col=c('blue','red'),lty=c(1,1), lwd=1, cex=0.9,  
 y.intersp=0.75, legend=c("Open","Ultrasound"),bty="n",  
 inset=c(-0.01,-0.01))

#### Subsection 1.2.8: What to look for in plots

##### Outliers

##### Asymmetry of the distribution

##### Changes in variability

##### Clustering

##### Nonlinearity

##### Time trends in the data

### Section 1.3: Data Summary

#### Subsection 1.3.1: Counts

## Table of counts example: data frame nswpsid1 (DAAG)  
## Specify `useNA="ifany"` to ensure that any NAs are tabulated  
tab <- with(DAAG::nswpsid1, table(trt, nodeg, useNA="ifany"))  
dimnames(tab) <- list(trt=c("none", "training"), educ = c("completed", "dropout"))  
tab

##### Tabulation that accounts for frequencies or weights – the xtabs() function

gph <- lattice::bwplot(log(nassCDS$weight+1), xlab="Inverse sampling weights",  
 scales=list(x=list(at=c(0,log(c(10^(0:5)+1))), labels=c(0,10^(0:5)))))  
update(gph, par.settings=DAAG::DAAGtheme(color=F, col.points='gray50'))

sampNum <- table(nassCDS$dead)  
popNum <- as.vector(xtabs(weight ~ dead, data=nassCDS))  
rbind(Sample=sampNum, "Total number"=round(popNum,1))

nassCDS <- DAAG::nassCDS  
Atab <- xtabs(weight ~ airbag + dead, data=nassCDS)/1000  
## Define a function that calculates Deaths per 1000  
DeadPer1000 <- function(x)1000\*x[2]/sum(x)  
Atabm <- ftable(addmargins(Atab, margin=2, FUN=DeadPer1000))  
print(Atabm, digits=2, method="compact", big.mark=",")

SAtab <- xtabs(weight ~ seatbelt + airbag + dead, data=nassCDS)  
## SAtab <- addmargins(SAtab, margin=3, FUN=list(Total=sum)) ## Gdet Totals  
SAtabf <- ftable(addmargins(SAtab, margin=3, FUN=DeadPer1000), col.vars=3)  
print(SAtabf, digits=2, method="compact", big.mark=",")

FSAtab <- xtabs(weight ~ dvcat + seatbelt + airbag + dead, data=nassCDS)  
FSAtabf <- ftable(addmargins(FSAtab, margin=4, FUN=DeadPer1000), col.vars=3:4)  
print(FSAtabf, digits=1)

#### Subsection 1.3.2: Summaries of information from data frames

## Individual vine yields, with means by block and treatment overlaid  
kiwishade <- DAAG::kiwishade  
kiwishade$block <- factor(kiwishade$block, levels=c("west","north","east"))  
keyset <- list(space="top", columns=2,  
text=list(c("Individual vine yields", "Plot means (4 vines)")),  
points=list(pch=c(1,3), cex=c(1,1.35), col=c("gray40","black")))  
panelfun <- function(x,y,...){panel.dotplot(x,y, pch=1, ...)  
av <- sapply(split(x,y),mean); ypos <- unique(y)  
lpoints(ypos~av, pch=3, col="black")}  
dotplot(shade~yield | block, data=kiwishade, col="gray40", aspect=0.65,  
 panel=panelfun, key=keyset, layout=c(3,1))  
## Note that parameter settings were given both in the calls  
## to the panel functions and in the list supplied to key.

## mean yield by block by shade: data frame kiwishade (DAAG)  
kiwimeans <- with(DAAG::kiwishade,   
 aggregate(yield, by=list(block, shade), mean))  
names(kiwimeans) <- c("block","shade","meanyield")  
head(kiwimeans, 4) # First 4 rows

##### The benefits of data summary – dengue status example

#### Subsection 1.3.3: Measures of variation

##### Cuckoo eggs example

options(width=72)  
## SD of length, by species: data frame cuckoos (DAAG)  
z <- with(cuckoos, sapply(split(length,species), function(x)c(sd(x),length(x))))  
print(setNames(paste0(round(z[1,],2)," (",z[2,],")"),  
 abbreviate(colnames(z),11)), quote=FALSE)

##### Degrees of freedom

#### Subsection 1.3.4: Inter-quartile range (IQR) and median absolute deviation (MAD)

#### Subsection 1.3.5: A pooled standard deviation estimate

##### Elastic bands example

<>= sapply(DAAG::two65, function(x) c(Mean=mean(x), sd=sd(x))) |> round(2) @

#### Subsection 1.3.6: Effect size

setNames(diff(c(ambient=244.1, heated=253.5))/c(sd=10.91), "Effect size")

Data are available in the data frame DAAG::two65.

vignette('effectsize', package='effectsize')

#### Subsection 1.3.7: Correlation

set.seed(17)  
x1 <- x2 <- x3 <- (11:30)/5  
y1 <- x1 + rnorm(20, sd=0.5)  
y2 <- 2 - 0.05 \* x1 + 0.1 \* ((x1 - 1.75))^4 + rnorm(20, sd=1.5)  
y3 <- (x1 - 3.85)^2 + 0.015 + rnorm(20)  
theta <- ((2 \* pi) \* (1:20))/20  
x4 <- 10 + 4 \* cos(theta)  
y4 <- 10 + 4 \* sin(theta) + rnorm(20, sd=0.6)  
xy <- data.frame(x = c(rep(x1, 3), x4), y = c(y1, y2, y3, y4),  
 gp = factor(rep(1:4, rep(20, 4))))  
xysplit <- split(xy, xy$gp)  
rho <- sapply(xysplit, function(z)with(z,cor(x,y, method=c("pearson"))))  
rhoS <- sapply(xysplit, function(z)with(z,cor(x,y, method=c("spearman"))))  
rnam <- as.list(setNames(round(c(rho,rhoS),2), paste0("r",1:8)))  
striplabs <- bquote(expression(paste(r==.(r1), " ",r[s]==.(r5)),   
 paste(r==.(r2), " ",r[s]==.(r6)),  
 paste(r==.(r3), " ",r[s]==.(r7)),   
 paste(r==.(r4), " ",r[s]==.(r8))), rnam)  
xyplot(y ~ x | gp, data=xy, layout=c(4,1), xlab="", ylab="",   
 strip=strip.custom(factor.levels=striplabs), aspect=1,  
 scales=list(relation='free', draw=FALSE), between=list(x=0.5,y=0)  
)

### Section 1.4: Distributions: quantifying uncertainty

#### Subsection 1.4.1: Discrete distributions

## dbinom(0:10, size=10, prob=0.15)  
setNames(round(dbinom(0:10, size=10, prob=0.15), 3), 0:10)

pbinom(q=4, size=10, prob=0.15)

qbinom(p = 0.70, size = 10, prob = 0.15)  
## Check that this lies between the two cumulative probabilities:  
## pbinom(q = 1:2, size=10, prob=0.15)

rbinom(15, size=4, p=0.5)

## dpois(x = 0:8, lambda = 3)  
setNames(round(dpois(x = 0:8, lambda = 3),4), 0:8)  
## Probability of > 8 raisins  
## 1-ppois(q = 8, lambda = 3) ## Or, ppois(q=8, lambda=3, lower.tail=FALSE)

1 - ppois(q = 8, lambda = 3)  
ppois(q=8, lambda=3, lower.tail=FALSE) ## Alternative  
1-sum(dpois(x = 0:8, lambda = 3)) ## Another alternative

raisins <- rpois(20, 3)  
raisins

##### Initializing the random number generator

set.seed(23286) # Use to reproduce the sample below  
rbinom(15, size=1, p=0.5)

##### Means and variances

#### Subsection 1.4.2: Continuous distributions

z <- seq(-3,3,length=101)  
plot(z, dnorm(z), type="l", ylab="Normal density",  
 yaxs="i", bty="L", tcl=-0.3, fg="gray",  
 xlab="Distance, in SDs, from mean", cex.lab=0.9)  
polygon(c(z[z <= 1.0],1.0),c(dnorm(z[z <= 1.0]), dnorm(-3)), col="grey")  
chh <- par()$cxy[2]  
arrows(-1.8, 0.32, -0.25, 0.2, length=0.07, xpd=T)  
cump <- round(pnorm(1), 3)  
text(-1.8, 0.32+0.75\*chh, paste("pnorm(1)\n", "=", cump), xpd=T, cex=0.8)

pnormExs <- c('pnorm(0)', 'pnorm(1)', 'pnorm(-1.96)', 'pnorm(1.96)',  
'pnorm(1.96, mean=2)', 'pnorm(1.96, sd=2)')  
Prob <- sapply(pnormExs, function(x)eval(parse(text=x)))  
df <- as.data.frame(Prob)  
df$Prob <- round(df$Prob,3)  
print(df)

## Plot the normal density, in the range -3 to 3  
z <- pretty(c(-3,3), 30) # Find ~30 equally spaced points  
ht <- dnorm(z) # Equivalent to dnorm(z, mean=0, sd=1)  
plot(z, ht, type="l", xlab="Normal variate", ylab="Density", yaxs="i")  
# yaxs="i" locates the axes at the limits of the data

qnorm(.9) # 90th percentile; mean=0 and SD=1

## Additional examples:  
setNames(qnorm(c(.5,.841,.975)), nm=c(.5,.841,.975))  
qnorm(c(.1,.2,.3)) # -1.282 -0.842 -0.524 (10th, 20th and 30th percentiles)  
qnorm(.1, mean=100, sd=10) # 87.2 (10th percentile, mean=100, SD=10)

##### Different ways to represent distributions

##### Generating simulated samples from the normal and other continuous distributions

options(digits=2) # Suggest number of digits to display  
rnorm(10) # 10 random values from the normal distribution

mu <- 10  
sigma <- 1  
n <- 1  
m <- 50  
four <- 4  
nrep <- 5  
seed <- 21  
totrows <- 1  
if(is.null(totrows))  
totrows <- floor(sqrt(nrep))  
totcols <- ceiling(nrep/totrows)  
z <- range(pretty(mu + (c(-3.4, 3.4) \* sigma), 50))  
xy <- data.frame(x=rep(0,nrep),y=rep(0,nrep),n=rep(n,nrep),  
 mm=rep(m,nrep),four=rep(four,nrep))  
fac <- factor(paste("Simulation", 1:nrep),  
 lev <- paste("Simulation", 1:nrep))  
xlim<-z  
## ylim<-c(0,dnorm(0)\*sqrt(n))  
ylim <- c(0,1)  
xy <- split(xy,fac)  
xy<-lapply(1:length(xy),function(i){c(as.list(xy[[i]]), list(xlim=xlim,  
 ylim=ylim))})  
panel.mean <- function(data, mu = 10, sigma = 1, n2 = 1,  
 mm = 100, nrows, ncols, ...)  
{  
 vline <- function(x, y, lty = 1, col = 1)  
 lines(c(x, x), c(0, y), lty = lty, col = col)  
 n2<-data$n[1]  
 mm<-data$mm[1]  
 our<-data$four[1] ## Four characters in each unit interval of x  
 nmid <- round(four \* 4)  
 nn <- array(0, 2 \* nmid + 1)  
 #########################################  
 z <- mu+seq(from=-3.4\*sigma, to=3.4\*sigma, length=mm)  
 atx<-pretty(z)  
 qz <- pnorm((z - mu)/sigma)  
 dz <- dnorm((z - mu)/sigma)  
 chw <- sigma/four  
 chh <- strheight("O")\*0.75  
 htfac <- (mm \* chh)/four  
 if(nrows==1&&ncols==1)  
 lines(z, dz \* htfac)  
 if(nrows==1)axis(1,at=atx, lwd=0, lwd.ticks=1)  
 y <- rnorm(mm, mu, sigma/sqrt(n2))  
 pos <- round((y - mu)/sigma \* four)  
 for(i in 1:mm) {  
 nn[nmid + pos[i]] <- nn[nmid + pos[i]] + 1  
 xpos <- chw \* pos[i]  
 text(mu + xpos, nn[nmid + pos[i]] \* chh - chh/4, "x")  
 }  
}  
DAAG::panelplot(xy,panel=panel.mean,totrows=totrows,totcols=totcols,  
 oma=c(1.5, 0, rep(0.5,2)), fg='gray')

## The following gives a conventional histogram representations:  
set.seed (21) # Use to reproduce the data in the figure  
df <- data.frame(x=rnorm(250), gp=rep(1:5, rep(50,5)))  
lattice::histogram(~x|gp, data=df, layout=c(5,1))

runif(n = 20, min=0, max=1) # 20 numbers, uniform distn on (0, 1)  
rexp(n=10, rate=3) # 10 numbers, exponential, mean 1/3.

#### Subsection 1.4.3: Graphical checks for normality

tab <- t(as.matrix(DAAG::pair65))  
rbind(tab,"heated-ambient"=tab[1,]-tab[2,])

## Normal quantile-quantile plot for heated-ambient differences,  
## compared with plots for random normal samples of the same size  
plt <- with(DAAG::pair65, DAAG::qreference(heated-ambient, nrep=10, nrows=2))  
update(plt, scales=list(tck=0.4), xlab="")

#### Subsection 1.4.4: Population parameters and sample statistics

##### The sampling distribution of the mean

library(lattice)  
## Generate n sample values; skew population  
sampfun = function(n) exp(rnorm(n, mean = 0.5, sd = 0.3))  
gph <- DAAG::sampdist(sampsize = c(3, 9, 30), seed = 23, nsamp = 1000,  
 FUN = mean, sampvals=sampfun, plot.type = "density")  
samptheme <- DAAG::DAAGtheme(color=FALSE)  
print(update(gph, scales=list(tck=0.4), layout = c(3,1),  
 par.settings=samptheme, main=list("A: Density curves", cex=1.25)),  
 position=c(0,0.5,1,1), more=TRUE)  
sampfun = function(n) exp(rnorm(n, mean = 0.5, sd = 0.3))  
gph <- DAAG::sampdist(sampsize = c(3, 9, 30), seed = 23, nsamp = 1000,   
 FUN = mean, sampvals=sampfun, plot.type = "qq")  
print(update(gph, scales=list(tck=0.4), layout = c(3,1),  
 par.settings=samptheme,   
 main=list("B: Normal quantile-quantile plots", cex=1.25)),  
 position=c(0,0,1,0.5))

##### The standard error of the median

##### Simulation in learning and research

#### Subsection 1.4.5: The -distribution

x <- seq(from=-4.2, to = 4.2, length.out = 50)  
ylim <- c(0, dnorm(0))  
ylim[2] <- ylim[2]+0.1\*diff(ylim)  
h1 <- dnorm(x)  
h3 <- dt(x, 3)  
h8 <- dt(x,8)  
plot(x, h1, type="l", xlab = "", xaxs="i", ylab = "", yaxs="i",  
bty="L", ylim=ylim, fg="gray")  
mtext(side=3,line=0.5, "A: Normal (t8 overlaid)", adj=-0.2)  
lines(x, h8, col="grey60")  
mtext(side=1, line=1.75, "No. of SEMs from mean")  
mtext(side=2, line=2.0, "Probability density")  
chh <- par()$cxy[2]  
topleft <- par()$usr[c(1,4)] + c(0, 0.6\*chh)  
legend(topleft[1], topleft[2], col=c("black","grey60"),  
lty=c(1,1), legend=c("Normal","t (8 d.f.)"), bty="n", cex=0.8)  
plot(x, h1, type="l", xlab = "", xaxs="i",  
ylab = "", yaxs="i", bty="L", ylim=ylim, fg="gray")  
mtext(side=3,line=0.5, "B: Normal (t3 overlaid)", adj=-0.2)  
lines(x, h3, col="grey60")  
mtext(side=1, line=1.75, "No. of SEMs from mean")  
## mtext(side=2, line=2.0, "Probability density")  
legend(topleft[1], topleft[2], col=c("black","grey60"),  
lty=c(1,1), legend=c("Normal","t (3 d.f.)"), bty="n", cex=0.8)  
## Panels C and D  
cump <- 0.975  
x <- seq(from=-3.9, to = 3.9, length.out = 50)  
ylim <- c(0, dnorm(0))  
plotfun <- function(cump, dfun = dnorm, qfun=qnorm,  
ytxt = "Probability density",  
txt1="qnorm", txt2="", ...)  
{  
h <- dfun(x)  
plot(x, h, type="l", xlab = "", xaxs="i", xaxt="n",  
ylab = ytxt, yaxs="i", bty="L", ylim=ylim, fg="gray",  
...)  
axis(1, at=c(-2, 0), cex=0.8, lwd=0, lwd.ticks=1)  
axis(1, at=c((-3):3), labels=F, lwd=0, lwd.ticks=1)  
tailp <- 1-cump  
z <- qfun(cump)  
ztail <- pretty(c(z,4),20)  
htail <- dfun(ztail)  
polygon(c(z,z,ztail,max(ztail)), c(0,dfun(z),htail,0), col="gray")  
text(0, 0.5\*dfun(z)+0.08\*dfun(0),  
paste(round(tailp, 3), " + ", round(1-2\*tailp,3),  
"\n= ", round(cump, 3), sep=""), cex=0.8)  
lines(rep(z, 2), c(0, dfun(z)))  
lines(rep(-z, 2), c(0, dfun(z)), col="gray60")  
chh <- par()$cxy[2]  
arrows(z, -1.5\*chh,z,-0.1\*chh, length=0.1, xpd=T)  
text(z, -2.5\*chh, paste(txt1, "(", cump, txt2, ")", "\n= ",  
round(z,2), sep=""), xpd=T)  
x1 <- z + .3  
y1 <- dfun(x1)\*0.35  
y0 <- dfun(0)\*0.2  
arrows(-2.75, y0, -x1, y1, length=0.1, col="gray60")  
arrows(2.75, y0, x1, y1, length=0.1)  
text(-2.75, y0+0.5\*chh, tailp, col="gray60")  
text(2.75, y0+0.5\*chh, tailp)  
}  
## ytxt <- "t probability density (8 d.f.)"  
plotfun(cump=cump, cex.lab=1.05)  
mtext(side=3, line=1.25, "C: Normal distribution", adj=-0.2)  
ytxt <- "t probability density (8 d.f.)"  
plotfun(cump=cump, dfun=function(x)dt(x, 8),  
 qfun=function(x)qt(x, 8),  
 ytxt="", txt1="qt", txt2=", 8", cex.lab=1.05)  
mtext(side=3, line=1.25, "D: t distribution (8 df)", adj=-0.2)

qnorm(c(0.975,0.995), mean=0) # normal distribution  
qt(c(0.975, 0.995), df=8) # t-distribution with 8 d.f.

#### Subsection 1.4.6: The likelihood, and maximum likelihood estimation

### Section 1.5: Simple forms of regression model

#### Subsection 1.5.1: Line or curve?

roller <- DAAG::roller  
t(cbind(roller, "depression/weight ratio"=round(roller[,2]/roller[,1],2)))

##### Using models to predict

y <- DAAG::roller$depression  
x <- DAAG::roller$weight  
pretext <- c(reg = "A", lo = "B")  
for(curve in c("reg", "lo")) {  
 plot(x, y, xlab = "Roller weight (t)", xlim=c(0,12.75), fg="gray",  
 ylab = "Depression in lawn (mm)", type="n")  
 points(x, y, cex=0.8, pch = 4)  
 mtext(side = 3, line = 0.25, pretext[curve], adj = 0)  
 topleft <- par()$usr[c(1, 4)]  
 chw <- strwidth("O"); chh <- strheight("O")  
 points(topleft[1]+rep(0.75,2)\*chw,topleft[2]-c(0.75,1.8)\*chh,  
 pch=c(4,1), col=c("black","gray40"), cex=0.8)  
 text(topleft[1]+rep(1.2,2)\*chw, topleft[2]-c(0.75,1.8)\*chh,  
 c("Data values", "Fitted values"),adj=0, cex=0.8)  
 if(curve=="lo")  
 text(topleft[1]+1.2\*chw, topleft[2]-2.85\*chh,"(smooth)", adj=0, cex=0.8)  
 if(curve[1] == "reg") {  
 u <- lm(y ~ -1 + x)  
 abline(0, u$coef[1])  
 yhat <- predict(u)  
}  
else {  
 lawn.lm<-lm(y~x+I(x^2))  
 yhat<-predict(lawn.lm)  
 xnew<-pretty(x,20)  
 b<-lawn.lm$coef  
 ynew<-b[1]+b[2]\*xnew+b[3]\*xnew^2  
 lines(xnew,ynew)  
}  
here <- y < yhat  
yyhat <- as.vector(rbind(y[here], yhat[here], rep(NA, sum(here))))  
xx <- as.vector(rbind(x[here], x[here], rep(NA, sum(here))))  
lines(xx, yyhat, lty = 2, col="gray")  
here <- y > yhat  
yyhat <- as.vector(rbind(y[here], yhat[here], rep(NA, sum(here))))  
xx <- as.vector(rbind(x[here], x[here], rep(NA, sum(here))))  
lines(xx, yyhat, lty = 1, col="gray")  
n <- length(y)  
ns <- min((1:n)[y - yhat >= 0.75\*max(y - yhat)])  
ypos <- 0.5 \* (y[ns] + yhat[ns])  
chw <- par()$cxy[1]  
text(x[ns] - 0.25\*chw, ypos, "+ve residual", adj = 1,cex=0.75, col="gray30")  
points(x, yhat, pch = 1, col="gray40")  
ns <- (1:n)[y - yhat == min(y - yhat)][1]  
ypos <- 0.5 \* (y[ns] + yhat[ns])  
text(x[ns] + 0.4\*chw, ypos, "-ve residual", adj = 0,cex=0.75,col="gray30")  
}

##### Which model is best — line or curve?

#### Subsection 1.5.2: Fitting models – the model formula

## Fit line - by default, this fits intercept & slope.  
roller.lm <- lm(depression ~ weight, data=DAAG::roller)  
## Compare with the code used to plot the data  
plot(depression ~ weight, data=DAAG::roller)  
## Add the fitted line to the plot  
abline(roller.lm)

## For a model that omits the intercept term, specify  
lm(depression ~ 0 + weight, data=roller) # Or, if preferred, replace `0` by `-1`

##### Model objects

roller.lm <- lm(depression ~ weight, data=DAAG::roller)  
names(roller.lm) # Get names of list elements

coef(roller.lm) # Extract coefficients  
summary(roller.lm) # Extract model summary information  
coef(summary(roller.lm)) # Extract coefficients and SEs  
fitted(roller.lm) # Extract fitted values  
predict(roller.lm) # Predictions for existing or new data, with SE  
 # or confidence interval information if required.  
resid(roller.lm) # Extract residuals

roller.lm$coef # An alternative is roller.lm[["coef"]]

print(summary(roller.lm), digits=3)

##### Residual plots

## Normal quantile-quantile plot, plus 7 reference plots  
DAAG::qreference(residuals(roller.lm), nrep=8, nrows=2, xlab="")

##### Simulation of regression data

roller.lm <- lm(depression ~ weight, data=DAAG::roller)  
roller.sim <- simulate(roller.lm, nsim=20) # 20 simulations

with(DAAG::roller, matplot(weight, roller.sim, pch=1, ylim=range(depression)))  
points(DAAG::roller, pch=16)

#### Subsection 1.5.3: The model matrix in regression

model.matrix(roller.lm)  
## Specify coef(roller.lm) to obtain the column multipliers.

##### From straight line regression to multiple regression

mouse.lm <- lm(brainwt ~ lsize+bodywt, data=DAAG::litters)  
coef(summary(mouse.lm))

### Section 1.6: Data-based judgments – frequentist, in a Bayesian world

#### Subsection 1.6.1: Inference with known prior probabilities

## `before` is the `prevalence` or `prior`.   
after <- function(prevalence, sens, spec){  
 prPos <- sens\*prevalence + (1-spec)\*(1-prevalence)  
 sens\*prevalence/prPos}  
## Compare posterior for a prior of 0.002 with those for 0.02 and 0.2  
setNames(round(after(prevalence=c(0.002, 0.02, 0.2), sens=.8, spec=.95), 3),  
 c("Prevalence=0.002", "Prevalence=0.02", "Prevalence=0.2"))

##### Relating ‘incriminating’ evidence to the probability of guilt

#### Subsection 1.6.2: Treatment differences that are on a continuous scale

## Use pipe syntax, introduced in R 4.1.0  
sleep <- with(datasets::sleep, extra[group==2] - extra[group==1])  
sleep |> (function(x)c(mean = mean(x), SD = sd(x), n=length(x)))() |>   
 (function(x)c(x, SEM=x['SD']/sqrt(x['n'])))() |>  
 setNames(c("mean","SD","n","SEM")) -> stats  
 print(stats, digits=3)

## Sum of tail probabilities  
2\*pt(1.580/0.389, 9, lower.tail=FALSE)

## 95% CI for mean of heated-ambient: data frame DAAG::pair65  
t.test(sleep, conf.level=0.95)

##### An hypothesis test

pt(4.06, 9, lower.tail=F)

##### The -value probability relates to a decision process

#### Subsection 1.6.3: Use of simulation with -values

eff2stat <- function(eff=c(.2,.4,.8,1.2), n=c(10,40), numreps=100,  
 FUN=function(x,N)pt(sqrt(N)\*mean(x)/sd(x), df=N-1,   
 lower.tail=FALSE)){  
 simStat <- function(eff=c(.2,.4,.8,1.2), N=10, nrep=100, FUN){  
 num <- N\*nrep\*length(eff)  
 array(rnorm(num, mean=eff), dim=c(length(eff),nrep,N)) |>  
 apply(2:1, FUN, N=N)   
 }  
 mat <- matrix(nrow=numreps\*length(eff),ncol=length(n))  
 for(j in 1:length(n)) mat[,j] <-   
 as.vector(simStat(eff, N=n[j], numreps, FUN=FUN)) ## length(eff)\*numep  
 data.frame(effsize=rep(rep(eff, each=numreps), length(n)),  
 N=rep(n, each=numreps\*length(eff)), stat=as.vector(mat))  
}

set.seed(31)  
df200 <- eff2stat(eff=c(.2,.4,.8,1.2), n=c(10, 40), numreps=200)  
labx <- c(0.001,0.01,0.05,0.2,0.4,0.8)  
gph <- bwplot(factor(effsize) ~ I(stat^0.25) | factor(N), data=df200,   
 layout=c(2,1), xlab="P-value", ylab="Effect size",   
 scales=list(x=list(at=labx^0.25, labels =labx)))  
update(gph+latticeExtra::layer(panel.abline(v=labx[1:3]^0.25, col='lightgray')),  
 strip=strip.custom(factor.levels=paste0("n=",c(10,40))),  
 par.settings=DAAG::DAAGtheme(color=F, col.points="gray50"))

eff10 <- with(subset(df200, N==10&effsize==0.2), c(gt5pc=sum(stat>0.05), lohi=fivenum(stat)[c(2,4)]))  
eff40 <- with(subset(df200, N==40&effsize==0.2), c(gt5pc=sum(stat>0.05), lohi=fivenum(stat)[c(2,4)]))

#### Subsection 1.6.4: Power — minimizing the chance of false positives

tf1 <- rbind('R=0.2'=c(0.8\*50, 0.05\*250),  
'R=1'=c(0.8\*150, 0.05\*150),  
'R=5'=c(0.8\*200, 0.05\*50))  
tf2 <- rbind(c('0.8 x50', '0.05x250'),  
c('0.8x150', '0.05x150'),  
c('0.8x250', '0.05 x50'))  
tf <- cbind("True positives"=paste(tf2[,2],tf1[,2],sep="="),  
"False positives"=paste(tf2[,1],tf1[,1],sep="="))  
rownames(tf) <- rownames(tf1)  
print(tf, quote=FALSE)

##### Power calculations – examples

power.t.test(d=0.5, sig.level=0.05, type="one.sample", power=0.8)  
pwr1 <- power.t.test(d=0.5, sig.level=0.005, type="one.sample", power=0.8)  
pwr2 <- power.t.test(d=0.5, sig.level=0.005, type="two.sample", power=0.8)  
## d=0.5, sig.level=0.005, One- and two-sample numbers   
c("One sample"=pwr1$n, "Two sample"=pwr2$n)

effsize <- c(.05,.2,.4,.8,1.2); npairs <- c(10,20,40)  
pwr0.05 <- matrix(nrow=length(effsize), ncol=length(npairs),  
 dimnames=list(paste0('ES=',effsize), paste0('n=',npairs)))  
pwr0.005 <- matrix(nrow=length(effsize), ncol=length(npairs),  
 dimnames=list(paste0(effsize), paste0('n=',npairs)))  
for(i in 1:length(effsize)) for(j in 1:length(npairs)){  
 pwr0.05[i,j] <- power.t.test(n=npairs[j],d=effsize[i],sig.level=0.05,  
 type='one.sample')$power  
 pwr0.005[i,j] <- power.t.test(n=npairs[j],d=effsize[i],sig.level=0.005,  
 type='one.sample')$power}  
tab <- cbind(round(pwr0.05,4), round(pwr0.005,4))  
tab[1:3,] <- round(tab[1:3,],3)  
tab[5,3] <- '~1.0000'  
tab[5,6] <- '~1.0000'

print(tab[,1:3], quote=F)

print(tab[,4:6], quote=F)

effsize <- c(.05,.2,.4,.8,1.2); npairs <- c(10,20,40)  
pwr0.05 <- matrix(nrow=length(effsize), ncol=length(npairs),  
 dimnames=list(paste0('ES=',effsize), paste0('n=',npairs)))  
pwr0.005 <- matrix(nrow=length(effsize), ncol=length(npairs),  
 dimnames=list(paste0(effsize), paste0('n=',npairs)))  
for(i in 1:length(effsize)) for(j in 1:length(npairs)){  
 pwr0.05[i,j] <- power.t.test(n=npairs[j],d=effsize[i],sig.level=0.05,  
 type='one.sample')$power  
 pwr0.005[i,j] <- power.t.test(n=npairs[j],d=effsize[i],sig.level=0.005,  
 type='one.sample')$power}  
tab <- cbind(round(pwr0.05,4), round(pwr0.005,4))  
tab[1:3,] <- round(tab[1:3,],3)  
tab[5,3] <- '~1.0000'  
tab[5,6] <- '~1.0000'

##### Positive Predictive Values

R <- pretty(0:3, 40)  
postOdds <- outer(R/0.05,c(.8,.3,.08))  
PPV <- as.data.frame(cbind(R,postOdds/(1+postOdds)))  
names(PPV) <- c("R","p80","p30","p8")  
key <- list(text = list(text=c("80% power","30% power", "8% power"), cex = 1.0),  
 x = .6, y = .25, color=F)  
gph <- lattice::xyplot(p80+p30+p8~R, data=PPV, lwd=2, type=c("l","g"),   
 xlab="Pre-study odds R", ylab="Post-study probability (PPV)")  
update(gph, scales=list(tck=0.5), key=key)

#### Subsection 1.6.5: The future for -values

##### How small a -value is needed?

#### Subsection 1.6.6: Reporting results

##### Is there an alternative that is more likely?

### Section 1.7: Information statistics and Bayesian methods with Bayes Factors

#### Subsection 1.7.1: Information statistics – using likelihoods for model choice

## Calculations using mouse brain weight data  
mouse.lm <- lm(brainwt ~ lsize+bodywt, data=DAAG::litters)  
n <- nrow(DAAG::litters)  
RSSlogLik <- with(mouse.lm, n\*(log(sum(residuals^2)/n)+1+log(2\*pi)))  
p <- length(coef(mouse.lm))+1 # NB: p=4 (3 coefficients + 1 scale parameter)  
k <- 2\*n/(n-p-1)  
c("AICc" = AICcmodavg::AICc(mouse.lm), fromlogL=k\*p-2\*logLik(mouse.lm)[1],   
 fromFit=k\*p + RSSlogLik) |> print(digits=4)

##### The sampling properties of the difference in AIC statistics

sim0vs1 <- function(mu=0, n=15, ntimes=200){  
a0 <- a1 <- numeric(ntimes)  
for(i in 1:ntimes){  
 y <- rnorm(n, mean=mu, sd=1)  
 m0 <- lm(y ~ 0); m1 <- lm(y ~ 1)   
 a0[i] <- AIC(m0); a1[i] <- AIC(m1)  
}  
data.frame(a0=a0, a1=a1, diff01=a0-a1, mu=rep(paste0("mu=",mu)))  
}  
library(latticeExtra)  
sim0 <- sim0vs1(mu=0)  
sim0.5 <- sim0vs1(mu=0.5)  
simboth <- rbind(sim0, sim0.5)  
cdiff <- with(list(n=15, p=2), 2\*(p+1)\*p/(n-(p+1)-1))  
xyplot(diff01 ~ a0 | mu, data=simboth, xlab="AIC(m0)", ylab="AIC(m0) - AIC(m1)") +   
 latticeExtra::layer({panel.abline(h=0, col='red');   
 panel.abline(h=cdiff, lwd=1.5, lty=3, col='red', alpha=0.5);  
 panel.abline(h=-2, lty=2, col='red')})

tab <- rbind(c(with(sim0, sum(diff01>0))/200, with(sim0.5, sum(diff01>0))/200),  
 c(with(sim0,sum(diff01>-cdiff))/200, with(sim0.5, sum(diff01>-cdiff))/200))  
dimnames(tab) <- list(c("AIC: Proportion choosing m1",  
 "AICc: Proportion choosing m1"),  
 c("True model is m0", "True model is m1"))  
tab

#### Subsection 1.7.2: Bayesian methods with Bayes Factors

## Setting `scale=1/sqrt(2)` gives a mildly narrower distribution  
print(c("pcauchy(1, scale=1)"=pcauchy(1, scale=1),   
 " pcauchy(1, scale=1/sqrt(2))"=pcauchy(1, scale=1/sqrt(2))),  
 quote=FALSE)

##### The Cauchy prior with different choices of scale parameter

x <- seq(from=-4.5, to=4.5, by=0.1)  
densMed <- dcauchy(x,scale=sqrt(2)/2)  
densUltra <- dcauchy(x, scale=sqrt(2))  
denn <- dnorm(x, sd=1)  
plot(x,densMed, type='l', mgp=c(2,0.5,0), xlab="",  
 ylab="Prior density", col="red", fg='gray')  
mtext(side=1, line=2, expression("Effect size "==phantom(0)\*delta), cex=1.1)  
lines(x, denn, col="blue", lty=2)  
lines(x, densUltra,col=2, lty=2)  
legend("topleft", title="Normal prior",  
 y.intersp=0.8, lty=2, col="blue", bty='n', cex=0.8,  
 legend=expression(bold('sd=1')))  
legend("topright", title="Cauchy priors", y.intersp=0.8,  
 col=c('red', 'red'),lty=c(1,2), cex=0.8,  
 legend=c(expression(bold('medium')),  
 expression(bold('ultrawide'))),bty="n")  
mtext(side=3, line=0.25, adj=0, cex=1.15,  
 expression("A: Alternative priors for "\*delta==frac(mu,sigma)))  
## Panel B  
pairedDiffs <- with(datasets::sleep, extra[group==2] - extra[group==1])  
ttBF0 <- BayesFactor::ttestBF(pairedDiffs)  
simpost <- BayesFactor::posterior(ttBF0, iterations=10000)  
plot(density(simpost[,'mu']), main="", xlab="", col="red",  
 mgp=c(2,0.5,0), ylab="Posterior density", fg='gray')  
mtext(side=1, line=2, expression(mu), cex=1.1)  
abline(v=mean(pairedDiffs), col="gray")  
mtext(side=3, line=0.5, expression("B: Posterior density for "\*mu), adj=0, cex=1.15)

## Calculate and plot density for default prior - Selected lines of code  
x <- seq(from=-4.5, to=4.5, by=0.1)  
densMed <- dcauchy(x, scale=sqrt(2)/2)  
plot(x, densMed, type='l')  
## Panel B  
pairedDiffs <- with(datasets::sleep, extra[group==2] - extra[group==1])  
ttBF0 <- BayesFactor::ttestBF(pairedDiffs)  
## Sample from posterior, and show density plot for mu  
simpost <- BayesFactor::posterior(ttBF0, iterations=10000)  
plot(density(simpost[,'mu']))

##### A thought experiment

tval <- setNames(qt(1-c(.05,.01,.005)/2, df=19), paste(c(.05,.01,.005)))  
bf01 <- setNames(numeric(3), paste(c(.05,.01,.005)))  
for(i in 1:3)bf01[i] <- BayesFactor::ttest.tstat(tval[i],n1=20, simple=T)

pairedDiffs <- with(datasets::sleep, extra[group==2] - extra[group==1])  
ttBF0 <- BayesFactor::ttestBF(pairedDiffs)  
ttBFwide <- BayesFactor::ttestBF(pairedDiffs, rscale=1)  
ttBFultra <- BayesFactor::ttestBF(pairedDiffs, rscale=sqrt(2))  
rscales <- c("medium"=sqrt(2)/2, "wide"=1, ultrawide=sqrt(2))  
BF3 <- c(as.data.frame(ttBF0)[['bf']], as.data.frame(ttBFwide)[['bf']],  
 as.data.frame(ttBFultra)[['bf']])  
setNames(round(BF3,2), c("medium", "wide", "ultrawide"))

pval <- t.test(pairedDiffs)[['p.value']]  
1/(-exp(1)\*pval\*log(pval))

##### A null interval may make better sense

min45 <- round(0.75/sd(pairedDiffs),2) ## Use standardized units  
ttBFint <- BayesFactor::ttestBF(pairedDiffs, nullInterval=c(-min45,min45))  
round(as.data.frame(ttBFint)['bf'],3)

bf01 <- as.data.frame(ttBFint)[['bf']]

##### The effect of changing sample size

t2bfInterval <- function(t, n=10, rscale="medium", mu=c(-.1,.1)){  
 null0 <- BayesFactor::ttest.tstat(t=t, n1=n, nullInterval=mu,  
 rscale=rscale,simple=TRUE)  
alt0 <- BayesFactor::ttest.tstat(t=t, n1=n, nullInterval=mu, rscale=rscale,   
 complement=TRUE, simple=TRUE)  
alt0/null0  
}  
##  
## Calculate Bayes factors  
pval <- c(0.05,0.01,0.001); nval <- c(4,6,10,20,40,80,160)  
bfDF <- expand.grid(p=pval, n=nval)  
pcol <- 1; ncol <- 2; tcol <- 3  
bfDF[,'t'] <- apply(bfDF,1,function(x){qt(x[pcol]/2, df=x[ncol]-1, lower.tail=FALSE)})  
other <- apply(bfDF,1,function(x)  
 c(BayesFactor::ttest.tstat(t=x[tcol], n1=x[ncol], rscale="medium",  
 simple=TRUE),  
## Now specify a null interval  
 t2bfInterval(t=x[tcol], n=x[ncol], mu=c(-0.1,0.1),rscale="medium")  
 ))  
bfDF <- setNames(cbind(bfDF, t(other)),  
 c('p','n','t','bf','bfInterval'))

plabpos <- with(subset(bfDF, n==max(bfDF$n)), log((bf+bfInterval)/2))  
gphA1 <- lattice::xyplot(log(bf)~log(n), groups=factor(p), data=bfDF,  
 panel=function(x,y,...){  
 lattice::panel.xyplot(x,y,type='b',...)})  
ylabA <- 10^((-3):6/2)  
scalesA <- list(x=list(at=log(nval), labels=nval),  
 y=list(at=log(ylabA), labels=signif(ylabA,2)))  
keyA <- list(corner=c(0.99,0.98), lines=list(col=c(1,1), lty=1:2),  
 text=list(c('Point null at 0', "null=(-0.1,0.1)")))  
ylim2 <- log(c(min(bfDF[['bfInterval']])-0.05,150))   
gphA2 <- lattice::xyplot(log(bfInterval)~log(n), groups=factor(p), lty=2,  
 xlim=c(log(3.5),log(max(nval)\*3.25)), ylim=ylim2, data=bfDF,  
 panel=function(x,y,...){  
 panel.xyplot(x,y,type='b',...)  
 panel.grid(h=-1,v=-1)  
 panel.text(rep(log(max(nval\*0.975)),3), plabpos,   
 labels=c('p=0.05','0.01','0.001'), pos=4)  
 },  
 par.settings=DAAG::DAAGtheme(color=T),  
 main="A: Bayes factor vs sample size",   
 xlab="Sample size", ylab="Bayes factor", scales=scalesA, key=keyA)  
## Panel B  
bfDF[['eff']] = bfDF[["t"]]/sqrt(bfDF[['n']])  
ylabB <- 10^((-3):2/3)  
scalesB= list(x=list(at=log(nval), labels=nval),  
 y=list(at=log(ylabB), labels=signif(ylabB,2)))  
keyB <- list(corner=c(0.98,0.975), lines=list(lty=1:3),   
 points=list(pch=1:3), text=list(c('p=0.001','p=0.01','p=0.05')))  
gphB <- xyplot(log(eff)~log(n), groups=log(p), data=bfDF, pch=1:3, lty=1:3,   
 type='b', xlab="Sample size", ylab="Effect size",  
 par.settings=DAAG::DAAGtheme(color=T),  
 main="B: Effect size vs sample size", key=keyB, scales=scalesB) +  
 latticeExtra::layer(panel.grid(h=-1,v=-1))  
plot(gphA2+latticeExtra::as.layer(gphA1), position=c(0, 0, 0.525, 1), more=T)  
plot(gphB, position=c(0.52, 0, 1, 1), par.settings=DAAG::DAAGtheme(color=T))

##### Different statistics give different perspectives

n1 <- BayesFactor::ttest.tstat(qt(0.00001, df=40), n1=40, simple=T)  
n2 <- BayesFactor::ttest.tstat(qt(0.000001, df=40), n1=40, simple=T)

bf1 <- BayesFactor::ttest.tstat(qt(0.00001, df=40), n1=40, simple=T)  
bf2 <- BayesFactor::ttest.tstat(qt(0.000001, df=40), n1=40, simple=T)  
rbind("Bayes Factors"=setNames(c(bf1,bf2), c("p=0.00001","p=0.000001")),  
 "t-statistics"=c(qt(0.00001, df=40), qt(0.000001, df=40)))

knitr::kable(matrix(c("A bare mention","Positive","Strong","Very strong"), nrow=1),  
 col.names=c("1 -- 3", "3 -- 20", "20 -- 150", ">150"), align='c',  
 midrule='', vline="")

##### \*Technical details of the family of priors used in the BayesFactor package

### Section 1.8: Resampling methods for SEs, tests and confidence intervals

#### Subsection 1.8.1: The one-sample permutation test

tab <- t(as.matrix(DAAG::pair65))  
rbind(tab,"heated-ambient"=tab[1,]-tab[2,])

#### Subsection 1.8.2: The two-sample permutation test

## First of 3 curves; permutation distribution of difference in means  
two65 <- DAAG::two65  
set.seed(47) # Repeat curves shown here  
nsim <- 2000; dsims <- numeric(nsim)  
x <- with(two65, c(ambient, heated))  
n <- length(x); n2 <- length(two65$heated)  
dbar <- with(two65, mean(heated)-mean(ambient))  
for(i in 1:nsim){  
 mn <- sample(n,n2,replace=FALSE); dsims[i] <- mean(x[mn]) - mean(x[-mn]) }  
plot(density(dsims), xlab="", main="", lwd=0.5, yaxs="i", ylim=c(0,0.08), bty="n")  
abline(v=c(dbar, -dbar), lty=3)  
pval1 <- (sum(dsims >= abs(dbar)) + sum (dsims <= -abs(dbar)))/nsim  
mtext(side=3,line=0.25,  
 text=expression(bar(italic(x))[2]-bar(italic(x))[1]), at=dbar)  
mtext(side=3,line=0.25,  
 text=expression(-(bar(italic(x))[2] - bar(italic(x))[1])), at=-dbar)  
## Second permutation density  
for(i in 1:nsim){  
mn <- sample(n,n2,replace=FALSE)  
dsims[i] <- mean(x[mn]) - mean(x[-mn])  
}  
pval2 <- (sum(dsims >= abs(dbar)) + sum (dsims <= -abs(dbar)))/nsim  
lines(density(dsims),lty=2,lwd=1)  
## Third permutation density  
for(i in 1:nsim){  
mn <- sample(n,n2,replace=FALSE)  
dsims[i] <- mean(x[mn]) - mean(x[-mn])  
}  
pval3 <- (sum(dsims >= abs(dbar)) + sum (dsims <= -abs(dbar)))/nsim  
lines(density(dsims),lty=3,lwd=1.25)  
box(col="gray")  
leg3 <- paste(c(pval1,pval2,pval3))  
legend(x=20, y=0.078, title="P-values are", cex=1, xpd=TRUE,  
 bty="n", lty=c(1,2,3), lwd=c(1,1,1,1.25), legend=leg3, y.intersp=0.8)

#### Subsection 1.8.3: Estimating the standard error of the median: bootstrapping

## Bootstrap estimate of median of wren length: data frame cuckoos  
wren <- subset(DAAG::cuckoos, species=="wren")[, "length"]  
library(boot)  
## First define median.fun(), with two required arguments:  
## data specifies the data vector,  
## indices selects vector elements for each resample  
median.fun <- function(data, indices){median(data[indices])}  
## Call boot(), with statistic=median.fun, R = # of resamples  
set.seed(23)  
(wren.boot <- boot(data = wren, statistic = median.fun, R = 4999))

#### Subsection 1.8.4: Bootstrap estimates of confidence intervals

##### Bootstrap 95% confidence intervals for the median

## Call the function boot.ci() , with boot.out=wren.boot  
boot.ci(boot.out=wren.boot, type=c("perc","bca"))

##### The correlation coefficient

## Bootstrap estimate of 95% CI for `cor(chest, belly)`: `DAAG::possum`  
corr.fun <- function(data, indices)   
 with(data, cor(belly[indices], chest[indices]))  
set.seed(29)  
corr.boot <- boot(DAAG::possum, corr.fun, R=9999)

library(boot)  
boot.ci(boot.out = corr.boot, type = c("perc", "bca"))

##### The bootstrap – parting comments

### Section 1.9: Organizing and managing work, and tools that can assist

##### The RStudio Integrated Development Environment

#### Subsection 1.9.1: Reproducible reporting — the knitr package

### Section 1.10: The changing environment for data analysis

#### Subsection 1.10.1: Models and machine learning

##### The limits of current machine learning systems

##### Traps in big data analysis

##### Of mice and machine learning — missing data values

##### Humans are not good intuitive statisticians

#### Subsection 1.10.2: Replicability is the definitive check

##### To what extent is published work replicable? What is the evidence?

##### Some major replication studies

##### Replicability in pre-clinical cancer research

##### Studies where there may be strong social influences

##### The scientific study of scientific processes

##### Would lowering the -value threshold help?

##### Peer review at the study planning stage

### Section 1.11: Further, or supplementary, reading

### Exercises (1\_12)

1.4

Animals <- MASS::Animals  
manyMals <- rbind(Animals, sqrt(Animals), Animals^0.1, log(Animals))  
manyMals$transgp <- rep(c("Untransformed", "Square root transform",  
 "Power transform, lambda=0.1", "log transform"),  
rep(nrow(Animals),4))  
manyMals$transgp <- with(manyMals, factor(transgp, levels=unique(transgp)))  
lattice::xyplot(brain~body|transgp, data=manyMals,  
 scales=list(relation='free'), layout=c(2,2))

1.5

with(Animals, c(cor(brain,body), cor(brain,body, method="spearman")))  
with(Animals, c(cor(log(brain),log(body)),  
 cor(log(brain),log(body), method="spearman")))

1.9

usableDF <- DAAG::cuckoohosts[c(1:6,8),]  
nr <- nrow(usableDF)  
with(usableDF, {  
 plot(c(clength, hlength), c(cbreadth, hbreadth), col=rep(1:2,c(nr,nr)))  
 for(i in 1:nr)lines(c(clength[i], hlength[i]), c(cbreadth[i], hbreadth[i]))  
 text(hlength, hbreadth, abbreviate(rownames(usableDF),8), pos=c(2,4,2,1,2,4,2))  
})

1.10

## Take a random sample of 100 values from the normal distribution  
x <- rnorm(100, mean=3, sd=5)  
(xbar <- mean(x))  
## Plot, against `xbar`, the sum of squared deviations from `xbar`  
lsfun <- function(xbar) apply(outer(x, xbar, "-")^2, 2, sum)  
curve(lsfun, from=xbar-0.01, to=xbar+0.01)

boxplot(avs, meds, horizontal=T)

1.15

x <- rpois(7, 78.3)  
mean(x); var(x)

1.16

nvals100 <- rnorm(100)  
heavytail <- rt(100, df = 4)  
veryheavytail <- rt(100, df = 2)  
boxplot(nvals100, heavytail, veryheavytail, horizontal=TRUE)

1.19

boxdists <- function(n=1000, times=10){  
 df <- data.frame(normal=rnorm(n\*times), t=rt(n\*times, 7),  
 sampnum <- rep(1:times, rep(n,times)))  
 lattice::bwplot(sampnum ~ normal+t, data=df, outer=TRUE, xlab="",   
 horizontal=T)  
}

1.20

a <- 1  
form <- ~rchisq(1000,1)^a+rchisq(1000,25)^a+rchisq(1000,500)^a  
lattice::qqmath(form, scales=list(relation="free"), outer=TRUE)

1.21

y <- rnorm(51)  
ydep <- y1[-1] + y1[-51]  
acf(y) # acf plots `autocorrelation function'(see Chapter 6)  
acf(ydep)

1.24

ptFun <- function(x,N)pt(sqrt(N)\*mean(x)/sd(x), df=N-1, lower.tail=FALSE)  
simStat <- function(eff=.4, N=10, nrep=200, FUN)  
 array(rnorm(n=N\*nrep\*length(eff), mean=eff), dim=c(length(eff),nrep,N)) |>  
 apply(2:1, FUN, N=N)   
pval <- simStat(eff=.4, N=10, nrep=200, FUN=ptFun)  
# Suggest a power transform that makes the distribution more symmetric  
car::powerTransform(pval) # See Subsection 2.5.6  
labx <- c(0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25)  
bwplot(~I(pval^0.2), scales=list(x=list(at=labx^0.2, labels=paste(labx))),  
 xlab=expression("P-value (scale is "\*p^{0.2}\*")") )

1.24a

pvalDF <- subset(df200, effsize==0.4 & N==10)$stat  
plot(sort(pval^0.2), sort(pvalDF^0.2))  
abline(0,1)

1.24c

## Estimated effect sizes: Set `FUN=effFun` in the call to `eff2stat()`  
effFun <- function(x,N)mean(x)/sd(x)   
 # Try: `labx <- ((-1):6)/2`; `at = log(labx)`; `v = log(labx)   
## NB also, Bayes Factors: Set `FUN=BFfun` in the call to `eff2stat()`  
BFfun <- function(x,N)BayesFactor::ttest.tstat(sqrt(N)\*mean(x)/sd(x),  
 n1=N, simple=T)  
 # A few very large Bayes Factors are likely to dominate the plots

1.27

(degC <- setNames(c(21,30,38,46),paste('rep',1:4)) )

1.27a

radonC <- tidyr::pivot\_longer(MPV::radon, names\_to='key',   
 cols=names(degC), values\_to='percent')  
radonC$temp <- degC[radonC$key]  
lattice::xyplot(percent ~ temp|factor(diameter), data = radonC)

matplot(scale(t(MPV::radon[,-1])), type="l", ylab="scaled residuals")

1.27d

radon.res <- aggregate(percent ~ diameter, data = radonC, FUN = scale,   
 scale = FALSE)

1.30

diamonds <- ggplot2::diamonds  
with(diamonds, plot(carat, price, pch=16, cex=0.25))  
with(diamonds, smoothScatter(carat, price))

t2bfInterval <- function(t, n=10, rscale="medium", mu=c(-.1,.1)){  
 null0 <- BayesFactor::ttest.tstat(t=t, n1=n, nullInterval=mu,  
 rscale=rscale,simple=TRUE)  
alt0 <- BayesFactor::ttest.tstat(t=t, n1=n, nullInterval=mu, rscale=rscale,   
 complement=TRUE, simple=TRUE)  
alt0/null0  
}

pval <- c(0.05,0.01,0.001); nval <- c(10,40,160)  
bfDF <- expand.grid(p=pval, n=nval)  
pcol <- 1; ncol <- 2; tcol <- 3  
bfDF[,'t'] <- apply(bfDF,1,function(x){qt(x[pcol]/2, df=x[ncol]-1, lower.tail=FALSE)})  
other <- apply(bfDF,1,function(x)  
 c(BayesFactor::ttest.tstat(t=x[tcol], n1=x[ncol], rscale="medium",  
 simple=TRUE),  
 BayesFactor::ttest.tstat(t=x[tcol], n1=x[ncol], rscale="wide",  
 simple=TRUE),  
## Now specify a null interval  
 t2bfInterval(t=x[tcol], n=x[ncol], mu=c(-0.1,0.1),rscale="medium"),  
 t2bfInterval(t=x[tcol], n=x[ncol], mu=c(-0.1,0.1),rscale="wide")  
 ))  
bfDF <- setNames(cbind(bfDF, t(other)),  
 c('p','n','t','bf','bfInterval'))

df <- data.frame(d = with(datasets::sleep, extra[group==2] - extra[group==1]))  
library(statsr)  
BayesFactor::ttestBF(df$d, rscale=1/sqrt(2)) # Or, `rscale="medium"`  
 # `rscale="medium"` is the default  
bayes\_inference(d, type='ht', data=df, statistic='mean', method='t', rscale=1/sqrt(2),  
 alternative='twosided', null=0, prior\_family = "JZS")  
 # Set `rscale=1/sqrt(2)` (default is 1.0)   
 # as for BayesFactor; gives same BF  
# Compare with `prior\_family = "JUI"` (`"JZS"` is the default),   
# with (if not supplied) default settings  
bayes\_inference(d, type='ht', data=df, statistic='mean', method='t',  
 alternative='twosided', null=0, prior\_family = "JUI")

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch1.R")  
}

# 2. Chapter 2: Generalizing from models

##### Packages required (plus any dependencies)

DAAG MASS qra investr HistData BHH2 xtable BayesFactor boot zoo boot MCMCpack,

Additionally, knitr and Hmisc are required in order to process the Rmd source file.

### Section 2.1 Model assumptions

#### Subsection 2.1.1: Inferences are never assumption free

#### Subsection 2.1.2: Has account been taken of all relevant effects?

## Tabulate by Admit and Gender  
byGender <- 100\*prop.table(margin.table(UCBAdmissions, margin=1:2), margin=2)  
round(byGender,1)

## Admission rates, by department  
pcAdmit <- 100\*prop.table(UCBAdmissions, margin=2:3)["Admitted", , ]  
round(pcAdmit,1)

applied <- margin.table(UCBAdmissions, margin=2:3)  
pcAdmit <- 100\*prop.table(UCBAdmissions, margin=2:3)["Admitted", , ]  
 byGender <- 100\*prop.table(margin.table(UCBAdmissions,  
 margin=1:2), margin=2)  
dimnam <- dimnames(UCBAdmissions)  
mfStats <- data.frame(Admit=c(pcAdmit[1,],pcAdmit[2,]),  
 Applicants=c(applied[1,], applied[2,]),  
 mf=factor(rep(dimnam[['Gender']],c(6,6)),  
 levels=dimnam[['Gender']]), Department=rep(dimnam[["Dept"]],2))  
xlim <- c(0, max(mfStats$Admit)\*1.025)  
ylim <- c(0, max(mfStats$Applicants)\*1.075)  
plot(Applicants~Admit, data=mfStats, type="h",lwd=2, xlim=xlim, ylim=ylim,  
 fg="gray", cex.lab=1.2, col=rep(c("blue","red"),rep(6,2)),  
 xlab="UCB Admission rates (%), 1973", ylab="Number of applicants")  
pcA <- rbind(pcAdmit[1,], apply(pcAdmit,2, mean)+2, pcAdmit[2,], rep(NA,6))  
pcA[2,3] <- pcA[2,3]+1  
appA <- rbind(applied[1,], apply(applied,2, mean)+80,  
 applied[2,], rep(NA,6))  
deptNam <- dimnam[[3]]  
for(j in 1: ncol(appA)) lines(pcA[,j], appA[,j], col="gray", lwd=0.8)  
points(pcA[2,],appA[2,], pch=16, cex=1.1, col="white")  
text(pcA[2,],appA[2,],deptNam, cex=0.85)  
##  
par(xpd=TRUE)  
text(byGender[1,1:2], rep(par()$usr[4],2)+0.5\*strheight("^"),  
 labels=c("^","^"), col=c("blue","red"),cex=1.2,srt=180)  
text(byGender[1,], par()$usr[4]+1.4\*strheight("A"),  
 labels=paste(round(byGender[1,],1)),cex=0.85)  
text(byGender[1,1:2]+c(-3.5,3.5), rep(par()$usr[4],2)+2.65\*strheight("A"),  
 labels=c("All males","All females"), pos=c(4,2), cex=1.2)  
par(xpd=FALSE)  
abline(h=200\*(0:4),col="lightgray",lty="dotted")  
abline(v=20\*(0:4),col="lightgray",lty="dotted")  
legend("topleft", col=c('blue','red'),lty=c(1,1), lwd=0.75, cex=0.9,  
 y.intersp=0.65, legend=c("Males","Females"),bty="n")

## Calculate totals, by department, of males & females applying  
margin.table(UCBAdmissions, margin=2:3)

#### Subsection 2.1.3: The limitations of models

#### Subsection 2.1.4: Use the methodology that best suits the task in hand?

### Section 2.2: t-statistics, binomial proportions, and correlations

#### Subsection 2.2.1: One- and two-sample t-tests

#### Subsection 2.2.2: A two-sample comparison

stats2 <- sapply(DAAG::two65,  
 function(x) c(av=mean(x), sd=sd(x), n=length(x)))  
pooledsd <- sqrt( sum(stats2['n',]\*stats2['sd',]^2)/sum(stats2['n',]-1) )  
stats2 <- setNames(c(as.vector(stats2), pooledsd),  
 c('av1','sd1','n1','av2','sd2','n2','pooledsd'))  
print(stats2, digits=4)

with(DAAG::two65, t.test(heated, ambient, var.equal=TRUE))

##### When is pairing helpful?

titl <- paste("Second versus first member, for each pair. The first",  
"\npanel is for the elastic band data. The second (from",  
"\nDarwin) is for plants of the species Reseda lutea")  
oldpar <- par(pty="s")  
on.exit(par(oldpar))  
DAAG::onesamp(dset = DAAG::pair65, x = "ambient", y = "heated",  
 xlab = "Amount of stretch (ambient)",  
 ylab = "Amount of stretch (heated)", fg='gray')  
## Data set mignonette holds the Darwin (1877) data on Reseda lutea.  
## Data were in 5 pots, holding 5,5,5,5,4 pairs of plants respectively.  
DAAG::onesamp(dset = DAAG::mignonette, x = "self", y = "cross",  
 xlab = "Height of self-fertilised plant", ylab =  
 "Height of cross-fertilised plant", dubious = 0, cex=0.7, fg='gray')

##### What if the standard deviations are unequal?

#### Subsection 2.2.3: The normal approximation to the binomial

#### Subsection 2.2.4: The Pearson or product–moment correlation

## Pearson correlation between `body` and `brain`: Animals  
Animals <- MASS::Animals  
rho <- with(Animals, cor(body, brain))  
## Pearson correlation, after log transformation  
rhoLogged <- with(log(Animals), cor(body, brain))  
## Spearman rank correlation  
rhoSpearman <- with(Animals, cor(body, brain, method="spearman"))  
c(Pearson=round(rho,2), " Pearson:log values"=round(rhoLogged,2),  
 Spearman=round(rhoSpearman,2))

### Section 2.3 Extra-binomial and extra-Poisson variation

maleDF <- data.frame(number=0:12, freq=unname(qra::malesINfirst12[["freq"]]))  
N <- sum(maleDF$freq)  
pihat <- with(maleDF, weighted.mean(number, freq))/12  
probBin <- dbinom(0:12, size=12, prob=pihat)  
rbind(Frequency=setNames(maleDF$freq, nm=0:12),  
 binomialFit=setNames(probBin\*N, nm=0:12),  
 rawResiduals = maleDF$freq-probBin\*N,  
 SDbinomial=sqrt(probBin\*(1-probBin)\*N)) |>  
 formatC(digits=2, format="fg") |> print(digits=2, quote=F, right=T)

set.seed(29)  
rqres.plot(doBI, plot.type='all', type="QQ", main=""); box(col='white')  
mtext(side=3, line=0.5, "A: Binomial model, Q-Q", adj=0, cex=1.25)  
rqres.plot(doBI, plot.type='all', type="wp", main=""); box(col='white')  
## Plots C, D, E, F: Set object name; set`type="wp" (C, E, F), or`"QQ"` (D)  
mtext(side=3, line=0.5, "B: Binomial, worm plot 1", adj=-0.05, cex=1.25)  
rqres.plot(doBI, plot.type='all', type="wp", main=""); box(col='white')  
mtext(side=3, line=0.5, "C: Binomial, worm plot 2", adj=-0.05, cex=1.25)  
rqres.plot(doBB, plot.type='all', type="QQ", main="", ylab=''); box(col='white')  
mtext(side=3, line=0.5, "D: BB model, Q-Q", adj=0, cex=1.25)  
rqres.plot(doBB, plot.type='all', type="wp", main="", ylab=''); box(col='white')  
mtext(side=3, line=0.5, "E: BB, worm plot 1", adj=0, cex=1.25)  
rqres.plot(doBB, plot.type='all', type="wp", main="", ylab=''); box(col='white')  
mtext(side=3, line=0.5, "F: BB, worm plot 2", adj=0, cex=1.25)

aicStat <- AIC(doBI, doBB)  
rownames(aicStat) <-  
 c(doBI="Binomial", doBB="Betabinomial")[rownames(aicStat)]  
aicStat$dAIC <- with(aicStat, round(AIC-AIC[1],1))  
aicStat

## Numbers of accidents in three months, with Poisson fit  
machinists <- data.frame(number=0:8, freq=c(296, 74, 26, 8, 4, 4, 1, 0, 1))  
N <- sum(machinists[['freq']])  
lambda <- with(machinists, weighted.mean(number, freq))  
fitPoisson <- dpois(0:8, lambda)\*sum(machinists[['freq']])  
rbind(Frequency=with(machinists, setNames(freq, number)),  
 poissonFit=fitPoisson) |>  
 formatC(digits=2, format="fg") |> print(quote=F, digits=2, right=T)

set.seed(23)  
rqres.plot(doPO, plot.type='all', type="QQ", main=""); box(col='white')  
## Repeat, changing the argument, for remaining plots  
mtext(side=3, line=0.5, "A: Poisson, Q-Q plot", adj=0, cex=1.25)  
rqres.plot(doPO, plot.type='all', type="wp", main="", ylab=''); box(col='white')  
mtext(side=3, line=0.5, "B: Poisson, worm plot", adj=0, cex=1.25)  
rqres.plot(doNBI, plot.type='all', type="wp", main="", ylab='')  
mtext(side=3, line=0.5, "C: NBI, worm plot", adj=0, cex=1.25); box(col='white')

### Subsection 2.3.2: \*Technical details – extra-binomial or extra-Poisson variation

sigma <- exp(coef(doBB, "sigma"))  
cat("Phi =", (1+12\*sigma)/(1+sigma))

mu <- exp(coef(doNBI, "mu"))  
sigma <- exp(coef(doNBI, "sigma"))  
cat("Phi =", (1+sigma\*mu))

### Section 2.4 Contingency tables

## 'Untreated' rows (no training) from psid3, 'treated' rows from nswdemo  
nswpsid3 <- rbind(DAAG::psid3, subset(DAAG::nswdemo, trt==1))  
degTAB <- with(nswpsid3, table(trt,nodeg))  
# Code 'Yes' if completed high school; 'No' if dropout  
dimnames(degTAB) <- list(trt=c("PSID3\_males","NSW\_male\_trainees"),  
 deg =c("Yes","No"))  
degTAB

# To agree with hand calculation below, specify correct=FALSE  
chisq.test(degTAB, correct=FALSE)

##### The mechanics of the chi-squared test

##### An example where a chi-squared test may not be valid

## Engine man data  
engineman <- matrix(c(5,3,17,85), 2,2)  
chisq.test(engineman)

##### Rare and endangered plant species

fisher.test(engineman)

## Enter the data thus:  
rareplants <- matrix(c(37,190,94,23, 59,23,10,141, 28,15,58,16), ncol=3,  
 byrow=TRUE, dimnames=list(c("CC","CR","RC","RR"), c("D","W","WD")))

(x2 <- chisq.test(rareplants))

##### Examination of departures from a consistent overall row pattern

## Expected values  
x2$expected

options(digits=2)  
## Standardized residuals  
residuals(x2)

##### Interpretation issues

### Section 2.5 Issues for Regression with a single explanatory variable

#### Subsection 2.5.1: Iron slag example — check residuals with care!

leg <- c("A: Fitted line", "B: Residuals from line", "C: Variance check")  
ord <- order(DAAG::ironslag[["magnetic"]])  
ironslag <- DAAG::ironslag[ord,]  
slagAlpha.lm <- lm(chemical~magnetic, data=ironslag)  
resval <- residuals(slagAlpha.lm)  
fitchem <- fitted(slagAlpha.lm)  
sqrtabs2 <- sqrt(abs(resval))  
plot(chemical~magnetic, xlab = "Magnetic", ylab = "Chemical",  
 pch = 1, data=ironslag, fg="gray")  
lines(fitchem~ironslag[["magnetic"]])  
mtext(side = 3, line = 0.25, leg[1], adj=-0.1, cex=0.925)  
scatter.smooth(resval~ironslag[["magnetic"]], lpars=list(col="red"), span=0.8,  
 xlab = "Magnetic", ylab = "Residual", fg="gray")  
mtext(side = 3, line = 0.25, leg[2], adj = -0.1, cex=0.925)  
scatter.smooth(sqrtabs2 ~ fitchem, lpars=list(col="red"), span=0.8,  
xlab = "Predicted chemical", fg="gray",  
ylab = expression(sqrt(abs(residual))))  
mtext(side = 3, line = 0.25, leg[3], adj = -0.1, cex=0.8)  
## Diagnostics from fit using loess()  
leg2 <- c("D: Smooth, using loess()",  
 "E: Residuals from smooth",  
 "F: Variance check")  
slag.loess <- loess(chemical~magnetic, data=ironslag, span=0.8)  
resval2 <- slag.loess[["residuals"]]  
fitchem2 <- slag.loess[["fitted"]]  
sqrtabs2 <- sqrt(abs(resval2))  
plot(chemical~magnetic, xlab = "Magnetic", ylab = "Chemical",  
pch = 1, data=ironslag, fg="gray")  
lines(fitchem2 ~ ironslag[["magnetic"]], col="red")  
mtext(side = 3, line = 0.25, leg2[1], adj=-0.1, cex=0.925)  
scatter.smooth(resval2~ironslag[["magnetic"]], span=0.8,  
lpars=list(col="red"),  
xlab = "Magnetic", ylab = "Residual", fg="gray")  
mtext(side = 3, line = 0.25, leg2[2], adj = -0.1, cex=0.925)  
scatter.smooth(sqrtabs2 ~ fitchem2, lpars=list(col="red"),  
span=0.8, xlab = "Predicted chemical", fg="gray",  
ylab = expression(sqrt(abs(residual))))  
mtext(side = 3, line = 0.25, leg2[3], adj = -0.1, cex=0.925)

#### Subsection 2.5.2: The analysis of variance table

roller.lm <- lm(depression ~ weight, data=DAAG::roller)  
anova(roller.lm)

#### Subsection 2.5.3: Outliers, influence, and robust regression

softbacks <- DAAG::softbacks  
x <- softbacks[,"volume"]  
y <- softbacks[,"weight"]  
u <- lm(y ~ x)  
yhat <- predict(u)  
res <- resid(u)  
r <- with(softbacks, cor(x, y))  
xlim <- with(softbacks, range(volume))  
xlim[2] <- xlim[2]+diff(xlim)\*0.08  
plot(y ~ x, xlab = "Volume (cc)", xlim=xlim,  
data=softbacks, ylab = "Weight (g)", pch = 4,  
ylim = range(c(y, yhat)), cex.lab=0.9, fg="gray")  
abline(u$coef[1], u$coef[2], lty = 1)  
bottomright <- par()$usr[c(2, 3)]  
chw <- par()$cxy[1]  
chh <- par()$cxy[2]  
z <- summary(u)$coef  
btxt <- c(paste("a =", format(round(z[1, 1], 1)),  
" SE =", format(round(z[1, 2], 1))),  
paste("b =", format(round(z[2, 1], 2)),  
" SE =", format(round(z[2, 2], 2))))  
legend(bottomright[1], bottomright[2],  
legend=btxt, xjust=1, yjust=0, cex=0.8, bty="n")

softbacks.lm <- lm(weight ~ volume, data=DAAG::softbacks)  
print(coef(summary(softbacks.lm)), digits=3)

plot(softbacks.lm, fg="gray",  
caption = c("A: Residuals vs Fitted", "B: Normal Q-Q",  
"C: Scale-Location", "", "D: Resids vs Leverage"))

##### Robust regression

#### Subsection 2.5.4: Standard errors and confidence intervals

##### Confidence intervals and tests for the slope

SEb <- coef(summary(roller.lm))[2, 2]  
coef(roller.lm)[2] + qt(c(0.025,.975), 8)\*SEb

##### SEs and confidence intervals for predicted values

## Code to obtain fitted values and standard errors (SE, then SE.OBS)  
fit.with.se <- predict(roller.lm, se.fit=TRUE)  
fit.with.se$se.fit # SE  
sqrt(fit.with.se[["se.fit"]]^2+fit.with.se$residual.scale^2) # SE.OBS

predict(roller.lm, interval="confidence", level=0.95)  
predict(roller.lm, interval="prediction", level=0.95) # CI for a new observation

## Depression vs weight, with 95\% pointwise bounds for both  
## the fitted line and predicted values  
investr::plotFit(roller.lm, interval="both", col.conf="red", fg="gray")  
mtext(side=3,line=0.75, "A: Lawn roller data", cex=1.2, adj=-0.25)  
## Male child vs father height, Galton's data  
galtonMales <- subset(HistData::GaltonFamilies, gender=="male")  
galton.lm <- lm(childHeight~father, data=galtonMales)  
investr::plotFit(galton.lm, interval="both", col.conf="red", hide=FALSE,  
 col=adjustcolor('black',alpha=0.5), fg="gray")  
mtext(side=3,line=0.75, "B: Son vs father heights", cex=1.2, adj=-0.25)

##### Implications for design

panelci<-function(data,...)  
{  
nrows<-list(...)$nrows  
ncols<-list(...)$ncols  
if(ncols==1)axis(2, lwd=0, lwd.ticks=1)  
if(ncols==1)axis(1, lwd=0, lwd.ticks=1) else  
axis(3, lwd=0, lwd.ticks=1)  
x<-data$stretch; y<-data$distance  
u <- lm(y ~ x)  
upred <- predict(u, interval="confidence")  
ci <- data.frame(fit=upred[,"fit"],lower=upred[,"lwr"], upper=upred[,"upr"])  
ord<-order(x)  
lines(x[ord], ci[["fit"]][ord], lty=1, lwd=2)  
lines(lowess(x[ord], ci[["upper"]][ord]), lty=2, lwd=2, col="grey")  
lines(lowess(x[ord], ci[["lower"]][ord]), lty=2, lwd=2, col="grey")  
}  
elastic1 <- DAAG::elastic1  
elastic2 <- DAAG::elastic2  
xy<-rbind(elastic2,elastic1)  
nam <- c("Range of stretch 30-65 mm","Range of stretch 42-54 mm")  
trial<-rep(nam, c(dim(elastic2)[1],dim(elastic1)[1]))  
xlim<-range(elastic2$stretch)  
ylim<-range(elastic2$distance)  
xy<-split(xy,trial)  
xy<-lapply(1:length(xy),function(i){c(as.list(xy[[i]]), list(xlim=xlim,  
ylim=ylim))})  
names(xy) <- nam  
DAAG::panelplot(xy,panel=panelci,totrows=1,totcols=2,  
 par.strip.text=list(cex=.9), oma=c(4,4,2.5,2), fg='gray')  
mtext(side = 2, line = 3.35, "Distance moved (cm)", cex=1.1, las=0)  
mtext(side=1,line=3,"Amount of stretch (mm)", cex=1.1)

#### Subsection 2.5.5: There are two regression lines!

## There are two regression lines!  
pair65 <- DAAG::pair65  
bothregs <- function(x=pair65[, "ambient"], y=pair65[, "heated"],  
 xlab="Stretch (band at ambient)", ylab = "Stretch (heated band)", pch=16){  
 plot(y ~ x, xlab = xlab, ylab = ylab, pch = pch, fg="gray")  
 topleft <- par()$usr[c(1, 4)] + c(0.5, -0.5) \* par()$cxy  
 text(topleft[1], topleft[2], paste("r =", round(cor(x, y), 2)), adj = 0)  
 u1 <- lm(y ~ x)  
 abline(u1$coef[1], u1$coef[2])  
 u2 <- lm(x ~ y)  
 abline( - coef(u2)[1]/coef(u2)[2], 1/coef(u2)[2], lty = 2)  
}  
bothregs()  
mtext(side = 3, line = 0.5, "A", adj = 0)  
bothregs(x=trees[, "Girth"], y=trees[, "Height"],  
 xlab="Girth (in)", ylab <- "Height (ft)", pch=16)  
mtext(side = 3, line = 0.5, "B", adj = 0)

##### An alternative to a regression line

#### Subsection 2.5.6: Logarithmic and Power Transformations

## Logarithmic and Power Transformations  
DAAG::powerplot(expr="sqrt(x)", xlab="")  
DAAG::powerplot(expr="x^0.25", xlab="", ylab="")  
DAAG::powerplot(expr="log(x)", xlab="", ylab="")  
DAAG::powerplot(expr="x^2")  
DAAG::powerplot(expr="x^4", ylab="")  
DAAG::powerplot(expr="exp(x)", ylab="")

##### General power transformations — Box-Cox and Yeo-Johnson

#### Subsection 2.5.7: General forms of nonlinear response

#### Subsection 2.5.8: Size and shape data – allometric growth

## Heart weight versus body weight, for 30 Cape fur seals.  
g2.12 <- function()  
{  
cfseal <- DAAG::cfseal  
x <- log(cfseal[,"weight"])  
y <- log(cfseal[, "heart"])  
ylim <- log(c(82.5,1100))  
xlim <- log(c(17,180))  
ylab <- "Heart weight (g, log scale)"  
xlab <- "Body weight (kg, log scale)"  
xtik <- c(20,40,80,160)  
ytik <- c(100,200,400,800)  
plot(x, y, xlab = xlab, ylab = ylab, axes = F, xlim =  
xlim, ylim = ylim, pch = 16, cex=0.85, fg="gray", cex.lab=1.1)  
axis(1, at = log(xtik), labels = paste(xtik), lwd=0, lwd.ticks=1)  
axis(2, at = log(ytik), labels = paste(ytik), lwd=0, lwd.ticks=1)  
box(col="gray")  
form1 <- formula(y ~ x)  
u <- lm(form1, data = cfseal)  
abline(u$coef[1], u$coef[2])  
usum <- summary(u)$coef  
options(digits=3)  
print(usum)  
cwh <- par()$cxy  
eqn <- paste("log y =", round(usum[1, 1], 2), " [",  
round(usum[1, 2], 2), "] +", round(usum[2, 1], 3),  
" [", round(usum[2, 2], 3), "] log x")  
mtext(side=3, line=1.15, eqn, adj = 0.4, cex = 0.8)  
mtext(side=3, line=0.25, "(Values in square brackets are SEs)", adj = 0.4, cex = 0.8)  
}  
g2.12()

##### The allometric growth equation

options(scipen=4)  
cfseal.lm <- lm(log(heart) ~ log(weight), data=DAAG::cfseal)  
print(coef(summary(cfseal.lm)), digits=4)

### Section 2.6 Empirical assessment of predictive accuracy

#### Subsection 2.6.1: The training/test approach, and cross-validation

##### Cross-validation – a tutorial example

houseprices <- DAAG::houseprices  
df <- DAAG::CVlm(houseprices, form.lm = formula(sale.price ~ area),m=3,printit=F,plotit=FALSE)  
panelfun <- function(x,y,subscripts,groups, ...){  
 lattice::panel.superpose(x,y,subscripts,groups, ...)  
 lattice::panel.superpose(x,df[["cvpred"]],subscripts,groups,type="b", cex=0.5, ...)  
}  
gph <- lattice::xyplot(sale.price ~ area, groups=fold, data=df, pch=1:3, panel=panelfun)  
parset <- DAAG::DAAGtheme(color=T, lty=1:3, pch=1:3, lwd=2)  
keylist <- list(lines=TRUE, columns=3, between.columns=1.5, between=1, cex=0.85)  
update(gph, par.settings=parset, auto.key=keylist)

set.seed(29) # Generate results shown  
rand <- sample(rep(1:3, length=15))  
## sample() randomly permutes the vector of values 1:3  
for(i in 1:3) cat(paste0(i,":"), (1:15)[rand == i],"\n")

houseprices <- DAAG::houseprices  
row.names(houseprices) <- (1:nrow(houseprices))  
DAAG::CVlm(houseprices, form.lm = formula(sale.price ~ area), plotit=FALSE)

## Estimate of sigma^2 from regression output  
houseprices <- DAAG::houseprices  
houseprices.lm <- lm(sale.price ~ area, houseprices)  
summary(houseprices.lm)[["sigma"]]^2

#### Subsection 2.6.2: Bootstrapping in regression

houseprices <- DAAG::houseprices  
houseprices.lm <- lm(sale.price ~ area, houseprices)  
print(coef(summary(houseprices.lm)),digits=2)

houseprices.fn <-  
 function (houseprices, index,  
 statfun=function(obj)coef(obj)[2]){  
 house.resample <- houseprices[index, ]  
 house.lm <- lm(sale.price ~ area, data=house.resample)  
 statfun(house.lm) # slope estimate for resampled data  
 }

set.seed(1028) # use to replicate the exact results below  
library(boot) # ensure that the boot package is loaded  
## requires the data frame houseprices (DAAG)  
(houseprices.boot <- boot(houseprices, R=999, statistic=houseprices.fn))

statfun1200 <- function(obj)predict(obj, newdata=data.frame(area=1200))  
price1200.boot <- boot(houseprices, R=999, statistic=houseprices.fn,  
statfun=statfun1200)  
boot.ci(price1200.boot, type="perc") # "basic" is an alternative to "perc"

set.seed(1111)  
library(boot)  
par(las=0)  
houseprices2.fn<-function (houseprices,index){  
house.resample<-houseprices[index,]  
house.lm<-lm(sale.price~area,data=house.resample)  
houseprices$sale.price-predict(house.lm,houseprices)  
# resampled prediction errors  
}  
houseprices <- DAAG::houseprices  
n<-nrow(houseprices)  
R <- 199 ## Will obtain 199 estimates of prediction error  
houseprices.lm<-lm(sale.price~area,data=houseprices)  
houseprices2.boot<-boot(houseprices, R=R, statistic=houseprices2.fn)  
house.fac<-factor(rep(1:n,rep(R,n)))  
plot(house.fac,as.vector(houseprices2.boot$t),  
 ylab="", xlab="House", fg="gray")  
mtext(side=2, line=2, "Prediction Errors")  
mtext(side = 3, line = 0.5, "A", adj = 0)  
boot.se <- apply(houseprices2.boot$t,2,sd)  
model.se <- predict.lm(houseprices.lm,se.fit=T)$se.fit  
plot(boot.se/model.se, ylab="", xlab="House",pch=16, fg="gray")  
mtext(side=2, line=2.0, "Ratio of SEs\nBootstrap to Model-Based", cex=0.9)  
mtext(side = 3, line = 0.5, "B", adj = 0)  
abline(1,0)

##### Commentary

### Section 2.7 One- and two-way comparisons

#### Subsection 2.7.1: One-way comparisons

tomato <- data.frame(Weight = c(1.5, 1.9, 1.3, 1.5, 2.4, 1.5, # Water  
 1.5, 1.2, 1.2, 2.1, 2.9, 1.6, # Nutrient  
 1.9, 1.6, 0.8, 1.15, 0.9, 1.6), # Nutrient+24D  
 trt = factor(rep(c("Water", "Nutrient", "Nutrient+24D"), c(6, 6, 6))))  
## Make `Water` the first level of trt. In aov or lm calculations, it is  
## then taken as the baseline or reference level.  
tomato$trt <- relevel(tomato$trt, ref="Water")

## A: Weights of tomato plants (g)  
library(lattice, quietly=TRUE)  
gph <- stripplot(trt~Weight, aspect=0.35, scale=list(tck=0.6), data=tomato)  
update(gph, scales=list(tck=0.4), cex=0.9, col="black", xlab="",  
 main=list('A: Weights of tomato plants (g)', y=0, cex=1.1))

## B: Summarize comparison between LSD and Tukey's HSD graphically  
tomato.aov <- aov(Weight ~ trt, data=tomato)  
DAAG::onewayPlot(obj=tomato.aov)  
title(main="B: LSD, compared with Tukey HSD", adj=0.1, outer=T,  
 line=-1.0, font.main=1, cex.main=1.25)

BHH2::anovaPlot(tomato.aov)

##### The analysis of variance table

## Do analysis of variance calculations  
anova(tomato.aov)

##### Other multiple comparison tests

#### Subsection 2.7.2: Regression versus qualitative comparisons – issues of power

gph <- DAAG::simulateLinear(alpha=0.6, seed=17, aspect='iso')  
update(gph, par.settings=DAAG::DAAGtheme(color=FALSE, alpha=0.4))

#### Subsection 2.7.3: \*Severe multiplicity — the false discovery rate

##### \*Microarrays and alternatives — technical note

##### The false discovery rate (FDR)

coralPval <- DAAG::coralPval  
pcrit <- c(0.05, 0.02, 0.01, 0.001)  
under <- sapply(pcrit, function(x)sum(coralPval<=x))

expected <- pcrit\*length(coralPval)

fdrtab <- data.frame(Threshold=pcrit, Expected=expected,  
Discoveries=under, FDR=round(expected/under, 4))  
print(xtable::xtable(fdrtab), include.rownames=FALSE, hline.after=FALSE)

fdr <- p.adjust(coralPval, method="BH")

fdrcrit <- c(0.05, 0.04, 0.02, 0.01)  
under <- sapply(fdrcrit, function(x)sum(coralPval<=x))  
setNames(under, paste(fdrcrit))

#### Subsection 2.7.4: Data with a two-way structure, i.e., two factors

par(fig=c(0.525,1,0,1), mgp=c(1.5,0.4,0))  
lev <- c("F10", "NH4Cl", "NH4NO3", "F10 +ANU843",  
 "NH4Cl +ANU843", "NH4NO3 +ANU843")  
rice <- within(DAAG::rice, trt <- factor(trt, levels=lev))  
with(rice, interaction.plot(fert, variety, ShootDryMass, fg="gray",  
 legend = FALSE, xlab="Fertiliser", cex.lab=0.95, mex=0.65))  
xleg <- par()$usr[2]  
yleg <- par()$usr[4] - 0.72 \* diff(par()$usr[3:4])  
leginfo <- legend(xleg, yleg, bty = "n", legend = levels(rice$variety),  
 col = 1, lty = 2:1, lwd=1, xjust = 1, cex = 0.8,  
 y.intersp=0.8)$rect  
text(leginfo$left + 0.5 \* leginfo$w, leginfo$top, " variety",  
 adj = 0.5, cex = 0.8)  
mtext(side=3, line=0.65, cex=0.9, adj=-0.15, "B")  
gph <- dotplot(trt ~ ShootDryMass, pch=1, cex=0.9, las=2,  
 xlab="Shoot dry mass (g)", data=rice,  
 panel=function(x,y,...){panel.dotplot(x,y,...)  
 av <- sapply(split(x,y),mean);  
 ypos <- unique(y)  
 lpoints(ypos~av, pch=3, col="gray40", cex=1.25)},  
 main=list("A", cex=0.88, just="left", x=0.1, y=-0.7, font=1))  
pars <- DAAG::DAAGtheme(fontsize=list(text=9, points=6), color=FALSE)  
print(update(gph, scales=list(tck=0.5), par.settings=pars, aspect=0.9),  
 position=c(-0.065,0.0,0.6,1), newpage=FALSE)

#### Subsection 2.7.5: Presentation issues

### Section 2.8 Data with a nested variation structure

#### Subsection 2.8.1: Degrees of freedom considerations

#### Subsection 2.8.2: General multi-way analysis of variance designs

### Section 2.9 Bayesian estimation – further commentary and approaches

#### Subsection 2.9.1: Bayesian estimation with normal priors and likelihood

#### Subsection 2.9.2: Further comments on Bayes Factors

##### The Sellke calibration upper limit

##### A note on the Bayesian Information Criterion

pval <- c(.05,.01,.001); np <- length(pval)  
Nval <- c(4,6,10,20,40,80,160); nlen <- length(Nval)  
## Difference in BIC statistics, interpreted as Bayes factor  
t2BFbic <- function(p,N){t <- qt(p/2, df=N-1, lower.tail=FALSE)  
 exp((N\*log(1+t^2/(N-1))-log(N))/2)}  
bicVal <- outer(pval, Nval, t2BFbic)  
## Bayes factor, calculated using BayesFactor::ttest.tstat()  
t2BF <- function(p, N){t <- qt(p/2, df=N-1, lower.tail=FALSE)  
 BayesFactor::ttest.tstat(t=t, n1=N, simple=TRUE, rscale = "medium")}  
BFval <- matrix(nrow=np, ncol=nlen)  
for(i in 1:np)for(j in 1:nlen) BFval[i,j] <- t2BF(pval[i], Nval[j])  
cfVal <- rbind(BFval, bicVal)[c(1,4,2,5,3,6),]  
dimnames(cfVal) <- list(  
 paste(rep(pval,rep(2,np)), rep(c("- from ttest.tstat", "- from BIC"),np)),  
 paste0(c("n=",rep("",nlen-1)),Nval))  
round(cfVal,1)

#### Subsection 2.9.3: Bayesian regression estimation using the MCMCpack package

suppressPackageStartupMessages(library(MCMCpack))  
roller.mcmc <- MCMCregress(depression ~ weight, data=DAAG::roller)  
summary(roller.mcmc)

mat <- matrix(c(1:6), byrow=TRUE, ncol=2)  
layout(mat, widths=rep(c(2,1.1),3), heights=rep(0.9,8))  
 # NB: widths & heights are relative  
plot(roller.mcmc, auto.layout=FALSE, ask=FALSE, col="gray", fg="gray")

### Section 2.10: Recap

##### Regress on , or on~?

### Section 2.11: Further reading

### Exercises (2.12)

2.2

## UCBAdmissions is in the datasets package  
## For each combination of margins 1 and 2, calculate the sum  
UCBtotal <- apply(UCBAdmissions, c(1,2), sum)

2.2b

apply(UCBAdmissions, 3, function(x)(x[1,1]\*x[2,2])/(x[1,2]\*x[2,1]))

2.3

tabA <- array(c(30,30,10,10,15,5,30,10), dim=c(2,2,2))  
tabB <- array(c(30,30,20,10,10,5,20,25), dim=c(2,2,2))

2.5

z.transform <- function(r) .5\*log((1+r)/(1-r))  
z.inverse <- function(z) (exp(2\*z)-1)/(exp(2\*z)+1)  
 possum.fun <- function(data, indices) {  
 chest <- data$chest[indices]  
 belly <- data$belly[indices]  
 z.transform(cor(belly, chest))}  
possum.boot <- boot::boot(DAAG::possum, possum.fun, R=999)  
z.inverse(boot.ci(possum.boot, type="perc")$percent[4:5])  
 # The 4th and 5th elements of the percent list element  
 # hold the interval endpoints. See ?boot.ci

2.11

with(pressure, MASS::boxcox(pressure ~ I(1/(temperature+273))))

2.14

"funRel" <-  
function(x=leafshape$logpet, y=leafshape$loglen, scale=c(1,1)){  
 ## Find principal components rotation; see Subsection 9.1.2  
 ## Here (unlike 9.1.2) the interest is in the final component  
 xy.prc <- prcomp(cbind(x,y), scale=scale)  
 b <- xy.prc$rotation[,2]/scale  
 c(bxy = -b[1]/b[2]) # slope of functional equation line  
}  
## Try the following:  
leafshape <- DAAG::leafshape  
funRel(scale=c(1,1)) # Take x and y errors as equally important  
 # Note that all lines pass through (mean(x), mean(y))

2.15

P <- rbind(  
 c(1 , 0 , 0 , 0 , 0 , 0),  
 c(.5, 0 , .5, 0 , 0 , 0),  
 c(0 , .5, 0 , .5, 0 , 0),  
 c(0 , 0 , .5, 0 , .5, 0),  
 c(0 , 0 , 0 , .5, 0 , .5),  
 c(0 , 0 , 0 , 0 , 0 , 1))  
dimnames(P) <- list(0:5,0:5)  
P

Markov <- function(N=15, initial.value=1, transition=P, stopval=NULL)  
 {X <- numeric(N)  
 X[1] <- initial.value + 1 # States 0:(n-1); subscripts 1:n  
 n <- nrow(transition)  
 for (i in 2:N){  
 X[i] <- sample(1:n, size=1, prob=transition[X[i-1], ])  
 if(length(stopval)>0)if(X[i] %in% (stopval+1)){X <- X[1:i]; break}}  
 X - 1  
}  
 # Set `stopval=c(0,5)` to stop when the player's fortune is $0 or $5

2.16

Pb <- rbind(  
 Sun = c(Sun=0.6, Cloud=0.2, Rain=0.2),  
 Cloud= c(0.2, 0.4, 0.4),  
 Rain= c(0.4, 0.3, 0.3))  
Pb

2.16b

plotmarkov <-  
 function(n=1000, width=101, start=0, transition=Pb, npanels=5){  
 xc2 <- Markov(n, initial.value=start, transition)  
 mav0 <- zoo::rollmean(as.integer(xc2==0), k=width)  
 mav1 <- zoo::rollmean(as.integer(xc2==1), k=width)  
 npanel <- cut(1:length(mav0), breaks=seq(from=1, to=length(mav0),  
 length=npanels+1), include.lowest=TRUE)  
 df <- data.frame(av0=mav0, av1=mav1, x=1:length(mav0), gp=npanel)  
 print(xyplot(av0+av1 ~ x | gp, data=df, layout=c(1,npanels), type="l",  
 par.strip.text=list(cex=0.65), auto.key=list(columns=2),  
 scales=list(x=list(relation="free"))))  
}

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch2.R")  
}

# 3. Chapter 3: Multiple linear regression

##### Packages required (plus any dependencies)

DAAG car MASS AICcmodavg leaps BayesFactor splines

Additionally, Hmisc and knitr are required in order to process the Rmd source file.

### Section 3.1 Basic ideas: the allbacks book weight data

allbacks <- DAAG::allbacks # Place the data in the workspace  
allbacks.lm <- lm(weight ~ volume+area, data=allbacks)  
print(coef(summary(allbacks.lm)), digits=2)

xlim <- range(allbacks$volume)  
xlim <- xlim+c(-.075,.075)\*diff(xlim)  
## Plot of weight vs volume: data frame allbacks (DAAG)  
plot(weight ~ volume, data=allbacks, pch=c(16,1)[unclass(cover)],  
lwd=1.25, xlim=xlim, fg="gray")  
## unclass(cover) gives the integer codes that identify levels  
## As text() does not accept the parameter data, use with()  
## to specify the data frame.  
with(allbacks, text(weight ~ volume, labels=paste(1:15), cex=0.75, offset=0.35,  
pos=c(2,4)[unclass(cover)]))  
legend(x='topleft', pch=c(16,1), legend=c("hardback ","softback"),  
horiz=T, bty="n", xjust=0.5, x.intersp=0.75, )

## Correlations between estimates -- model with intercept  
round(summary(allbacks.lm, corr=TRUE)$correlation, 3)

out <- capture.output(summary(allbacks.lm,digits=2))  
cat(out[15:17], sep='\n')

## 5% critical value; t-statistic with 12 d.f.  
qt(0.975, 12)

cat(out[5:7], sep='\n')

#### Subsection 3.1.1: A sequential analysis of variance table

anova(allbacks.lm)

##### Omission of the intercept term

## Show rows 1, 7, 8 and 15 only  
model.matrix(allbacks.lm)[c(1,7,8,15), ]  
## NB, also, code that returns the data frame used  
model.frame(allbacks.lm)

allbacks.lm0 <- lm(weight ~ -1+volume+area, data=allbacks)  
print(coef(summary(allbacks.lm0)), digits=2)

## Correlations between estimates -- no intercept  
print(round(summary(allbacks.lm0, corr=TRUE)$correlation, 3))

#### Subsection 3.1.2: Diagnostic plots

allbacks.lm0 <- lm(weight ~ -1+volume+area, data=allbacks)  
plot(allbacks.lm0, caption=c('A: Resids vs Fitted', 'B: Normal Q-Q',  
 'C: Scale-Location', '', 'D: Resids vs Leverage'), cex.caption=0.85,  
 fg='gray')

## To show all plots in the one row, precede with  
par(mfrow=c(1,4)) # Follow with par(mfrow=c(1,1))

## The following has the default captions  
plot(allbacks.lm0)

allbacks.lm13 <- lm(weight ~ -1+volume+area, data=allbacks[-13, ])  
print(coef(summary(allbacks.lm13)), digits=2)

### Section 3.2 The interpretation of model coefficients

#### Subsection 3.2.1: Times for Northern Irish hill races

oldpar <- par(fg='gray20',col.axis='gray20',lwd=0.5,col.lab='gray20')  
nihr <- within(DAAG::nihills, {mph <- dist/time; gradient <- climb/dist})  
nihr <- nihr[, c("time", "dist", "climb", "gradient", "mph")]  
varLabs <- c("\ntime\n(hours)","\ndist\n(miles)","\nclimb\n(feet)",  
 "\ngradient\n(ft/mi)", "\nmph\n(mph)")  
smoothPars <- list(col.smooth='red', lty.smooth=2, lwd.smooth=0.5, spread=0)  
car::spm(nihr, cex.labels=1.2, regLine=FALSE, col='blue',  
 oma=c(1.95,3,4, 3), gap=.25, var.labels=varLabs, smooth=smoothPars)  
title(main="A: Untransformed scales:", outer=TRUE,  
adj=0, line=-1.0, cex.main=1, font.main=1)  
## Panel B: Repeat with log(nihills) in place of nihills,  
## and with variable labels suitably modified.  
varLabs <- c("\ntime\n(log h)","\ndist\n(log miles)", "\nclimb\n(log feet)",  
 "\ngradient\n(log ft/mi)", "\nmph\n(log mph)")  
car::spm(log(nihr), regLine=FALSE, col="blue", oma=c(1.95,2.5,4, 2.5),  
 gap=.25, var.labels=varLabs, smooth=smoothPars)  
title("B: Logarithmic scales", outer=TRUE,  
 adj=0, line=-1.0, cex.main=1, font.main=1)  
par(oldpar)

##### What is special about logarithmic transformations?

#### Subsection 3.2.2: An equation that predicts dist/time

## Hold climb constant at mean on logarithmic scale  
mphClimb.lm <- lm(mph ~ log(dist)+log(climb), data = nihr)  
## Hold `gradient=climb/dist` constant at mean on logarithmic scale  
mphGradient.lm <- lm(mph ~ log(dist)+log(gradient), data = nihr)  
avRate <- mean(nihr$mph)  
bClimb <- coef(mphClimb.lm)  
constCl <- c(bClimb[1]+bClimb[3]\*mean(log(nihr$climb)), bClimb[2])  
bGradient <- coef(mphGradient.lm)  
constSl <- c(bGradient[1]+bGradient[3]\*mean((log(nihr$climb/nihr$dist))),  
 bGradient[2])  
# Use `dist` and `climb` as explanatory variables  
coef(mphClimb.lm)  
# Use `dist` and `gradient` as explanatory variables  
coef(mphGradient.lm)

opar <- par(mfrow=c(1,2), mgp=c(2.25,0.5,0), mar=c(3.6,4.1,2.1,1.6))  
lineCols <- c("red", adjustcolor("magenta",0.4))  
yaxlab<-substitute(paste("Minutes per mile (Add ", ym, ")"), list(ym=round(avRate,2)))  
car::crPlots(mphClimb.lm, terms = . ~ log(dist), xaxt='n',  
 xlab="Distance", col.lines=lineCols, ylab=yaxlab)  
axis(2, at=4:7, labels=paste(4:7))  
labx <- c(4,8,16,32)  
axis(1, at=log(2^(2:5)), labels=paste(2^(2:5)))  
box(col="white")  
mtext("A: Hold climb constant at mean value", adj=0,  
 line=0.8, at=0.6, cex=1.15)  
car::crPlots(mphGradient.lm, terms = . ~log(dist), xaxt='n',  
 xlab="Distance", col.lines=lineCols, ylab=yaxlab)  
axis(1, at=log(2^(2:5)), labels=paste(2^(2:5)))  
axis(2, at=4:7, labels=paste(4:7))  
box(col="white")  
mtext("B: Hold log(gradient) constant at mean", adj=0, line=0.8, at=0.6, cex=1.15)  
par(opar)

summary(mphClimb.lm, corr=T)$correlation["log(dist)", "log(climb)"]  
summary(mphGradient.lm, corr=T)$correlation["log(dist)", "log(gradient)"]

## Show the plots, with default captions  
plot(mphClimb.lm, fg='gray')

plot(mphGradient.lm, caption=c('A: Resids vs Fitted', 'B: Normal Q-Q',  
'C: Scale-Location', '', 'D: Resids vs Leverage'),  
cex.caption=0.85, fg='gray')

#### Subsection 3.2.3: Equations that predict log(time)

lognihr <- setNames(log(nihr), paste0("log", names(nihr)))  
timeClimb.lm <- lm(logtime ~ logdist + logclimb, data = lognihr)

print(coef(summary(timeClimb.lm)), digits=2)

timeGradient.lm <- lm(logtime ~ logdist + loggradient, data=lognihr)  
print(coef(summary(timeGradient.lm)), digits=3)

#### Subsection 3.2.4: Book dimensions — the oddbooks dataset

oldpar <- par(fg='gray40',col.axis='gray20',lwd=0.5,col.lab='gray20')  
## Code for Panel A  
oddbooks <- DAAG::oddbooks  
pairs(log(oddbooks), lower.panel=panel.smooth, upper.panel=panel.smooth,  
 labels=c("log(thick)", "log(breadth)", "log(height)", "log(weight)"),  
 gap=0.25, oma=c(1.95,1.95,4, 1.95), col='blue')  
title(main="A: Columns from log(oddbooks)",  
 outer=TRUE, adj=0, line=-1.0, cex.main=1.1, font.main=1)  
## Panel B  
oddothers <-  
 with(oddbooks, data.frame(density = weight/(breadth\*height\*thick),  
area = breadth\*height, thick=thick, weight=weight))  
pairs(log(oddothers), lower.panel=panel.smooth, upper.panel=panel.smooth,  
labels=c("log(density)", "log(area)", "log(thick)", "log(weight)"),  
gap=0.5, oma=c(1.95,1.95,4, 1.95), col='blue')  
title("B: Add density & area; omit breadth & height",  
outer=TRUE, adj=0, line=-1.0, cex.main=1.1, font.main=1)  
par(oldpar)

lob3.lm <- lm(log(weight) ~ log(thick)+log(breadth)+log(height),  
 data=oddbooks)  
# coef(summary(lob3.lm))

lob2.lm <- lm(log(weight) ~ log(thick)+log(breadth), data=oddbooks)  
coef(summary(lob2.lm))

lob0.lm <- lm(log(weight) ~ 1, data=oddbooks)  
add1(lob0.lm, scope=~log(breadth) + log(thick) + log(height))  
lob1.lm <- update(lob0.lm, formula=. ~ .+log(breadth))

round(rbind("lob1.lm"=predict(lob1.lm), "lob2.lm"=predict(lob2.lm),  
 "lob3.lm"=predict(lob3.lm)),2)

oddbooks <- within(oddbooks, density <- weight/(thick\*breadth\*height))  
lm(log(weight) ~ log(density), data=oddbooks) |> summary() |> coef() |>  
 round(3)

## Code that the reader may care to try  
lm(log(weight) ~ log(thick)+log(breadth)+log(height)+log(density),  
 data=oddbooks) |> summary() |> coef() |> round(3)

#### Subsection 3.2.5: Mouse brain weight example

oldpar <- par(fg='gray40',col.axis='gray20',lwd=0.5,col.lab='gray20')  
litters <- DAAG::litters  
pairs(litters, labels=c("lsize\n\n(litter size)", "bodywt\n\n(Body Weight)",  
 "brainwt\n\n(Brain Weight)"), gap=0.5, fg='gray',  
 col="blue", oma=rep(1.95,4))  
par(oldpar)

## Regression of brainwt on lsize  
summary(lm(brainwt ~ lsize, data = litters), digits=3)$coef  
## Regression of brainwt on lsize and bodywt  
summary(lm(brainwt ~ lsize + bodywt, data = litters), digits=3)$coef

#### Subsection 3.2.6: Issues for causal interpretation

##### Effects of lifestyle on health

##### The studies mostly agree. But what do they say?

##### Adjusting for confounders

### Section 3.3 Choosing the model, and checking it out

##### Effects of lifestyle on health

##### The studies mostly agree. But what do they say?

##### Adjusting for confounders

oddbooks.lm <- lm((weight) ~ log(thick)+log(height)+log(breadth),  
data=DAAG::oddbooks)  
yterms <- predict(oddbooks.lm, type="terms")

#### Subsection 3.3.3: A more formal approach to the choice of transformation

## Use car::powerTransform  
nihr <- within(DAAG::nihills, {mph <- dist/time; gradient <- climb/dist})  
summary(car::powerTransform(nihr[, c("dist", "gradient")]), digits=3)

form <- mph ~ log(dist) + log(gradient)  
summary(car::powerTransform(form, data=nihr))

##### The use of transformations — further comments

#### Subsection 3.3.4: Accuracy estimates, fitted values and new observations

lognihr <- log(DAAG::nihills)  
names(lognihr) <- paste0("log", names(lognihr))  
timeClimb.lm <- lm(logtime ~ logdist + logclimb, data = lognihr)  
## Coverage intervals; use exp() to undo the log transformation  
citimes <- exp(predict(timeClimb.lm, interval="confidence"))  
## Prediction intervals, i.e., for new observations  
pitimes <- exp(predict(timeClimb.lm, newdata=lognihr, interval="prediction"))  
## fit ci:lwr ci:pwr pi:lwr pi:upr  
ci\_then\_pi <- cbind(citimes, pitimes[,2:3])  
colnames(ci\_then\_pi) <- paste0(c("", rep(c("ci-","pi-"), c(2,2))),  
 colnames(ci\_then\_pi))  
## First 4 rows  
print(ci\_then\_pi[1:4,], digits=2)

timeClimb2.lm <- update(timeClimb.lm, formula = . ~ . + I(logdist^2))  
g3.10 <-  
function(model1=timeClimb.lm, model2=timeClimb2.lm)  
{  
## Panel A  
citimes <- predict(model1, interval="confidence")  
ord <- order(citimes[,"fit"])  
citimes <- citimes[ord,]  
hat <- citimes[,"fit"]  
pitimes <- predict(model1, newdata=lognihr, interval="prediction")[ord,]  
logobs <- log(nihr[ord,"time"])  
xtiks <- pretty(exp(hat))  
ylim <- range(c(pitimes[,"lwr"], pitimes[,"upr"], logobs)-rep(hat,3))  
logytiks <- pretty(ylim,5)  
ytiks <- round(exp(logytiks),2)  
xlim <- range(hat)  
plot(hat, citimes[,"lwr"]-hat, type="n", xlab = "Time (fitted)",  
ylab = "Difference from fit",  
xlim=xlim, ylim = ylim, xaxt="n", yaxt="n", fg="gray")  
mtext(side=3, line=0.75, adj=0, at=-2.0, "A: CIs and PIs: Mean, prediction")  
mtext(side=4, line=1.25, "exp(Difference from fit)", las=0)  
axis(1, at=log(xtiks), labels=paste(xtiks), lwd=0, lwd.ticks=1)  
axis(2, at=logytiks, las=1, lwd=0, lwd.ticks=1)  
axis(4, at=logytiks, labels=paste(ytiks), las=0, lwd=0, lwd.ticks=1)  
points(hat, logobs-hat, pch=16, cex=0.65)  
lines(hat, citimes[,"lwr"]-hat, col = "red")  
lines(hat, citimes[,"upr"]-hat, col = "red")  
lines(hat, pitimes[,"lwr"]-hat, col = "black")  
lines(hat, pitimes[,"upr"]-hat, col = "black")  
## Panel B  
citimes2 <- predict(model2, interval="confidence")[ord,]  
plot(hat, citimes[,"lwr"]-hat, type="n", xlab = "Time (fitted)",  
ylab = "Difference from fit",  
xlim=xlim, ylim = ylim, xaxt="n", yaxt="n", fg="gray")  
mtext(side=3, line=0.75, adj=0, at=-2.0,  
"B: CIs for fit, compare two models")  
mtext(side=4, line=1.25, "exp(Difference from fit)", las=0)  
axis(1, at=log(xtiks), labels=paste(xtiks), lwd=0, lwd.ticks=1)  
axis(2, at=logytiks,las=1, lwd=0, lwd.ticks=1)  
axis(4, at=logytiks, labels=paste(ytiks), las=0,, lwd=0, lwd.ticks=1)  
points(hat, logobs-hat, pch=16, cex=0.65)  
lines(hat, citimes[,"lwr"]-hat, col = "red")  
lines(hat, citimes[,"upr"]-hat, col = "red")  
hat2 <- citimes2[,"fit"]  
lines(hat, citimes2[,"lwr"]-hat2, col = "blue", lty=2, lwd=1.5)  
lines(hat, citimes2[,"upr"]-hat2, col = "blue", lty=2, lwd=1.5)  
}

timeClimb2.lm <- update(timeClimb.lm, formula = . ~ . + I(logdist^2))

#### Subsection 3.3.5: Choosing the model — deaths from Atlantic hurricanes

oldpar <- par(fg='gray20',col.axis='gray20',lwd=0.5,col.lab='gray20')  
hurric <- DAAG::hurricNamed[,c("LF.PressureMB", "BaseDam2014", "deaths")]  
thurric <- car::powerTransform(hurric, family="yjPower")  
transY <- car::yjPower(hurric, coef(thurric, round=TRUE))  
smoothPars <- list(col.smooth='red', lty.smooth=2, lwd.smooth=1, spread=0)  
car::spm(transY, lwd=0.5, regLine=FALSE, oma=rep(2.5,4), gap=0.5,  
 col="blue", smooth=smoothPars, cex.labels=1)  
par(oldpar)

modelform <- deaths ~ log(BaseDam2014) + LF.PressureMB  
powerT <- car::powerTransform(modelform, data=as.data.frame(hurric),  
 family="yjPower")  
summary(powerT, digits=3)

deathP <- with(hurric, car::yjPower(deaths, lambda=-0.2))  
power.lm <- MASS::rlm(deathP ~ log(BaseDam2014) + LF.PressureMB, data=hurric)  
print(coef(summary(power.lm)),digits=2)

## Use (deaths+1)^(-0.2) as outcome variable  
plot(power.lm, cex.caption=0.85, fg="gray",  
 caption=list('A: Resids vs Fitted', 'B: Normal Q-Q', 'C: Scale-Location', '',  
 'D: Resids vs Leverage'))

#### Subsection 3.3.6: Strategies for fitting models — suggested steps

##### Diagnostic checks

### Section 3.4 Robust regression, outliers, and influence

#### Subsection 3.4.1: Making outliers obvious — robust regression

hills2000 <- DAAG::hills2000[,c("dist", "climb", "time")]  
varLabels <- c("\ndist\n(log miles)", "\nclimb\n(log feet)", "\ntime\n(log hours)")  
smoothPars <- list(col.smooth='red', lty.smooth=2, lwd.smooth=1, spread=0)  
hills2000 <- DAAG::hills2000[,c("dist", "climb", "time")]  
varLabels <- c("\ndist\n(log miles)", "\nclimb\n(log feet)", "\ntime\n(log hours)")  
car::spm(log(hills2000), smooth=smoothPars, regLine=FALSE, cex.labels=1.5,  
var.labels = varLabels, lwd=0.5, gap=0.5, oma=c(1.95,1.95,1.95,1.95))

## Panel A  
lhills2k.lm <- lm(log(time) ~ log(climb) + log(dist), data = hills2000)  
plot(lhills2k.lm, caption="", which=1, fg="gray", col=adjustcolor("black", alpha=0.8))  
mtext(side=3, line=0.75, "A: Least squares (lm) fit", adj=0, cex=1.1)  
## Panel B  
lhills2k.lqs <- MASS::lqs(log(time) ~ log(climb) + log(dist), data = hills2000)  
reres <- residuals(lhills2k.lqs)  
refit <- fitted(lhills2k.lqs)  
big3 <- which(abs(reres) >= sort(abs(reres), decreasing=TRUE)[3])  
plot(reres ~ refit, xlab="Fitted values (resistant fit)",  
ylab="Residuals (resistant fit)", col=adjustcolor("black", alpha=0.8), fg="gray")  
lines(lowess(reres ~ refit), col=2)  
text(reres[big3] ~ refit[big3], labels=rownames(hills2000)[big3],  
pos=4-2\*(refit[big3] > mean(refit)), cex=0.8)  
mtext(side=3, line=0.75, "B: Resistant (lqs) fit", adj=0, cex=1.1)

## Show only the 2nd diognostic plot, i.e., a normal Q-Q plot  
## plot(lhills2k.lm, which=2)

##### Outliers, influential or not, should be taken seriously

#### Subsection 3.4.2: Leverage, influence, and Cook’s distance

##### Leverage and the hat matrix — technical details

round(unname(hatvalues(timeClimb.lm)),2)

##### Influential points and Cook’s distance

##### Dynamic graphics

## Residuals versus leverages  
nihills <- DAAG::nihills  
timeClimb.lm <- lm(log(time) ~ log(dist) + log(climb), data = nihills)  
plot(timeClimb.lm, which=5, add.smooth=FALSE, ps=9, sub.caption="",  
 cex.caption=1.1, fg="gray")  
 ## The points can alternatively be plotted using  
 ## plot(hatvalues(model.matrix(timeClimb.lm)), residuals(timeClimb.lm))

## Residuals versus leverages  
plot(timeClimb.lm, which=5, add.smooth=FALSE)  
## The points can alternatively be plotted using  
## plot(hatvalues(model.matrix(timeClimb.lm)), residuals(timeClimb.lm))

## This code is designed to be evaluated separately from other chunks  
with(nihills, scatter3d(x=log(dist), y=log(climb), z=log(time), grid=FALSE,  
 point.col="black", surface.col="gray60",  
 surface.alpha=0.2, axis.scales=FALSE))  
with(nihills, Identify3d(x=log(dist), y=log(climb), z=log(time),  
 labels=row.names(DAAG::nihills), minlength=8), offset=0.05)  
## To rotate display, hold down the left mouse button and move the mouse.  
## To put labels on points, right-click and drag a box around them, perhaps  
## repeatedly. Create an empty box to exit from point identification mode.

##### Influence on the regression coefficients

## Residuals versus leverages  
nihills <- DAAG::nihills  
timeClimb.lm <- lm(log(time) ~ log(dist) + log(climb), data = nihills)  
plot(timeClimb.lm, which=5, add.smooth=FALSE, ps=9, sub.caption="",  
 cex.caption=1.1, fg="gray")  
 ## The points can alternatively be plotted using  
 ## plot(hatvalues(model.matrix(timeClimb.lm)), residuals(timeClimb.lm))

##### \*Additional diagnostic plots

## As an indication of what is available, try  
car::influencePlot(allbacks.lm)

### Section 3.5 Assessment and comparison of regression models

#### Subsection 3.5.1: \*AIC, AICc, BIC, and Bayes Factors for normal theory regression models

## Calculations using mouse brain weight data  
mouse.lm <- lm(brainwt ~ lsize+bodywt, data=DAAG::litters)  
mouse0.lm <- update(mouse.lm, formula = . ~ . - lsize)

aicc <- sapply(list(mouse0.lm, mouse.lm), AICcmodavg::AICc)  
infstats <- cbind(AIC(mouse0.lm, mouse.lm), AICc=aicc,  
 BIC=BIC(mouse0.lm, mouse.lm)[,-1])  
print(rbind(infstats, "Difference"=apply(infstats,2,diff)), digits=3)

library(lattice)  
df <- data.frame(n=5:35, AIC=rep(2,31), BIC=log(5:35))  
cfAICc <- function(n,p,d) 2\*(p+d)\*n/(n-(p+d)-1) - 2\*p\*n/(n-p-1)  
df <- cbind(df, AICc12=cfAICc(5:35,1,1), AICc34=cfAICc(5:35,3,1))  
labs <- sort(c(2^(0:6),2^(0:6)\*1.5))  
xyplot(AICc12+AICc34+AIC+BIC ~ n, data=df, type='l', auto.key=list(columns=4),  
 scales=list(y=list(log=T, at=labs, labels=paste(labs))),  
 par.settings=simpleTheme(lty=c(1,1:3), lwd=2, col=rep(c('gray','black'), c(1,3))))

##### The functions drop1() and add1()

## Obtain AIC or BIC using `drop1()` or `add1()`  
n <- nrow(DAAG::litters)  
drop1(mouse.lm, scope=~lsize) # AIC, with/without `lsize`  
drop1(mouse.lm, scope=~lsize, k=log(n)) # BIC, w/wo `lsize`  
add1(mouse0.lm, scope=~bodywt+lsize) # AIC, w/wo `lsize`, alternative

##### The use of Bayesfactor::lmBF to compare the two models

suppressPackageStartupMessages(library(BayesFactor))  
bf1 <- lmBF(brainwt ~ bodywt, data=DAAG::litters)  
bf2 <- lmBF(brainwt ~ bodywt+lsize, data=DAAG::litters)  
bf2/bf1

## Relative support statistics  
setNames(exp(-apply(infstats[,-1],2,diff)/2), c("AIC","AICc","BIC"))

#### Subsection 3.5.2: Using anova() to compare models — the ihills data

lognihr <- log(DAAG::nihills)  
lognihr <- setNames(log(nihr), paste0("log", names(nihr)))  
timeClimb.lm <- lm(logtime ~ logdist + logclimb, data = lognihr)  
timeClimb2.lm <- update(timeClimb.lm, formula = . ~ . + I(logdist^2))  
print(anova(timeClimb.lm, timeClimb2.lm, test="F"), digits=4)

print(anova(timeClimb.lm, timeClimb2.lm, test="Cp"), digits=3)  
## Compare with the AICc difference  
sapply(list(timeClimb.lm, timeClimb2.lm), AICcmodavg::AICc)

form1 <- update(formula(timeClimb.lm), ~ . + I(logdist^2) + logdist:logclimb)  
addcheck <- add1(timeClimb.lm, scope=form1, test="F")  
print(addcheck, digits=4)

#### Subsection 3.5.3: Training/test approaches, and cross-validation

## Check how well timeClimb.lm model predicts for hills2000 data  
timeClimb.lm <- lm(logtime ~ logdist + logclimb, data = lognihr)  
logscot <- log(subset(DAAG::hills2000,  
 !row.names(DAAG::hills2000)=="Caerketton"))  
names(logscot) <- paste0("log", names(hills2000))  
scotpred <- predict(timeClimb.lm, newdata=logscot, se=TRUE)  
trainVar <- summary(timeClimb.lm)[["sigma"]]^2  
trainDF <- summary(timeClimb.lm)[["df"]][2]  
mspe <- mean((logscot[,'logtime']-scotpred[['fit']])^2)  
mspeDF <- nrow(logscot)

pf(mspe/trainVar, mspeDF, trainDF, lower.tail=FALSE)

scot.lm <- lm(logtime ~ logdist+logclimb, data=logscot)  
signif(summary(scot.lm)[['sigma']]^2, 4)

#### Subsection 3.5.4: Further points and issues

##### Patterns in the diagnostic plots – are they more than hints?

{r 3\_18, eval=F|

##### What is the scatter about the fitted response

##### Model selection and tuning risks

##### Generalization to new contexts requires a random sample of contexts

##### What happens if we do not transform the hillrace data?

##### Are “errors in x” an issue?

### Section 3.6 Problems with many explanatory variables

#### Subsection 3.6.1: Variable selection issues

##### Variable selection – a simulation with random data

y <- rnorm(100)  
## Generate a 100 by 40 matrix of random normal data  
xx <- matrix(rnorm(4000), ncol = 40)  
dimnames(xx)<- list(NULL, paste("X",1:40, sep=""))

## ## Find the best fitting model. (The 'leaps' package must be installed.)  
xx.subsets <- leaps::regsubsets(xx, y, method = "exhaustive", nvmax = 3, nbest = 1)  
subvar <- summary(xx.subsets)$which[3,-1]  
best3.lm <- lm(y ~ -1+xx[, subvar])  
print(summary(best3.lm, corr = FALSE))

## DAAG::bestsetNoise(m=100, n=40)  
best3 <- capture.output(DAAG::bestsetNoise(m=100, n=40))  
cat(best3[9:14], sep='\n')

##### The extent of selection effects – a detailed simulation:

oldpar <- par(fg='gray20',col.axis='gray20',lwd=0.5,col.lab='gray20')  
set.seed(41)  
library(splines)  
DAAG::bsnVaryNvar(nvmax=3, nvar = 3:35, xlab="")  
mtext(side=1, line=1.75, "Number selected from")

##### Cross-validation that accounts for the variable selection process

##### \*Regularization approaches

#### Subsection 3.6.2: Multicollinearity

##### An example – compositional data

data(Coxite, package="compositions") # Places Coxite in the workspace  
 # NB: Proceed thus because `Coxite` is not exported from `compositions`  
coxite <- as.data.frame(Coxite)

oldpar <- par(fg='gray20',col.axis='gray20',lwd=0.5,col.lab='gray20', tcl=-0.25)  
panel.cor <- function(x, y, digits = 3, prefix = "", cex.cor=0.8, ...)  
{  
old.par <- par(usr = c(0, 1, 0, 1)); on.exit(par(old.par))  
r <- abs(cor(x, y))  
txt <- format(c(r, 0.123456789), digits = digits)[1]  
txt <- paste0(prefix, txt)  
if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)  
text(0.5, 0.5, txt, cex = cex.cor \* sqrt(r))  
}  
pairs(coxite, gap=0.4, col=adjustcolor("blue", alpha=0.9), upper.panel=panel.cor)  
par(oldpar)

coxiteAll.lm <- lm(porosity ~ A+B+C+D+E+depth, data=coxite)  
print(coef(summary(coxiteAll.lm)), digits=2)

coxiteAll.lm <- lm(porosity ~ A+B+C+D+E+depth, data=coxite)  
coxite.hat <- predict(coxiteAll.lm, interval="confidence")  
hat <- coxite.hat[,"fit"]  
plot(porosity ~ hat, data=coxite, fg="gray", type="n", xlab="Fitted values",  
ylab="Fitted values, with 95% CIs\n(Points are observed porosities)",  
tcl=-0.35)  
with(coxite, points(porosity ~ hat, cex=0.75, col="gray45"))  
lines(hat, hat, lwd=0.75)  
ord <- order(hat)  
sebar <- function(x, y1, y2, eps=0.15, lwd=0.75){  
lines(rep(x,2), c(y1,y2), lwd=lwd)  
lines(c(x-eps,x+eps), rep(y1,2), lwd=lwd)  
lines(c(x-eps,x+eps), rep(y2,2), lwd=lwd)  
}  
q <- ord[round(quantile(1:length(hat), (1:9)/10))]  
for(i in q)sebar(hat[i], coxite.hat[i,"lwr"], coxite.hat[i,"upr"])  
coxiteAll.lm <- lm(porosity ~ A+B+C+D+E+depth, data=coxite)  
coxite.hat <- predict(coxiteAll.lm, interval="confidence")  
hat <- coxite.hat[,"fit"]

## Pointwise confidence bounds can be obtained thus:  
hat <- predict(coxiteAll.lm, interval="confidence", level=0.95)

#### Subsection 3.6.3: The variance inflation factor (VIF)

print(DAAG::vif(lm(porosity ~ A+B+C+D+depth, data=coxite)), digits=2)

b <- leaps::regsubsets(porosity ~ ., data=coxite, nvmax=4, method='exhaustive')  
## The calculation fails for nvmax=5  
inOut <- summary(b)[["which"]]  
## Extract and print the coefficents for the four regressions  
dimnam <- list(rep("",4),c("Intercept", colnames(coxite)[-7]))  
cmat <- matrix(nrow=4, ncol=7, dimnames=dimnam)  
for(i in 1:4)cmat[i,inOut[i,]] <- signif(coef(b,id=1:4)[[i]],3)  
outMat <- cbind(cmat," "=rep(NA,4),  
as.matrix(as.data.frame(summary(b)[c("adjr2", "cp", "bic")])))  
print(signif(outMat,3),na.print="")

BC.lm <- lm(porosity ~ B+C, data=coxite)  
print(signif(coef(summary(BC.lm)), digits=3))  
car::vif(BC.lm)

## Diagnostic plots can be checked thus:  
plot(BC.lm, eval=xtras)

##### Numbers that do not quite add up

coxiteR <- coxite  
coxiteR[, 1:5] <- round(coxiteR[, 1:5])  
coxiteR.lm <- lm(porosity ~ ., data=coxiteR)  
print(coef(summary(coxiteR.lm)), digits=2)  
print(DAAG::vif(lm(porosity ~ .-E, data=coxiteR)), digits=2)

##### Remedies for multicollinearity

### Section 3.7 Errors in x

##### Measurement of dietary intake

##### Simulations of the effect of measurement error

gph <- DAAG::errorsINx(gpdiff=0, plotit=FALSE, timesSDx=(1:4)/2,  
 layout=c(5,1), print.summary=FALSE)[["gph"]]  
parset <- DAAG::DAAGtheme(color=FALSE, alpha=0.6, lwd=2,  
 col.points=c("gray50","black"),  
 col.line=c("gray50","black"), lty=1:2)  
update(gph, par.settings=parset)

##### \*Two explanatory variables

##### Two explanatory variables, one measured without error – a simulation

gph <- DAAG::errorsINx(gpdiff=1.5, timesSDx=(1:2)\*0.8, layout=c(3,1),  
print.summary=FALSE, plotit=FALSE)[["gph"]]  
parset <- DAAG::DAAGtheme(color=FALSE, alpha=0.6, lwd=2,  
 col.points=c("gray50","black"),  
 col.line=c("gray50","black"), lty=1:2)  
update(gph, par.settings=parset)

##### An arbitrary number of variables

##### \*The classical error model versus the Berkson error model

##### Using missing value approaches to address measurement error

### Section 3.8 Multiple regression models – additional points

coef(lm(area ~ volume + weight, data=allbacks))  
b <- as.vector(coef(lm(weight ~ volume + area, data=allbacks)))  
c("\_Intercept\_"=-b[1]/b[3], volume=-b[2]/b[3], weight=1/b[3])

##### Unintended correlations

#### Subsection 3.8.2: Missing explanatory variables

gaba <- DAAG::gaba  
gabalong <- stack(gaba["30", -match('min', colnames(gaba))])  
gabalong$sex <- factor(rep(c("male", "female","all"), rep(2,3)),  
levels=c("female","male","all"))  
gabalong$treatment <- factor(rep(c("Baclofen","No baclofen"), 3),  
levels=c("No baclofen","Baclofen"))  
gph <- lattice::stripplot(sex~values, groups=treatment, data=gabalong,  
panel=function(x,y,...){  
lattice::panel.stripplot(x,y,...)  
lattice::ltext(x,y,paste(c(3,9,15,7,22,12)), pos=1, cex=0.8)  
}, auto.key=list(space="right", points=TRUE, cex=0.8))  
bw9 <- list(fontsize=list(text=9, points=5),  
cex=c(1.5,1.5), pch=c(1,16))  
update(gph, par.settings=parset,  
xlab=list("Average reduction: 30 min vs 0 min", cex=1.0),  
scales=list(cex=1.0, tck=0.35))

##### Strategies

#### Subsection 3.8.3: Added variable plots

yONx.lm <- lm(logtime ~ logclimb, data=lognihr)  
e\_yONx <- resid(yONx.lm)  
print(coef(yONx.lm), digits=4)

zONx.lm <- lm(logdist ~ logclimb, data=lognihr)  
e\_zONx <- resid(zONx.lm)  
print(coef(yONx.lm), digits=4)

ey\_xONez\_x.lm <- lm(e\_yONx ~ 0+e\_zONx)  
e\_yONxz <- resid(ey\_xONez\_x.lm)  
print(coef(ey\_xONez\_x.lm), digits=4)

oldpar <- par(fg='gray')  
## Code for added variable plots  
logtime.lm <- lm(logtime ~ logclimb+logdist, data=lognihr)  
car::avPlots(logtime.lm, lwd=1, terms="logdist", fg="gray")  
mtext(side=3, line=0.5, "A: Added var: 'logdist'", col="black", adj=0, cex=1.15)  
car::avPlots(logtime.lm, lwd=1, terms="logclimb", fg="gray")  
mtext(side=3, line=0.5, "B: Added var: 'logclimb'", col="black", adj=0, cex=1.15)  
par(oldpar)

## One call to show both plots  
car::avPlots(timeClimb.lm, terms=~.)

## Alternative code for first plot  
plot(e\_yONx ~ e\_zONx)

plot(yONx.lm, which=1, caption="", fg="gray")  
mtext(side=3, line=0.5, "A: From 'logtime' on 'logclimb'", adj=0, cex=0.85)  
plot(zONx.lm, which=1, caption="", fg="gray")  
mtext(side=3, line=0.5, "B: From 'logdist' on 'logclimb'", adj=0, cex=0.85)  
plot(ey\_xONez\_x.lm, which=1, caption="", fg="gray")  
mtext(side=3, line=0.5, "C: From AVP", adj=-0, cex=0.85)

##### Alternatives to Added Variable Plots

##### \*Algebraic details

ab1 <- coef(yONx.lm)  
ab2 <- coef(zONx.lm)  
b2 <- coef(ey\_xONez\_x.lm)  
b1 <- ab1[2] - b2\*ab2[2]  
a <- ab1[1] - b2\*ab2[1]

coef(lm(logtime ~ logclimb + logdist, data=lognihr))

#### Subsection 3.8.4: Nonlinear methods – an alternative to transformation?

nihr$climb.mi <- nihr$climb/5280  
nihr.nls0 <- nls(time ~ (dist^alpha)\*(climb.mi^beta), start =  
 c(alpha = 0.68, beta = 0.465), data = nihr)  
## plot(residuals(nihr.nls0) ~ log(predict(nihr.nls0)))

signif(coef(summary(nihr.nls0)),3)

nihr.nls <- nls(time ~ gamma + delta1\*dist^alpha + delta2\*climb.mi^beta,  
start=c(gamma = .045, delta1 = .09, alpha = 1,  
delta2=.9, beta = 1.65), data=nihr)  
## plot(residuals(nihr.nls) ~ log(predict(nihr.nls)))

signif(coef(summary(nihr.nls)),3)

### Section 3.9: Recap

### Section 3.10: Further reading

### Exercises (3.11)

3.1

## ## Set up factor that identifies the `have' cities  
cities <- DAAG::cities  
cities$have <- with(cities, factor(REGION %in% c("ON","WEST"),  
 labels=c("Have-not","Have")))

gphA <- lattice::xyplot(POP1996~POP1992, groups=have, data=cities,  
 auto.key=list(columns=2))  
gphB<-lattice::xyplot(log(POP1996)~log(POP1992), groups=have, data=cities,  
 auto.key=list(columns=2))  
print(gphA, split=c(1,1,2,1), more=TRUE)  
print(gphB, split=c(2,1,2,1))

cities.lm1 <- lm(POP1996 ~ have+POP1992, data=cities)  
cities.lm2 <- lm(log(POP1996) ~ have+log(POP1992), data=cities)

3.8a

nihills.lm <- lm(time ~ dist+climb, data=DAAG::nihills)  
nihillsX.lm <- lm(time ~ dist+climb+dist:climb, data=DAAG::nihills)  
anova(nihills.lm, nihillsX.lm) # Use `anova()` to make the comparison  
coef(summary(nihillsX.lm)) # Check coefficient for interaction term  
drop1(nihillsX.lm)

3.11

log(time) ~ log(dist) + log(climb) ## lm model  
time ~ alpha\*dist + beta\*I(climb^2) ## nls model

3.13

x1 <- runif(10) # predictor which will be missing  
x2 <- rbinom(10, 1, 1-x1)  
 ## observed predictor, depends on missing predictor  
y <- 5\*x1 + x2 + rnorm(10,sd=.1) # simulated model; coef of x2 is positive  
y.lm <- lm(y ~ factor(x2)) # model fitted to observed data  
coef(y.lm)  
y.lm2 <- lm(y ~ x1 + factor(x2)) # correct model  
coef(y.lm2)

3.16

bomData <- DAAG::bomregions2021  
nraw.lqs <- MASS::lqs(northRain ~ SOI + CO2, data=bomData)  
north.lqs <- MASS::lqs(I(northRain^(1/3)) ~ SOI + CO2, data=bomData)  
plot(residuals(nraw.lqs) ~ Year, data=bomData)  
plot(residuals(north.lqs) ~ Year, data=bomData)

3.17f

socpsych <- subset(DAAG::repPsych, Discipline=='Social')  
with(socpsych, scatter.smooth(T\_r.R~T\_r.O))  
abline(v=.5)

soc.rlm <- MASS::rlm(T\_r.R~T\_r.O, data=subset(socpsych, T\_r.O<=0.5))  
## Look at summary statistics  
termplot(soc.rlm, partial.resid=T, se=T)

plot(soc.rlm)

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch3.R")  
}

# 4. Chapter 4: Exploiting the linear model framework

##### Packages required (with dependencies)

DAAG effects mgcv splines scam MASS latticeExtra car WDI AICcmodavg ggplot2 kableExtra qgam patchwork

Additionally, Hmisc and knitr are required in order to process the Rmd source file.

Note the use of the ‘patchwork’ package to make it easy to place two ggplot2 plots side by side.

Hmisc::knitrSet(basename="exploit", lang='markdown', fig.path="figs/g", w=7, h=7)  
oldopt <- options(digits=4, width=70, scipen=999)  
library(knitr)  
## knitr::render\_listings()  
opts\_chunk[['set']](cache.path='cache-', out.width="80%", fig.align="center",   
 fig.show='hold', formatR.arrow=FALSE, ps=10,   
 strip.white = TRUE, comment=NA, width=70,   
 tidy.opts = list(replace.assign=FALSE))

### Section 4.1 Levels of a factor – using indicator variables

#### Subsection 4.1.1: Example – sugar weight

sugar <- DAAG::sugar # Copy dataset 'sugar' into the workspace  
## Ensure that "Control" is the first level  
sugar[["trt"]] <- relevel(sugar[["trt"]], ref="Control")  
options()[["contrasts"]] # Check the default factor contrasts  
## If your output does not agree with the above, then enter  
## options(contrasts=c("contr.treatment", "contr.poly"))

sugar.aov <- aov(weight ~ trt, data=sugar)  
## To display the model matrix, enter: model.matrix(sugar.aov)  
## Note the use of summary.lm(), not summary() or summary.aov()  
round(signif(coef(summary.lm(sugar.aov)), 3), 4)

sem <- summary.lm(sugar.aov)$sigma/sqrt(3) # 3 results/trt  
# Alternatively, sem <- 6.33/sqrt(2)  
qtukey(p=.95, nmeans=4, df=8) \* sem

#### Subsection 4.1.2: Different choices for the model matrix when there are factors

contrasts(sugar$trt) <- 'contr.sum'  
sugarSum.aov <- aov(weight ~ trt, data = sugar)  
round(signif(coef(summary.lm(sugarSum.aov)), 3),4)

dummy.coef(sugarSum.aov)

##### Factor contrasts – further details

contrasts(sugar$trt) <- "contr.sum"

fish <- factor(1:3, labels=c("Trout","Cod","Perch"))

contr.treatment(fish)  
# Base is "Trout"

contr.SAS(fish)  
# Base is "Perch"

contr.sum(fish)  
# Base is mean of levels

##### \*Tests for main effects in the presence of interactions?

### Section 4.2 Block designs and balanced incomplete block designs

#### Subsection 4.2.1: Analysis of the rice data, allowing for block effects

rice <- DAAG::rice  
ricebl.aov <- aov(ShootDryMass ~ Block + variety \* fert, data=rice)  
print(summary(ricebl.aov), digits=3)

round(signif(coef(summary.lm(ricebl.aov)), 3), 5)  
with(summary.lm(ricebl.aov),  
cat("Residual standard error: ", sigma, "on", df[2], "degrees of freedom"))

## AOV calculations, ignoring block effects  
rice.aov <- aov(ShootDryMass ~ variety \* fert, data=rice)  
summary.lm(rice.aov)$sigma

ricebl.aov <- aov(ShootDryMass ~ factor(Block) + variety \* fert, data=rice)

model.tables(ricebl.aov, type="means", se=TRUE, cterms="variety:fert")

#### Subsection 4.2.2: A balanced incomplete block design

appletaste <- DAAG::appletaste  
with(appletaste, table(product, panelist))

sapply(appletaste, is.factor) # panelist & product are factors  
appletaste.aov <- aov(aftertaste ~ product + panelist, data=appletaste)  
summary(appletaste.aov)

as.data.frame(effects::Effect("product", appletaste.aov, confidence.level=0.95))

## NB that 'product' was first term in the model formula  
## Thus, the 1st 4 coefficients have the information required  
coef(summary.lm(appletaste.aov))[1:4, ]

### Section 4.3 Fitting multiple lines

## Fit various models to columns of data frame leaftemp (DAAG)  
leaftemp <- DAAG::leaftemp  
leaf.lm1 <- lm(tempDiff ~ 1 , data = leaftemp)  
leaf.lm2 <- lm(tempDiff ~ vapPress, data = leaftemp)  
leaf.lm3 <- lm(tempDiff ~ CO2level + vapPress, data = leaftemp)  
leaf.lm4 <- lm(tempDiff ~ CO2level + vapPress +  
 vapPress:CO2level, data = leaftemp)

anova(leaf.lm1, leaf.lm2, leaf.lm3, leaf.lm4)

print(coef(summary(leaf.lm3)), digits=2)

### Section 4.4 Methods for fitting smooth curves

#### Subsection 4.4.1: Polynomial Regression

seedrates <- DAAG::seedrates  
form2 <- grain ~ rate + I(rate^2)  
# Without the wrapper function I(), rate^2 would be interpreted  
# as the model formula term rate:rate, and hence as rate.  
quad.lm2 <- lm(form2, data = seedrates)  
## Alternative, using gam()  
## quad.gam <- mgcv::gam(form2, data = seedrates)

suppressPackageStartupMessages(library(ggplot2))

## Use ggplot2 functions to plot points, line, curve, & 95% CIs  
## library(ggplot2)  
gph <- ggplot(DAAG::seedrates, aes(rate,grain))+  
 geom\_point(aes(size=3), color='magenta')+xlim(c(25,185))  
colors <- c("Linear"="blue", "Quadratic"="red")  
ggdat <- ggplot\_build(gph+geom\_smooth(aes(rate,grain,color="Linear"),  
 method=lm, formula=y~poly(x,2),fullrange=TRUE))$data[[2]]  
gph1 <- gph+geom\_smooth(aes(color="Linear"), method=lm, formula=y~x, fullrange=TRUE, fill='dodgerblue')  
gph1 + geom\_line(data = ggdat, aes(x = x, y = y, color="Quadratic"),  
 linewidth=0.75)+  
 geom\_ribbon(data=ggdat, aes(x=x,y=y, ymin=ymin, ymax=ymax,  
 color="Quadratic"), linewidth=0.75,  
 fill=NA, linetype=2, outline.type='both', show.legend=FALSE) +  
 scale\_color\_manual(values=colors, aesthetics = "color")+  
 theme(legend.position=c(.8,.78)) +  
 coord\_cartesian(expand=FALSE) + xlab("Seeding rate (kg/ha)") +  
 ylab("Grains per head") + labs(color="Model") +  
 guides(size='none',  
 color = guide\_legend(override.aes = list(fill="transparent") ) )  
## detach("package:ggplot2")

quad.lm2 <- lm(grain ~ rate + I(rate^2), data = DAAG::seedrates)  
print(coef(summary(quad.lm2)), digits=2)  
cat("\nCorrelation matrix\n")  
print(summary(quad.lm2, corr=TRUE)$correlation, digits=2)

##### \*An alternative formulation using orthogonal polynomials

seedratesP.lm2 <- lm(grain ~ poly(rate,2), data = seedrates)  
print(coef(summary(seedratesP.lm2)), digits=2)

## Alternative, using mgcv::gam()  
seedratesP.gam <- mgcv::gam(grain ~ poly(rate,2), data = seedrates)

logseed.lm <- lm(log(grain) ~ log(rate), data=DAAG::seedrates)  
coef(summary(logseed.lm))

## Use ggplot2 functions to plot points, line, curve, & 95% CIs  
## library(ggplot2)  
gph <- ggplot(DAAG::seedrates, aes(rate,grain)) +  
 geom\_point(size=3, color="magenta")+xlim(c(25,185))  
colors <- c("Loglinear"="gray40", "Quadratic"="red")  
ggdat <- ggplot\_build(gph+geom\_smooth(method=lm, formula=y~poly(x,2),  
 fullrange=TRUE))$data[[2]]  
ggln <- ggplot\_build(gph+geom\_smooth(method=lm,  
 formula=log(y)~log(x),fullrange=TRUE))$data[[2]]  
## Assign to gphA rather than (as in text) plotting at this point  
gphA <- gph + geom\_line(data = ggdat, aes(x = x, y = y, color="Quadratic"),  
 linewidth=0.75) +  
geom\_ribbon(data=ggdat, aes(x=x,y=y, ymin=ymin, ymax=ymax, color="Quadratic"),  
 linewidth=0.75, fill=NA, linetype=2, outline.type='both',  
 show.legend=FALSE) +  
geom\_line(data = ggln, aes(x = x, y = exp(y), color="Loglinear"),  
 linewidth = 0.75) +  
geom\_ribbon(data=ggln, aes(x=x,y=exp(y), ymin=exp(ymin), ymax=exp(ymax),  
 color="Loglinear"), fill=NA, linewidth=0.75, linetype=3,  
 outline.type='both', show.legend=FALSE)+  
 scale\_color\_manual(values=colors, aesthetics = "color")+  
 coord\_cartesian(expand=FALSE) +  
 xlab("Seeding rate (kg/ha)") + ylab("Grains per head") +  
 labs(title="A: Loglinear fit vs quadratic fit", color="Model") +  
 guides(size='none',  
 color = guide\_legend(override.aes = list(fill="transparent") ) ) +  
 theme(legend.position=c(.8,.78))  
df <- data.frame(rate=rep(DAAG::seedrates$rate,2), res=c(resid(logseed.lm),  
 log(DAAG::seedrates$grain)-log(fitted(quad.lm2))),  
 Model=rep(c("Loglinear","Quadratic"),rep(nrow(DAAG::seedrates),2)))  
## Assign to gphB rather than (as in text) plotting at this point  
gphB <- ggplot(df, aes(x=rate, y=res, shape=Model,color=Model))+  
geom\_point(size=2.5) + scale\_color\_manual(values=colors) +  
xlab("Seeding rate (kg/ha)") + ylab("Residuals on log scale") +  
labs(title="B: Residuals") +  
 guides(size='none',  
 color = guide\_legend(override.aes = list(fill="transparent") ) ) +  
 theme(legend.position=c(.8,.78))  
## Now take advantage of the magic of the 'patchwork' package  
library(patchwork)  
gphA+gphB  
## detach("package:ggplot2")

aic <- AIC(quad.lm2, logseed.lm)  
aic["logseed.lm",2] <- aic["logseed.lm",2] + sum(2\*log(seedrates$grain))  
round(aic,1)

seedrates<-DAAG::seedrates  
quad.lm2 <- lm(grain ~ poly(rate,degree=2), data=seedrates)  
ns.lm2 <- lm(grain ~ splines::ns(rate,df=2), data=seedrates)  
tps.gam2 <- mgcv::gam(grain ~ s(rate, k=3, fx=T), data=seedrates)

mflist <- lapply(list(quad=quad.lm2, nsplines=ns.lm2, tps=tps.gam2), model.matrix)  
mftab <- with(mflist, cbind(quad, nsplines, tps))  
colnames(mftab) <- c("(Int)", "poly2.1", "poly2.2", "(Int)", "ns2.1", "ns2.2", "(Int)", "s3.1", "s3.2")  
library(kableExtra)  
linesep = c('', '', '', '\\addlinespace')  
kbl(mftab, booktabs=TRUE, format='latex', toprule=FALSE,  
format.args=list(justify="right", width=8)) |>  
kable\_styling(latex\_options = c("scale\_down",latex\_options = "hold\_position"), position='center') |>  
add\_header\_above(c('poly(rate,2)' = 3, 'splines::ns(rate,df=2)' = 3, 's(rate, k=3, fx=T)' = 3),  
align='c', monospace=rep(T,3))|>  
add\_header\_above(c('lm: grain~' = 3, 'lm: grain~'=3, 'gam: grain~'=3),  
 align='c', monospace=rep(T,3), line=F)

##### GAM models versus models fitted using lm()

##### Alternative fits – what is the best choice?

## Load required packages  
suppressPackageStartupMessages(library(splines))  
suppressPackageStartupMessages(library(mgcv))

ohms.tp <- gam(kohms~s(juice, bs="tp"), data=fruitohms)  
ohms.cs <- gam(kohms~s(juice, bs="cs"), data=fruitohms)  
range(fitted(ohms.tp)-fitted(ohms.cs))

summary(ohms.tp)

summary(ohms.tpBIC)

#### Subsection 4.4.3: The contributions of basis curves to the fit

#### Subsection 4.4.4: Checks on the fitted model

## Printed output from `gam.check(ohms.tpBIC)`  
cat(out, sep="\n")

#### Subsection 4.3.5: Monotone curves

ohms.scam <- scam::scam(kohms ~ s(juice,bs="mpd"), data=fruitohms)  
summary(ohms.scam)

AIC(ohms.scam, ohms.tp)

BIC(ohms.scam, ohms.tp)

#### Subsection 4.4.6: Different smooths for different levels of a factor

whiteside <- MASS::whiteside  
gas.gam <- gam(Gas ~ Insul+s(Temp, by=Insul), data=whiteside)

summary(gas.gam)

Box.test(resid(gas.gam)[whiteside$Insul=='Before'], lag=1)  
Box.test(resid(gas.gam)[whiteside$Insul=='After'], lag=1)

#### Subsection 4.4.7: The remarkable reach of mgcv and related packages

##### Departures from independence assumptions

#### Subsection 4.4.8: Multiple spline smoothing terms — dewpoint data

## GAM model -- `dewpoint` data  
dewpoint <- DAAG::dewpoint  
ds.gam <- gam(dewpt ~ s(mintemp) + s(maxtemp), data=dewpoint)  
plot(ds.gam, resid=TRUE, pch=".", se=2, cex=2, fg="gray")

##### Using residuals as a check for non-additive effects

library(lattice)  
## Residuals vs maxtemp, for different mintemp ranges  
mintempRange <- equal.count(dewpoint$mintemp, number=3)  
ds.xy <- xyplot(residuals(ds.gam) ~ maxtemp|mintempRange, data=dewpoint,  
 layout=c(3,1), scales=list(tck=0.5), aspect=1, cex=0.65,  
 par.strip.text=list(cex=0.75), type=c("p","smooth"),  
 xlab="Maximum temperature", ylab="Residual")  
ds.xy

##### \*A smooth surface

## Fit surface  
ds.tp <- gam(dewpt ~ s(mintemp, maxtemp), data=DAAG::dewpoint)  
vis.gam(ds.tp, plot.type="contour") # gives a contour plot of the  
# fitted regression surface  
vis.gam(ds.gam, plot.type="contour") # cf, model with 2 smooth terms

#### Subsection 4.4.9: Atlantic hurricanes that made landfall in the US

hurricNamed <- DAAG::hurricNamed  
hurricS.gam <- gam(car::yjPower(deaths, lambda=-0.2) ~  
 s(log(BaseDam2014)) + s(LF.PressureMB),  
 data=hurricNamed, method="ML")  
anova(hurricS.gam)

plot(hurricS.gam, resid=TRUE, pch=16, cex=0.5, select=1, fg="gray")  
mtext(side=3, line=1, "A: Term in log(BaseDam2014)", cex=1.0, adj=0, at=-3.75)  
plot(hurricS.gam, resid=TRUE, pch=16, cex=0.5, select=2, fg="gray")  
mtext(side=3, line=1, "B: Term in LF.PressureMB", cex=1.0, adj=0, at=878)  
qqnorm(resid(hurricS.gam), main="", fg="gray")  
mtext(side=3, line=1, "C: Q-Q plot of residuals", cex=1.0, adj=0, at=-4.25)

##### An explanatory variable with an overly long-tailed distribution

hurricSlog1.gam <- gam(log(deaths+1) ~ s(log(BaseDam2014)), data=hurricNamed)  
hurricSlog2.gam <- gam(log(deaths+1) ~ s(BaseDam2014), data=hurricNamed)

plot(hurricSlog1.gam, resid=TRUE, pch=16, cex=0.5, adj=0, fg="gray")  
mtext(side=3, "A: Use log(BaseDam2014)", cex=1.4, adj=0, line=1, at=-3.15)  
plot(hurricSlog2.gam, resid=TRUE, pch=16, cex=0.5, fg="gray")  
mtext(side=3, "B: Use BaseDam2014", cex=1.4, adj=0, line=1, at=-28500)

#### Subsection 4.4.10: Other smoothing methods

### Section 4.5 Quantile regression

## If necessary, install the 'WDI' package & download data  
if(!file.exists("wdi.RData")){  
 if(!is.element("WDI", installed.packages()[,1]) )install.packages("WDI")  
inds <- c('SP.DYN.TFRT.IN','SP.DYN.LE00.IN', 'SP.POP.TOTL')  
indnams <- c("FertilityRate", "LifeExpectancy", "population")  
wdi2020 <- WDI::WDI(country="all", indicator=inds, start=2020, end=2020,  
 extra=TRUE)  
wdi2020 <- na.omit(droplevels(subset(wdi2020, !region %in% "Aggregates")))  
wdi <- setNames(wdi2020[order(wdi2020[, inds[1]]),inds], indnams)  
save(wdi, file="wdi.RData")  
}

##### 2020 World Bank data on fertility and life expectancy

load("wdi.RData") # Needs `wdi.RData` in working directory; see footnote  
library(qgam)  
wdi[, "ppop"] <- with(wdi, population/sum(population))  
wdi[,"logFert"] <- log(wdi[,"FertilityRate"])  
form <- LifeExpectancy ~ s(logFert)  
## Panel A model  
fit.qgam <- qgam(form, data=wdi, qu=.5)  
## Panel B: Multiple (10%, 90% quantiles; unweighted, then weighted  
fit19.mqgam <- mqgam(form, data=wdi, qu=c(.1,.9))  
wtd19.mqgam <- mqgam(form, data=wdi, qu=c(.1,.9),  
 argGam=list(weights=wdi[["ppop"]]))

hat50 <- cbind(LifeExpectancy=wdi[, "LifeExpectancy"], logFert=wdi[,"logFert"],  
 as.data.frame(predict(fit.qgam, se=T)))  
hat50 <- within(hat50, {lo <- fit-2\*se.fit; hi <- fit+2\*se.fit})  
hat19 <- as.data.frame(matrix(nrow=nrow(wdi), ncol=4))  
for(i in 1:2){hat19[[i]] <- qdo(fit19.mqgam, c(.1,.9)[i], predict)  
 hat19[[i+2]] <- qdo(wtd19.mqgam, c(.1,.9)[i], predict) }  
 ## NB, can replace `predict` by `plot`, or `summary`  
colnames(hat19) <- c(paste0(rep(c('q','qwt'),c(2,2)), rep(c('10','90'),2)))  
hat19 <- cbind(hat19, logFert=wdi[,"logFert"])

## Panel A: Fit with SE limits, 50% quantile  
gphA <- xyplot(lo+fit+hi~logFert, data=hat50, lty=c(2,1,2),lwd=1.5,type='l') +  
 latticeExtra::as.layer(xyplot(LifeExpectancy~logFert,  
 data=hat50, pch='.', cex=2))  
## Panel B: Multiple quantiles; unweighted and weighted fits  
gph19 <- xyplot(q10+q90+qwt10+qwt90 ~ logFert, type="l",  
 data=hat19, lty=rep(1:2,c(2,2)),lwd=1.5)  
gphB <- xyplot(LifeExpectancy ~ logFert, data=wdi) + as.layer(gph19)  
update(c("A: 50% curve, 2 SE limits"=gphA, "B: 0.1, 0.9 quantiles"=gphB,  
 x.same=T, y.same=T), between=list(x=0.5),  
 xlab="Fertility Rate", ylab="Life Expectancy",  
 scales=list(x=list(at=log(2^((0:5)/2)), labels=round(2^((0:5)/2),1)),  
 alternating=F),  
 par.settings=DAAG::DAAGtheme(color=F, col='gray50', cex=2, pch='.'))

## Plots for the individual quantiles can be obtained thus:  
## ## Panel A  
plot(fit.qgam, shift=mean(predict(fit.qgam)))  
## Panel B, 10% quantile  
fitm10 <- qdo(fit19.mqgam, qu=0.1)  
plot(fitm10, resid=T, shift=mean(predict(fitm10)),  
 ylim=range(wdi$LifeExpectancy), cex=2)  
wfitm10 <- qdo(wtd19.mqgam, qu=0.1)  
plot(wfitm10, resid=T, shift=mean(predict(wfitm10)),  
 ylim=range(wdi$LifeExpectancy), cex=2)

### Section 4.6: Further reading and remarks

### Exercises (4.7)

4.2

roller.lm <- lm(depression~weight, data=DAAG::roller)  
roller.lm2 <- lm(depression~weight+I(weight^2), data=DAAG::roller)

4.4

toycars <- DAAG::toycars  
lattice::xyplot(distance ~ angle, groups=factor(car), type=c('p','r'),  
 data=toycars, auto.key=list(columns=3))

4.4a

parLines.lm <- lm(distance ~ 0+factor(car)+angle, data=toycars)  
sepLines.lm <- lm(distance ~ factor(car)/angle, data=toycars)

4.4b

sepPol3.lm <- lm(distance ~ factor(car)/angle+poly(angle,3)[,2:3], data=toycars)

4.4c

sapply(list(parLines.lm, sepLines.lm, sepPol3.lm), AICcmodavg::AICc)

4.4e

setNames(sapply(list(parLines.lm, sepLines.lm, sepPol3.lm),  
 function(x)summary(x)$adj.r.squared), c("parLines","sepLines","sepPol3"))

4,7

seedrates.lm <- lm(grain ~ rate + I(rate^2), data=seedrates)  
seedrates.pol <- lm(grain ~ poly(rate,2), data=seedrates)

4.10a

geo.gam <- gam(thickness ~ s(distance), data=DAAG::geophones)

4.11

plot(DAAG::geophones$distance, acf(resid(geo.gam), lag.max=55)$acf)  
Box.test(resid(geo.gam), lag=10)  
Box.test(resid(geo.gam), lag=20)  
Box.test(resid(geo.gam), lag=20, type="Ljung")

4.15

library(mgcv)  
xy <- data.frame(x=1:200, y=arima.sim(list(ar=0.75), n=200))  
df.gam <- gam(y ~ s(x), data=xy)  
plot(df.gam, residuals=TRUE)

4.16

library(mgcViz)  
ohms.tpBIC <- gam(kohms ~ s(juice, bs="tp"), data=fruitohms,   
 gamma=log(nrow(fruitohms))/2, method="REML")  
ohms.gamViz <- mgcViz::getViz(ohms.tpBIC) # Convert to a `gamViz` object   
g1 <- plot(sm(ohms.gamViz, 1)) # Graphics object for term 1 (of 1)  
g1 + l\_fitLine(colour = "red") + l\_rug(mapping = aes(x=x, y=y), alpha = 0.4) +  
 l\_ciLine(mul = 2, colour = "blue", linetype = 2) + # Multiply SE by `mul`  
 l\_points(shape = 19, size = 1, alpha = 0.5)

4.16a

plot(sm(ohms.gamViz, 1), nsim = 20) + l\_ciLine() + l\_fitLine() + l\_simLine()

4.16b

gam(Gas ~ Insul+s(Temp, by=Insul), data=whiteside) |>   
 getViz() -> gas.gamViz  
plot(sm(gas.gamViz,1), nsim = 20) + l\_ciLine() + l\_fitLine() + l\_simLine()

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch4.R")  
}

# 5. Chapter 5: Generalized linear models and survival analysis

##### Packages required (with dependencies)

DAAG car mgcv colorspace HistData gamlss dplyr tidyr MASS ggplot2 latticeExtra qgam VGAM survival HistData

Additionally, knitr is required in order to process the Rmd source file.

### Section 5.1 Generalized linear models

#### Subsection 5.1.1: Linking the expected value to the covariate

## Simplified plot showing the logit link function  
p <- (1:39)/40  
logitp <- log(p/(1 - p))  
plot(p, logitp, xlab = "Proportion", ylab = "logit(p)", type = "l", pch = 1)

par(las=0)  
p <- seq(from=1, to=99, by=1)/100; n<- 150; eps=0.001  
gitp <- log(p/(1 - p))  
plot(p, gitp, xlab = "", ylab = "", type = "l", pch = 1,  
las=1, xlim=0:1, xaxs="i", fg="gray")  
mtext(side = 1, line = 1.75, expression("Proportion "\*pi))  
mtext(side = 2, line = 1.75,  
expression("logit("\*pi\*") = log(Odds)"))  
mtext(side = 3, line = 0.5, "A: Logit link", adj=0, cex=1.0)  
pval <- c(0.001, 0.01, 0.1, 0.5, 0.9, 0.99, 0.999)  
par(mgp = c(2.5, 0.5, 0))  
## axis(1, at=c(0,1), lwd=0, labels=c(0,1), xpd=TRUE)  
axis(4, adj=0.075, at = log(pval/(1 - pval)), las=1,  
col="gray", labels = paste(pval), lwd=0, lwd.ticks=1)  
seP <- sqrt(p\*(1-p)/100)  
plot(p, seP, xlab = "", ylab = "", type = "l", pch = 1,  
las=1, xlim=0:1, xaxs="i", fg="gray")  
## axis(1, at=c(0,1), lwd=0, labels=c(0,1), xpd=TRUE)  
mtext(side = 1, line = 1.75, expression("Proportion "\*pi))  
mtext(side = 2, line = 2.25, expression("SD["\*p\*"], "\*n\*"=100"))  
mtext(side = 3, line = 0.5,  
expression("B: SD["\*p\*"], "\*n\*"=100"), adj=0, cex=1.0)  
seLP <- (p\*(1-p)/n)\*((p+eps)\*(1-p+eps))^-2  
plot(p, seLP, xlab = "", ylab = "", type = "l", pch = 1,  
las=1, xlim=0:1, xaxs="i", fg="gray")  
## axis(1, at=c(0,1), lwd=0, labels=c(0,1), xpd=TRUE)  
mtext(side = 1, line = 1.75, expression("Proportion "\*pi))  
mtext(side = 2, line = 1.75, expression("SD[logit("\*p\*")], "\*n\*"=100"))  
mtext(side = 3, line = 0.5, "C: SD[logit(p)]", adj=0, cex=1.0)

#### Subsection 5.1.2: Noise terms need not be normal

#### Subsection 5.1.3: Variation that is greater than binomial or Poisson

##### Least squares versus logistic regression

#### Subsection 5.1.4: Log odds in contingency tables

#### Subsection 5.1.5: Logistic regression with a continuous explanatory variable

anestot <- aggregate(DAAG::anesthetic[, c("move","nomove")],  
by=list(conc=DAAG::anesthetic$conc), FUN=sum)  
## The column 'conc', because from the 'by' list, is then a factor.  
## The next line recovers the numeric values  
anestot$conc <- as.numeric(as.character(anestot[["conc"]]))  
anestot$total <- apply(anestot[, c("move","nomove")], 1 , sum)  
anestot$prop <- anestot$nomove/anestot$total

par(mgp=c(2.5,.5,0))  
anesthetic <- DAAG::anesthetic  
z <- table(anesthetic$nomove, anesthetic$conc)  
tot <- apply(z, 2, sum)  
prop <- z[2, ]/(tot)  
oprop <- sum(z[2, ])/sum(tot)  
conc <- as.numeric(dimnames(z)[[2]])  
par(las=0)  
plot(conc, prop, xlab = "Concentration", ylab = "Proportion",  
 xlim=c(0.5, 2.5), ylim = c(0, 1), pch = 16, axes=F)  
axis(1, cex=0.9, lwd=0, lwd.ticks=1)  
axis(2, at=c(0, 0.5, 1.0), cex=0.9, lwd=0, lwd.ticks=1)  
axis(2, at=c(0.25, 0.75), cex=0.9, lwd=0, lwd.ticks=1)  
box(col="gray")  
chh <- par()$cxy[2]  
chw <- par()$cxy[1]  
text(conc - 0.3 \* chw, prop-sign(prop-0.5)\*chh/4, paste(tot),  
adj = 1, cex=0.65)  
abline(h = oprop, lty = 2)

## Fit model directly to the 0/1 data in nomove  
anes.glm <- glm(nomove ~ conc, family=binomial(link="logit"),  
 data=DAAG::anesthetic)  
## Fit model to the proportions; supply total numbers as weights  
anes1.logit <- glm(prop ~ conc, family=binomial(link="logit"),  
 weights=total, data=anestot)

DAAG::sumry(anes.glm, digits=2)

##### A note on model output

## Tp get coefficients, SEs, and associated statistics, specify:  
print(coef(summary(anes.glm)), digits=2)  
## Get full default output  
summary(anes.glm, digits=2)

### Section 5.2 Logistic multiple regression

frogs <- DAAG::frogs

## Presence/absence information: data frame frogs (DAAGS)  
suppressMessages(library(ggplot2))  
p <- ggplot(frogs, aes(easting, northing)) +  
 geom\_point(size=3, alpha=0.25) + coord\_fixed() +  
 xlab("Meters east of reference point")+ylab("Meters north") +  
 theme(axis.title=element\_text(size=11), axis.text=element\_text(size=8))  
p + geom\_point(data=subset(frogs, pres.abs==1),  
 aes(easting, northing), alpha=1, shape=3, col="white", size=1.5)

frogs <- within(frogs, {maxSubmin <- meanmax-meanmin  
 maxAddmin <- meanmax+meanmin})

##### Implications for the Variance Inflation Factor

#### Subsection 5.2.1: Choose explanatory terms, and fit model

## Find power transformations  
useCols <- c('distance','NoOfPools','NoOfSites','avrain','maxAddmin','maxSubmin')  
tfrogs <- car::powerTransform(frogs[,useCols], family="yjPower")  
## Create, for later use, a matrix with variables transformed as suggested  
transY <- car::yjPower(frogs[,useCols], coef(tfrogs, round=TRUE))  
summary(tfrogs, digits=2)

frogs0.glm <- glm(formula = pres.abs ~ log(distance) + log(NoOfPools)+  
 sqrt(NoOfSites) + avrain + maxAddmin + maxSubmin,  
 family = binomial, data = frogs)  
DAAG::sumry(frogs0.glm, digits=1)

## Check effect of omitting sqrt(NoOfSites) and avrain from the model  
## ~ . takes the existing formula. Precede terms to be  
## omitted by '-'. (For additions, precede with '+')  
frogs.glm <- update(frogs0.glm, ~ . -sqrt(NoOfSites)-avrain)  
frogsAlt.glm <- update(frogs.glm, ~ . -maxAddmin+altitude)  
AIC(frogs0.glm, frogs.glm,frogsAlt.glm)

rbind(  
'frogs0.glm'=coef(frogs0.glm)[c('log(distance)','log(NoOfPools)','maxAddmin','maxSubmin')],  
'frogs.glm'=coef(frogs.glm)[c('log(distance)','log(NoOfPools)','maxAddmin','maxSubmin')]  
)  
coef(frogsAlt.glm)[c('log(distance)','log(NoOfPools)','altitude','maxSubmin')]

coef(summary(frogs.glm))

#### Subsection 5.2.2: Fitted values

## Use of `predict()` and `fitted()` --- examples  
fitted(frogs.glm) # Fitted values' scale of response  
predict(frogs.glm, type="response") # Same as fitted(frogs.glm)  
predict(frogs.glm, type="link") # Scale of linear predictor  
## For approximate SEs, specify  
predict(frogs.glm, type="link", se.fit=TRUE)

library(ggplot2)  
frogs$Prob. <- fitted(frogs.glm)  
frogs$presAbs <- factor(frogs$pres.abs)  
p <- ggplot(frogs, aes(easting, northing, color=Prob.)) +  
 geom\_point(size=2, alpha=0.5) + coord\_fixed() +  
 xlab("Meters east of reference point")+ylab("Meters north") +  
 theme(axis.title=element\_text(size=9), axis.text=element\_text(size=6))  
p2 <- p+scale\_color\_gradientn(colours=colorspace::heat\_hcl(10,h=c(0,-100),  
 l=c(75,40), c=c(40,80), power=1)) +  
 guides(fill=guide\_legend(title=NULL))  
p2 + geom\_point(data=subset(frogs, presAbs==1),  
 aes(easting, northing), alpha=1, shape=3, col="white", size=1)

#### Subsection 5.2.3: Plots that show the contributions of explanatory variables

opar <- par(mgp=c(2.1,.4,0), mfrow=c(1,3))  
Cholera <- HistData::Cholera  
fitP2.glm <- glm(cholera\_deaths ~ offset(log(popn)) + water +  
 log(elevation+3) + poly(poor\_rate,2) +I(elevation==350),  
 data=Cholera, family=quasipoisson)  
Cholera[["water"]] <- factor(Cholera[["water"]], labels=c("Battersea",  
 "NewRiver","Kew"))  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
ylabs=rep("Partial residual",3), terms='water', fg="gray")  
axis(1, at=2, labels="NewRiver", lwd=0, line=0.75)  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
 ylabs=rep("Partial residual",3), terms='log(elevation + 3)', fg="gray")  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
 ylabs=rep("Partial residual",3), terms='poly(poor\_rate, 2)', fg="gray")  
par(opar)

#### Subsection 5.2.4: Cross-validation estimates of predictive accuracy

DAAG::CVbinary(frogs.glm)

set.seed(19)  
frogs.acc <- frogs0.acc <- numeric(6)  
for (j in 1:6){  
 randsam <- sample(1:10, 212, replace=TRUE)  
 ## Sample 212 values (one per pbservation) from 1:10  
 frogs.acc[j] <- DAAG::CVbinary(frogs.glm, rand=randsam,  
 print.details=FALSE)$acc.cv  
 frogs0.acc[j] <- DAAG::CVbinary(frogs0.glm, rand=randsam,  
 print.details=FALSE)$acc.cv  
}  
print(rbind("frogs (all variables)" = frogs.acc,  
 "frogs0 (selected variables)" = frogs0.acc), digits=3)

#### Subsection 5.2.5: Cholera deaths in London — 1849 to 1855

##### By air, or by water — the 1849 epidemic

opar <- par(mgp=c(2.1,.4,0), mfrow=c(1,3))  
Cholera <- HistData::Cholera  
fitP2.glm <- glm(cholera\_deaths ~ offset(log(popn)) + water +  
 log(elevation+3) + poly(poor\_rate,2) +I(elevation==350),  
 data=Cholera, family=quasipoisson)  
Cholera[["water"]] <- factor(Cholera[["water"]], labels=c("Battersea",  
 "NewRiver","Kew"))  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
ylabs=rep("Partial residual",3), terms='water', fg="gray")  
axis(1, at=2, labels="NewRiver", lwd=0, line=0.75)  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
 ylabs=rep("Partial residual",3), terms='log(elevation + 3)', fg="gray")  
termplot(fitP2.glm, partial=T, se=TRUE, pch =1,  
 ylabs=rep("Partial residual",3), terms='poly(poor\_rate, 2)', fg="gray")  
par(opar)

##### The 1854 epidemic — a natural experiment

### Section 5.3 Logistic models for categorical data – an example

## Create data frame from multi-way table UCBAdmissions (datasets)  
## dimnames(UCBAdmissions) # Check levels of table margins  
UCB <- as.data.frame.table(UCBAdmissions["Admitted", , ], responseName="admit")  
UCB$reject <- as.data.frame.table(UCBAdmissions["Rejected", , ])$Freq  
UCB$Gender <- relevel(UCB$Gender, ref="Male")  
## Add further columns total and p (proportion admitted)  
UCB$total <- UCB$admit + UCB$reject  
UCB$pAdmit <- UCB$admit/UCB$total

UCB.glm <- glm(pAdmit ~ Dept\*Gender, family=binomial, data=UCB, weights=total)  
## Abbreviated `anova()` output:  
anova(UCB.glm, test="Chisq") |>  
 capture.output() |> tail(8) |> (\(x)x[-c(2,3)])() |> cat(sep='\n')

round(signif(coef(summary(UCB.glm)),4), 3)

### Section 5.4 Models for counts — poisson, quasipoisson, and negative binomial

#### Subsection 5.4.1: Data on aberrant crypt foci

par(pty="s")  
plot(count ~ endtime, data=DAAG::ACF1, pch=16, fg="gray")

ACF.glm <- glm(formula = count ~ endtime + I(endtime^2),  
 family = poisson(link="identity"), data = DAAG::ACF1)  
DAAG::sumry(ACF.glm, digits=2)

unique(round(predict(ACF.glm),2))

sum(resid(ACF.glm, type="pearson")^2)/19

ACFq.glm <- glm(formula = count ~ endtime + I(endtime^2),  
family = quasipoisson, data = DAAG::ACF1)  
print(coef(summary(ACFq.glm)), digits=2)

sapply(split(residuals(ACFq.glm), DAAG::ACF1$endtime), var)

fligner.test(resid(ACFq.glm) ~ factor(DAAG::ACF1$endtime))

#### Subsection 5.4.2: Moth habitat example

## Number of moths by habitat: data frame DAAG::moths  
moths <- DAAG::moths  
tab <- rbind(Number=table(moths[, 4]),  
 sapply(split(moths[, -4], moths$habitat), apply, 2, sum))

## Number of zero counts, by habitats  
with(droplevels(subset(moths, A==0)), table(habitat))

library(lattice)  
gph <- dotplot(habitat ~ A+P, data=DAAG::moths, xlab="Number of moths", outer=TRUE,  
 strip=strip.custom(factor.levels=paste("Number of species",c("A","B"))),  
 panel=function(x, y, ...){  
panel.dotplot(x,y, pch=1, ...)  
av <- sapply(split(x,y),mean)  
ypos <- factor(names(av), levels=names(av))  
lpoints(ypos~av, pch=3, col="gray45", cex=1.25)  
},  
key=list(text=list(c("Individual transects", "Mean")),  
points=list(pch=c(1,3), cex=c(1,1.25), col=c("black","gray45")),  
columns=2), scales=list(tck=0.5, alternating=1))  
bw9 <- list(fontsize=list(text=9, points=5))  
update(gph, par.settings=bw9)

Astats <- with(DAAG::moths, sapply(split(A, habitat),  
function(x)c(Amean=mean(x),Avar=var(x))))  
avlength <- with(DAAG::moths, sapply(split(meters, habitat), mean))  
round(rbind(Astats, avlen=avlength),1)

##### A quasipoisson model

A.glm <- glm(A ~ habitat + log(meters), family=quasipoisson,  
data=DAAG::moths)  
DAAG::sumry(A.glm, digits=1)

subset(DAAG::moths, habitat=="Bank")

## Analysis with tighter convergence criterion  
A.glm <- update(A.glm, epsilon=1e-10)  
print(coef(summary(A.glm)), digits=2)

AfitSE <- predict(A.glm, se=TRUE)$se.fit  
cfSE <- with(DAAG::moths, c(AfitSE[habitat=="Bank"],  
range(AfitSE[habitat!="Bank"])))  
round(setNames(cfSE, c("SEbank", "SEotherMIN", "SEotherMAX")), digits=2)

##### A more satisfactory choice of reference level

moths <- DAAG::moths  
moths$habitat <- relevel(moths$habitat, ref="Lowerside")  
Alower.glm <- glm(A ~ habitat + log(meters),  
 family = quasipoisson, data = moths)  
print(coef(summary(Alower.glm)), digits=1)

#### Subsection 5.4.3: Models with negative binomial errors

dframe <- data.frame(sigma1A =(Astats[2,]-Astats[1,])/Astats[1,]^2,  
sigma2A =(Astats[2,]-Astats[1,])/Astats[1,]^1,  
mu = Astats[1,], habitat=colnames(Astats))  
bw9 <- list(fontsize=list(text=9, points=5), pch=1:7)  
xyplot(sigma1A+sigma2A ~ mu, groups=habitat, outer=TRUE,  
data=subset(dframe,habitat!="Bank"),  
par.settings=bw9, auto.key=list(columns=4),  
strip=strip.custom(factor.levels=paste("Model",c("NBI","NBII"))),  
xlab="Mean number of species A moths",  
ylab=expression("Estimate of "\*sigma))

library(gamlss, quietly=TRUE)  
noBank <- subset(moths, habitat!='Bank')  
mothsCon.lss <- gamlss(A ~ log(meters)+habitat, family=NBI(), data=noBank,  
 trace=F)  
mothsVary.lss <- gamlss(A ~ log(meters)+habitat, family=NBI(),  
 sigma.formula=~habitat, trace=FALSE, data=noBank)

LR.test(mothsCon.lss, mothsVary.lss)

## mothsCon.lss <- gamlss(A ~ log(meters)+habitat,family=NBI(),data=noBank)  
## summary(mothsCon.lss, type="qr") ## Main part of output

##### Diagnostic plots

plot(mothsCon.lss, panel=panel.smooth)

##### Use of the square root link function

Asqrt.lss <- gamlss(A ~ habitat + sqrt(meters), trace=FALSE,  
 family = NBI(mu.link='sqrt'), data = moths)

## Asqrt.lss <- gamlss(A ~ habitat + sqrt(meters),  
## family = NBI(mu.link='sqrt'), data = moths)  
## summary(Asqrt.lss, type="qr") ## Main part of output  
out <- capture.output(summary(Asqrt.lss, digits=1))[-(3:10)]  
cat(out, sep="\n")

#### Subsection 5.4.4: Negative binomial versus alternatives — hurricane deaths

##### Aside – a quasibinomial binomial fit

ordx <- with(DAAG::hurricNamed, order(BaseDam2014))  
hurric <- DAAG::hurricNamed[ordx,]  
# Ordering a/c values of BaseDam2014 simplifies later code  
hurr.glm <- glm(deaths ~ log(BaseDam2014), family=quasipoisson, data=hurric)  
plot(hurr.glm, col=adjustcolor('black', alpha=0.4),  
 cex.caption=0.95, sub.caption=rep("",4), fg="gray")

##### Negative binomial versus power transformed scale

##### Fit a negative binomial (NBI) model

library(gamlss)  
hurrNB.gamlss <- gamlss::gamlss(deaths ~ log(BaseDam2014), family=NBI(),  
 data=hurric[-56,])  
mures <- resid(hurrNB.gamlss, what="mu")  
zres <- resid(hurrNB.gamlss, what="z-scores") ## equivalent normal quantiles

table(sign(mures))

##### Fit linear model to power transformed response

hurr.lm <- lm(car::yjPower(deaths,-0.2) ~ log(BaseDam2014), data=hurric[-56,])  
## Use the following function to transform from power scale to log scale  
powerTOlog <- function(z, lambda)log(lambda\*z+1)/lambda  
## Calculate fitted values, and transform to log(deaths+1) scale  
hatPower <- powerTOlog(predict(hurr.lm), lambda=-0.2)  
resPower <- log(hurric[-56,"deaths"]+1) - hatPower

table(sign(resPower))

##### Compare NBI and power transform fits with smoothed quantiles

library(qgam, quietly=TRUE)  
hat68.8 <- predict(qgam(log(deaths+1) ~ s(log(BaseDam2014)), qu=.648,  
 data=hurric[-56,]))  
hat40.9 <- predict(qgam(log(deaths+1) ~ s(log(BaseDam2014)), qu=.409,  
 data=hurric[-56,]))

xvar <- log(hurric$BaseDam2014)[-56]  
plot(log(deaths+1) ~ log(BaseDam2014), data=hurric, xaxt="n", yaxt="n",  
 cex=4, pch=".", fg="gray", col=adjustcolor("black",alpha.f=0.65),  
 xlab="Damage, millions of US$ in 2014", ylab="Deaths")  
axis(1, at=log(c(1,10,1000, 100000)),  
 labels=paste(c(1,10,1000, 100000)), lwd=0, lwd.ticks=1)  
axis(2, at=log(c(0,10,100,1000)+1),  
 labels=paste(c(0,10,100,1000)), lwd=0, lwd.ticks=1)  
## Negative binomial regression fitted values  
hatNB <- fitted(hurrNB.gamlss)  
lines(xvar, log(hatNB+1), col="blue", lty=2)  
with(hurric, text(log(BaseDam2014)[56], log(deaths+1)[56], "Audrey", pos=3),  
 cex=0.72)  
## Show fit from power transform model  
lines(xvar, hatPower, col="blue", lty=1)  
## Show 68.8\% and 40.1\% fits from regression smooths  
lines(hat68.8 ~ xvar, lty=2, col='red')  
lines(hat40.9 ~ xvar, lty=1, col='red')  
legend("topleft", col=rep(c('blue','red'),c(2,2)), lty=rep(2:1,2), cex=0.8,  
 y.intersp=0.75, legend=c("Negative binomial fit","Power transform fit",  
 "68.8% quantile", "40.9% quantile"), bty="n")  
mtext(side=3, "A: Deaths vs damage", line=0.5, cex=1.15, adj=0)  
## Quantile-quantile plot -- negative binomial model  
qqnorm(zres, main="", fg="gray", cex=0.5,  
 col=adjustcolor("black",alpha.f=0.65)); qqline(zres, col=2)  
mtext(side=3, "B: Q-Q plot", line=0.5, cex=1.15, adj=0)

## a) Fitted and empirical centiles from hurrNB.gamlss  
pc <- t(centiles.split(hurrNB.gamlss, xvar=log(hurric$BaseDam2014)[-56],  
 cent=c(5,10,25,50,75,90,95), xcut.points=log(c(150, 1500)),  
 plot=FALSE))  
rownames(pc) <- c("up to 150M", "150M to 1500M", "above 1500M")  
round(pc,2)

hurrP.gamlss <- gamlss(car::yjPower(deaths, -0.2) ~ log(BaseDam2014), data=hurric)

## Fitted and empirical centiles from hurrP.gamlss  
pc <- t(centiles.split(hurrP.gamlss, xvar=log(hurric$BaseDam2014),  
cent=c(5,10,25,50,75,90,95),  
xcut.points=log(c(150, 1500)), plot=FALSE))  
rownames(pc) <- c("up to 150M", "150M to 1500M", "above 1500M")  
round(pc,2)

### Section 5.5 Fitting smooths

#### Subsection 5.5.1: Handedness of first-class cricketers in the UK

tab <- with(DAAG::cricketer, table(left,dead))  
colnames(tab) <- c('live','dead')  
tab <- cbind(addmargins(tab, margin=2), prop.table(tab, margin=1))  
tab

library(mgcv)  
library(latticeExtra)  
DAAG::cricketer |> dplyr::count(year, left, name="Freq") -> handYear  
names(handYear)[2] <- "hand"  
byYear <- tidyr::pivot\_wider(handYear, names\_from='hand', values\_from="Freq")  
hand.gam <- gam(cbind(left,right) ~ s(year), data=byYear, family=binomial)  
const <- attr(predict(hand.gam, type='terms'), "constant")  
 ## `const` is the mean on the scale of the linear predictor  
plot(hand.gam, shift=const, trans=function(x)exp(x)/(1+exp(x)), ylim=c(.05,.4),  
 xlab="", ylab="Proportion lefhanded", rug=FALSE, fg="gray",  
 main=list("Proportion lefthanded, with 2SE limits",font=1,cex=1.2))  
 ## Add `const`, then apply inverse link function.  
 ## Plots estimated proportions (i.e., on the scale of the response)  
with(byYear, points(year, I(left/(left+right)), cex=0.8, col="gray50"))  
leftrt.gam <- gam(Freq ~ hand + s(year, by=factor(hand)), data=handYear,  
 family=poisson)  
leftrt.pred <- predict(leftrt.gam, se=T, type='response')  
handYear <- cbind(handYear, as.data.frame(leftrt.pred))  
col2 <- DAAG::DAAGtheme(color=T)$superpose.symbol$col[c(2,2,1)]  
gph.key <- list(space="top", columns=3, lines=list(lty=c(1,2,1), lwd=2, col=col2),  
 text=list(c("left",expression(4.4%\*%"left"),"right")), cex=1.2)  
gph <- xyplot(leftrt.pred$fit ~ year, groups=hand, ylab=list("Number born", cex=1.2),  
 type="l", xlab="", data=handYear, key=gph.key, col=col2[c(3,1)], lwd=2)  
gph1 <- xyplot(Freq~year, groups=hand, data=handYear, col=col2[c(3,1)])  
gph2 <- xyplot(I(4.4\*fit) ~ year, data=subset(handYear, hand=="left"),  
 type="l", lty=2, lwd=2, col=col2[2])  
update(gph+as.layer(gph1)+as.layer(gph2), par.settings=DAAG::DAAGtheme(color=TRUE),  
 scales=list(cex=1.2))

### Section 5.6 Additional notes on generalized linear models

#### Subsection 5.6.1: Residuals, and estimating the dispersion

##### Other choices of link function for binomial models

##### Quasi models — estimating the dispersion

#### Subsection 5.6.2: Standard errors and - or -statistics for binomial models

fac <- factor(LETTERS[1:4])  
p <- c(73, 30, 11, 2)/500  
n <- rep(500,4)  
round(signif(coef(summary(glm(p ~ fac, family=binomial, weights=n))), 6), 6)

p <- c(0.001,0.002,(1:99)/100,0.998,0.999)  
for(i in 1:3){  
link <- c("logit", "probit", "cloglog")[i]  
fun <- make.link(link)$linkfun  
x <- fun(p)  
u <- glm(p ~ x, family=binomial(link=link), weights=rep(1000,103))  
if (i==1)  
plot(x, hatvalues(u), type="l", ylab="Leverage", xaxt="n", fg='gray',  
yaxt="n",  
ylim=c(0, 0.0425), yaxs="i", xlab="Fitted proportion") else {  
phat <- predict(u, type="response")  
lines(log(phat/(1-phat)), hatvalues(u), type="l",  
col=c("black","black","gray")[i], lwd=0.75,  
lty=c(1,2,1)[i])  
}  
}  
pos=c(0.001,0.002, 0.005, 0.01,0.02,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.98,0.99, 0.995, 0.998, 0.999)  
sub1 <- seq(from=1,to=17, by=2)  
sub3 <- seq(from=2,to=16, by=2)  
axis(1, at=log(pos/(1-pos))[sub1], labels=paste(pos)[sub1],  
cex.axis=0.7, lwd=0, lwd.ticks=1)  
axis(3, at=log(pos/(1-pos))[sub3], labels=paste(pos)[sub3],  
cex.axis=0.7, lwd=0, lwd.ticks=1)  
axis(2, at=c(0,.01,.02,.03), cex.axis=.7, lwd=0, lwd.ticks=1)  
legend("topleft", lty=c(1,2,1),  
legend=c("logit link", "probit link", "cloglog link"),  
col=c("black","black","gray"), bty="n", cex=0.8)

### Section 5.7 Models with an ordered categorical or categorical response

library(VGAM)  
inhaler <- data.frame(freq=c(99,76,41,55,2,13),  
 choice=rep(c("inh1","inh2"), 3),  
 ease=ordered(rep(c("easy","re-read","unclear"), rep(2,3))))  
inhaler1.vglm <- vglm(ease ~ 1, weights=freq, data=inhaler,  
 cumulative(link="logitlink"), subset=inhaler$choice=="inh1")  
inhaler2.vglm <- vglm(ease ~ 1, weights=freq, data=inhaler,  
 cumulative(link="logitlink"), subset=inhaler$choice=="inh2")

## Inhaler 1  
round(coef(summary(inhaler1.vglm)),3)  
## Inhaler 2  
round(coef(summary(inhaler2.vglm)),3)

inhaler.vglm <- vglm(ease ~ choice, weights=freq, data=inhaler,  
cumulative(link="logitlink", parallel=FALSE))  
round(coef(summary(inhaler.vglm)),3)

inhalerP.vglm <- vglm(ease ~ choice, weights=freq, data=inhaler,  
cumulative(link="logitlink", parallel=TRUE))  
round(coef(summary(inhalerP.vglm)),3)

pred <- predict(inhalerP.vglm, se.fit=TRUE, newdata=inhaler[1:2,])  
colnames(pred$se.fit) <- paste("SE", colnames(pred$se.fit))  
fitvals <- with(pred, cbind(fitted.values, se.fit))  
colnames(fitvals) <- gsub('link', '', colnames(fitvals))  
round(fitvals, 2)

d <- deviance(inhalerP.vglm) - deviance(inhaler.vglm)  
## Refer to chi-squared distribution with 1 degree of freedom  
c(Difference=d, "p-Value"=pchisq(3.416, df=1, lower.tail=FALSE))

#### Subsection 5.7.2: Loglinear Models

### Section 5.8 Survival analysis

suppressMessages(library(survival))

df <- data.frame(x0 = c(1, 5, 1, 2, 14, 10, 12, 19)\*30,  
x1 = c(46, 58, 85, 67, 17, 85, 18, 42)\*30,  
fail = c(1, 0, 0, 1, 1, 0, 0, 1))  
plot(c(0, 2610), c(0.65, 8.15), type = "n",  
xlab = "Days from beginning of study",  
ylab = "Subject number", axes = F)  
## mtext(side = 1, line = 2.5, "Days from beginning of study", adj = 0.5)  
m <- dim(df)[1]  
par(las=2)  
axis(2, at = (1:m), labels = paste((m:1)), lwd=0, lwd.ticks=1)  
par(las=1)  
abline(v = 600, lty = 4, col="gray40")  
abline(v = 2550)  
mtext(side = 3, line = 0.5, at = c(600, 2550),  
text = c("\nEnd of recruitment",  
"\nEnd of study"), cex = 0.9)  
lines(rep((0:8) \* 300, rep(3, 9)), rep(c(-0.4, -0.2, NA), 9),  
xpd = T)  
mtext(side = 1, line = 1.0, at = (0:8) \* 300,  
text = paste((0:8) \* 300), adj = 0.5)  
chw <- par()$cxy[1]  
xx <- as.vector(t(cbind(df[, 1], df[, 2] - 0.25 \* chw,  
rep(NA, m))))  
yy <- as.vector(t(cbind(matrix(rep(m:1, 2), ncol = 2),  
rep(NA, m))))  
lines(as.numeric(xx), as.numeric(yy))  
points(df[, 1], m:1, pch = 16)  
text(df[, 1]-0.25\*chw, m:1, paste(df[,1]), pos=1, cex=0.75)  
fail <- as.logical(df$fail)  
points(df[fail, 2], (m:1)[fail], pch = 15)  
points(df[!fail, 2], (m:1)[!fail], pch = 0)  
text(df[, 2]+0.25\*chw, m:1, paste(df[,2]), pos=1, cex=0.75)  
par(xpd=TRUE)  
legend(0, 11.5, pch = 16, legend = "Entry", y.intersp=0.15)  
legend(1230, 11.5, pch = c(15, 0),  
legend = c("Dead", "Censored"), ncol=2, y.intersp=0.15)

#### Subsection 5.8.1: Analysis of the Aids2 data

str(MASS::Aids2, vec.len=2)

bloodAids <- subset(MASS::Aids2, T.categ=="blood")  
bloodAids$days <- bloodAids$death-bloodAids$diag  
bloodAids$dead <- as.integer(bloodAids$status=="D")

bloodAids <- subset(MASS::Aids2, T.categ=="blood")  
bloodAids$days <- bloodAids$death-bloodAids$diag  
bloodAids$dead <- as.integer(bloodAids$status=="D")  
plot(survfit(Surv(days, dead) ~ sex, data=bloodAids),  
 col=c(2,4), conf.int=TRUE, lty=1, fg="gray",  
 xlab="Days from diagnosis", ylab="Survival probability")  
legend("top", legend=levels(bloodAids$sex), lty=c(1,1),  
 col=c(2,4), horiz=TRUE, bty="n")

## Pattern of censoring for male homosexuals  
hsaids <- subset(MASS::Aids2, sex=="M" & T.categ=="hs")  
hsaids$days <- hsaids$death-hsaids$diag  
hsaids$dead <- as.integer(hsaids$status=="D")  
table(hsaids$status,hsaids$death==11504)

hsaids <- subset(MASS::Aids2, sex=="M" & T.categ=="hs")  
hsaids$days <- hsaids$death-hsaids$diag  
hsaids$dead <- as.integer(hsaids$status=="D")  
hsaids.surv <- survfit(Surv(days, dead) ~ 1, data=hsaids)  
plot(hsaids.surv, col="gray", conf.int=F, tcl=-0.4, fg="gray")  
par(new=TRUE)  
plot(hsaids.surv,col=1, conf.int=F,mark.time=F, fg="gray",  
xlab="Days from diagnosis", ylab="Estimated survival probabality")  
chw <- par()$cxy[1]  
chh <- par()$cxy[2]  
surv <- hsaids.surv$surv  
xval <- c(200,700,1400,1900)  
hat <- approx(hsaids.surv$time, surv, xout=xval)$y  
for(i in 1:2) arrows(xval[i], hat[i], 0, hat[i],  
length=0.05, col="gray")  
lines(rep(xval[1],2), hat[1:2], col="gray")  
## lines(rep(xval[3],2), hat[3:4], col="gray")  
## Offset triangle 1  
chw <- par()$cxy[1]  
lines(xval[c(1,2,1,1)]+650, hat[c(2,2,1,2)]+0.2,col="gray40")  
xy1 <- c(mean(xval[c(1,1,2)]), mean(hat[c(1,2,2)]))  
arrows(xy1[1], xy1[2], xy1[1]+650, xy1[2]+0.2, col="gray40", length=0.1)  
text(xval[1]-0.1\*chw+650, hat[1]+0.2,  
paste(round(hat[1],2)), col="gray20",cex=0.75, adj=1)  
text(xval[1]+650-0.1\*chw, hat[2]+0.2,  
paste(round(hat[2],2)), col="gray20",cex=0.75, adj=1)  
text(mean(xval[1:2])+650, hat[2]+0.2-0.5\*chh,  
paste(round(diff(xval[1:2]))), col="gray20", cex=0.75)  
text(xval[1]+650-0.5\*chw, mean(hat[1:2]+0.2), paste(round(hat[1]-hat[2],3)),  
srt=90, adj=0.5, col="gray20", cex=0.75)

#### Subsection 5.8.4: Hazard rates

#### Subsection 5.8.5: The Cox proportional hazards model

bloodAids.coxph <- coxph(Surv(days, dead) ~ sex, data=bloodAids)  
print(summary(bloodAids.coxph), digits=6)

## Add `age` as explanatory variable  
bloodAids.coxph1 <- coxph(Surv(days, dead) ~ sex+age, data=bloodAids)

bloodAids <- subset(MASS::Aids2,T.categ=="blood")  
bloodAids <- within(bloodAids, {days <- death-diag  
dead <- as.integer(status=="D")})  
bloodAids.coxph <- coxph(Surv(days, dead) ~ sex, data = bloodAids)  
plot(cox.zph(bloodAids.coxph), cex=0.75, bty="n")  
box(col="gray")

cox.zph(bloodAids.coxph)

cricketer <- DAAG::cricketer  
kia4.coxph <- coxph(Surv(life, kia) ~ left/poly(year,4),  
 data = cricketer, model=T)  
kia6.coxph <- update(kia4.coxph, . ~ left/poly(year,6),  
 data = cricketer, model=T)  
# Type `plot(cox.zph(kia6.coxph)` to plot the two graphs  
# Perhaps check also `AIC(kia4.coxph, kia6.coxph)`  
cox.zph(kia6.coxph)

plot(cox.zph(kia6.coxph), cex=0.75, bty="n")  
box(col="gray")

### Section 5.9: Transformations for proportions and counts

### Section 5.10: Further reading

### Exercises (5.11)

5.1

inhibition <- rbind(  
conc =c(0.1,0.5, 1,10,20,30,50,70,80,100,150),  
no = c(7, 1, 10, 9, 2, 9, 13, 1, 1, 4, 3),  
yes = c(0, 0, 3, 4, 0, 6, 7, 0, 0, 1, 7)  
)  
colnames(inhibition) <- rep("", ncol(inhibition))  
inhibition

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch5.R")  
}

# 6. Chapter 9: Multivariate data exploration and discrimination

##### Packages required (plus any dependencies)

Packages used are: DAAG MASS RColorBrewer teigen BiocManager DAAGbio hddplot lmtest splines cobalt mice datasets car micemd oz randomForest ggplot2 latticeExtra mvtnorm teigen limma hddplot mgcv MatchIt sandwich gridExtra DAAGbio mlbench (exercise).

Additionally, knitr and Hmisc are required in order to process the Rmd source file.

Hmisc::knitrSet(basename="mva", lang='markdown', fig.path="figs/g", w=7, h=7)  
oldopt <- options(digits=4, formatR.arrow=FALSE, width=70, scipen=999)  
library(knitr)  
opts\_chunk[['set']](cache.path='cache-', out.width="80%", fig.align="center",   
 fig.show='hold', ps=10, strip.white = TRUE,  
 comment=NA, width=70, tidy.opts = list(replace.assign=FALSE))

### Section 9.1: Multivariate exploratory data analysis

## Make the lattice package and the possum dataset available  
library(latticeExtra)  
possum <- DAAG::possum

#### Subsection 9.1.1: Scatterplot matrices

## Colors distinguish sexes; symbols distinguish sites  
sitenames <- row.names(DAAG::possumsites)[c(1,2,4:6,3,7)]  
key <- list(points = list(pch=0:6), text=list(sitenames),  
 columns=4, between=1, between.columns=2)  
colr <- c("red","blue")  
vnames <- c("tail\nlength","foot\nlength", "conch\nlength")  
gphA <- with(possum, splom(~ possum[, 9:11], pch=(0:6)[site], col=colr[sex],  
 xlab="", varnames=vnames, key=key, axis.line.tck=0.6))  
gphB <- with(possum, cloud(earconch~taill+footlgth, data=possum,   
 col=colr[sex], key=key, pch = (0:6)[site],   
 zlab=list("earconch", rot=90), zoom=0.925))  
update(c("A: Scatterplot matrix"=gphA, "B: Cloud plot"=gphB),  
 between=list(x=1))

#### Subsection 9.1.2: Principal components analysis

##### Preliminary data scrutiny

## Ratios of largest to smallest values: possum[, 6:14] (DAAG)  
possum <- DAAG::possum  
sapply(na.omit(possum[, 6:14]), function(x)round(max(x)/min(x),2))

## Principal components calculations: possum[, 6:14] (DAAG)  
here <- complete.cases(possum[, 6:14])  
possum.prc <- prcomp(log(possum[here, 6:14]))  
scores <- cbind(predict(possum.prc), possum[here, c('sex', 'site')])

## For parset, key and colr; see code for Fig 9.1  
pchr <- c(3,4,0,8,2,10,1)  
parset <- list(fontsize=list(text=10, points=6), cex=0.75, pch=pchr, alpha=0.8)  
key <- modifyList(key, list(columns=1, space="right"))  
gph <- with(scores, xyplot(PC2 ~ PC1, aspect="iso", key = key,  
 col = colr[sex], pch = (0:6)[site]))  
update(gph, scales=list(tck=0.5), par.settings=parset,  
 xlab="1st Principal Component", ylab="2nd Principal Component")

print(summary(possum.prc),digits=2)  
cat("\nRotations (otherwise called Loadings)\n")  
print(possum.prc$rotation, digits=2)  
## By default, blanks are shown for loadings < 0.1 in magnitude

##### The stability of the principal components plot

suppressPackageStartupMessages(library(ggplot2))  
theme\_set(theme\_gray(base\_size=8))  
## Bootstrap principal components calculations: possum (DAAG)  
## Sample from rows where there are no missing values  
rowsfrom <- (1:nrow(possum))[complete.cases(possum[, 6:14])]  
logpossum6to14 <- log(possum[rowsfrom, 6:14])  
sexPop <- possum[rowsfrom, c("sex","Pop")]  
n <- length(rowsfrom); ntimes <- 3  
bootscores <- data.frame(scores1=numeric(ntimes\*n), scores2=numeric(ntimes\*n))  
for (i in 1:ntimes){  
 samprows <- sample(1:n, n, replace=TRUE)  
 bootscores[n\*(i-1)+(1:n), 1:2] <-  
 prcomp(logpossum6to14[samprows, ])$x[, 1:2]  
}  
bootscores[, c("sex","Pop")] <- sexPop[samprows, ]  
bootscores$sampleID <- factor(rep(1:ntimes, rep(n,ntimes)))  
gph <- quickplot(x=scores1, y=scores2, colour=sex, size=I(1.0),  
 asp=1, shape=Pop, facets=.~sampleID, data=bootscores) +   
 scale\_shape\_discrete(solid=F)  
gph + scale\_colour\_manual(values=c("m"="blue","f"="red")) +  
 xlab("First Principal Component") + ylab("Second Principal Component")

#### Subsection 9.1.3: Multi-dimensional scaling

##### Distance measures

##### Ordination

## Code that will display individual graphs  
d.possum <- dist(possum[,6:14]) # Euclidean distance matrix  
MASS::sammon(d.possum, k=2, trace=FALSE)$points |> as.data.frame() |>  
 setNames(paste0("ord",1:2)) |> cbind(Pop=DAAG::possum$Pop) -> sammon.possum  
MASS::isoMDS(d.possum, k=2, trace=FALSE) |> as.data.frame() |>   
 setNames(paste0("ord",1:2)) |> cbind(Pop=DAAG::possum$Pop) -> mds.possum  
gph1 <- xyplot(ord2~ord1, groups=Pop, aspect="iso", data=sammon.possum)  
gph2 <- xyplot(ord2~ord1, groups=Pop, aspect="iso", data=mds.possum)  
update(c(gph1, gph2, layout=c(2,1)),   
 par.settings=simpleTheme(pch=c(1,3)),  
 between=list(x=0.5), auto.key=list(columns=2),  
 strip=strip.custom(factor.levels=c("A: Sammon","B: ISOmds")))

##### Binary data

### Section 9.2: Principal component scores in regression

## Principal components: data frame socsupport (DAAG)  
socsupport <- DAAG::socsupport  
ss.pr1 <- prcomp(as.matrix(na.omit(socsupport[, 9:19])), retx=TRUE, scale=TRUE)

oldpar <- par(fg='gray40',col.axis='gray20',lwd=0.5,col.lab='gray20')  
pairs(ss.pr1$x[, 1:3], col='gray40', gap=0.2)  
par(oldpar)

summary(sort(ss.pr1$rotation[,1]))  
## Note the very large maximum value  
which.max(ss.pr1$x[,1])  
## Try also boxplot(ss.pr1$x[,1])  
## ss.pr1$x["36",1] ## Check that this returns 42

use <- complete.cases(socsupport[, 9:19])  
use[36] <- FALSE  
ss.pr <- prcomp(as.matrix(socsupport[use, 9:19]))

## Output from summary()  
print(summary(ss.pr), digits=1) # Compare contributions

comp <- as.data.frame(ss.pr$x[,1:6])  
ss.lm <- lm(socsupport[use, "BDI"] ~ ., data=comp)  
signif(round(coef(summary(ss.lm)),5), digits=3)

print(ss.pr$rotation[, 1], digits=2)

## Plot BDI against first principal component score  
gph <- xyplot(BDI ~ ss.pr$x[ ,1], groups=gender, data=socsupport[use,],  
par.settings=simpleTheme(pch=1:2), auto.key=list(columns=2))  
bw9 <- list(pch=c(1,3), list(text=9, points=5))  
update(gph, scales=list(tck=0.5), par.settings=bw9,  
xlab ="1st principal component")

### Section 9.3: Cluster analysis

#### Subsection 9.3.1: Hierarchical Clustering

library(mvtnorm)  
makeClust <- function(n=6, d1=4, d2=4, sigs=c(1, 1, 1, 1), seed=NULL){  
 if(!is.null(seed))set.seed(seed)  
 g1 <- rmvnorm(n, mean = c(-d1,d2), sigma=sigs[1]\*diag(2))  
 g2 <- rmvnorm(n, mean = c(d1,d2), sigma=sigs[2]\*diag(2))  
 g3 <- rmvnorm(n, mean = c(-d1,-d2), sigma=sigs[3]\*diag(2))  
 g4 <- rmvnorm(n, mean = c(d1,-d2), sigma=sigs[4]\*diag(2))  
 rbind(g1,g2,g3,g4)  
}

## Code for the plots  
datA <- makeClust(seed=35151)  
datB <- makeClust(d2=16, seed=35151)  
datC <- makeClust(d1=2,d2=2, seed=35151)  
plot(datA, xlab="X1", ylab="X2", fg="gray")   
title(main="A: 4blobsA", adj=0, line=0.5, font.main=1)  
## Repeat previous two lines for datB and datC  
plot(datB, xlab="X1", ylab="X2", fg="gray")   
title(main="B: 4blobsB", adj=0, line=0.5, font.main=1)  
plot(datC, xlab="X1", ylab="X2", fg="gray")   
title(main="C: 4blobsC", adj=0, line=0.5, font.main=1)

## Possible alternative  
config <- c('Equidistant blobs', 'Pulled vertically', 'Closer centers')  
dat123 <- cbind(as.data.frame(rbind(datA, datB, datC)),   
 gp=factor(rep(1:3, rep(6\*4,3)), labels=config))  
xyplot(V2 ~ V1 | gp, data=dat123, scales=list(relation='free'),  
 strip=strip.custom(factor.levels=config), between=list(x=0.5),  
 par.settings=DAAG::DAAGtheme(color=F))

## Code for single linkage plots: `?plot.hclust` gives help for the plot method  
clusres\_sing <- hclust(dist(datA), method="single")  
par(fig=c(0,0.75,0,1))  
plot(clusres\_sing, sub="", xlab="", ylab="Distance joined", adj=0.5,   
 main="", fg="gray")  
mtext('A: Single linkage cluster dendrogram, for 4blobsA layout', side=3, adj=0,   
 font=1, line=1, cex=1.15)  
par(fig=c(0.72,1,0,1), new=TRUE)  
membs <- cutree(clusres\_sing, 4)  
col4= RColorBrewer::brewer.pal(4,'Set1')  
plot(datA, xlab="X1", ylab="X2", col=col4[membs], fg='gray', pch=membs+1)  
mtext('B: 4blobsA, by color', side=3, adj=1.0, font=1, cex=1.15, line=1)  
## To see plots from 'average' and 'complete' linkage methods,do:  
# plot(hclust(dist(datB), method="average"))  
# plot(hclust(dist(datC), method="complete"))

## Dendrograms from data where blobs were pulled vertically  
## Follow each use of `hclust()` with a `plot()` command  
sclusres\_sing <- hclust(dist(datB), method="single")  
plot(sclusres\_sing, sub="", xlab="", ylab="Distance Joined", main="")  
title(main='A: Single linkage, blobs pulled vertically (4blobsB)',   
 adj=0, font.main=1)  
sclusres\_sing\_s <- hclust(dist(scale(datA)), method="single")  
plot(sclusres\_sing\_s, sub="", xlab="", ylab="Distance Joined", main="")  
title(main='B: Single linkage, (4blobsB, rescaled to variance 1)',   
 adj=0, font.main=1)  
# sclusres\_avg\_s <- hclust(dist(scale(datB)), method="average")  
# #plot(sclusres\_avg\_s, sub="", xlab="", ylab="")  
# sclusres\_comp\_s <- hclust(dist(scale(datB)), method="complete")  
# #plot(sclusres\_comp\_s, sub="", xlab="", ylab="")

## Code. Follow each use of `hclust()` with a `plot()` command  
clusres\_sing2 <- hclust(dist(datC), method="single")  
plot(clusres\_sing2, sub="", xlab="", ylab="", cex=1.25, cex.main=1.65,  
 main="A: Single linkage, closer clusters (4blobsC)", adj=0, font.main=1)  
clusres\_avg2 <- hclust(dist(datC), method="average")  
plot(clusres\_avg2, sub="", xlab="", ylab="", cex=1.25, cex.main=1.65,  
 main="B: Average linkage, closer clusters (4blobsC)", adj=0, font.main=1)  
clusres\_comp2 <- hclust(dist(datC), method="complete")  
plot(clusres\_comp2, sub="", xlab="", ylab="", cex=1.25, cex.main=1.65,   
 main="C: Complete linkage, closer clusters (4blobsC)", adj=0, font.main=1)

#### Subsection 9.3.2: -Means Clustering

set.seed(35151)  
kdat <- makeClust(n=100, d1=5, d2=5, sigs=c(.5, .5, 6, 6))  
plot(kdat, xlab="X1", ylab="X2", fg="gray")  
kmres <- kmeans(kdat, 4, nstart=30)  
plot(kdat, col=rainbow(4)[kmres$cluster], pch=kmres$cluster+1,   
 xlab="X1", ylab="X2", fg="gray")

##### Comments on -means and hierarchical clustering

#### Subsection 9.3.3: Mixture model-based clustering

## Code  
plotMix2 <- function(taus=c(.5, .5), means=c(10,15), sds=c(3,1), xlims=c(0,20)){  
 curve(taus[1]\*dnorm(x, mean=means[1], sd=sds[1]) +   
 taus[2]\*dnorm(x, mean=means[2], sd=sds[2]),   
 from=xlims[1], to=xlims[2], ylab="Density", fg="gray")  
 curve(taus[1]\*dnorm(x, mean=means[1], sd=sds[1]),   
 from=xlims[1], to=xlims[2], col="red", lty=2, add=TRUE, fg="gray")  
 curve(taus[2]\*dnorm(x, mean=means[2], sd=sds[2]),   
 from=xlims[1], to=xlims[2], col="blue", lty=3, add=TRUE, fg="gray")  
}  
plotMix2(taus=c(.2, .8))  
plotMix2(taus=c(.5, .5))  
plotMix2(taus=c(.9, .1))

library(teigen)  
possml <- na.omit(DAAG::possum[,c(3,9:11)])  
set.seed(513451)  
gaus\_fit <- teigen(possml[,2:4], models="UUUU", gauss=TRUE, verbose=FALSE,   
 scale=FALSE)

## BIC values are plotted against number of groups  
gaus\_fit$allbic  
plot(gaus\_fit$allbic, type="b", ylab="", xlab="Number of Groups", fg="gray")  
mtext(side=2, line=3.5, "BIC", las=0)  
axis(1, at=1:9, fg="gray")

table(possml$Pop, gaus\_fit$classification)

par(fig=c(0, 0.5, 0.5, 1))  
plot(gaus\_fit, what="contour", xmarg=1, ymarg=2, draw.legend=FALSE, fg="gray")  
## See ?teigen::plot.teigen for details of the plot command used here.  
par(fig=c(0, 0.5, 0, 0.5), new=TRUE)  
plot(gaus\_fit, what="contour", xmarg=1, ymarg=3, draw.legend=FALSE, fg="gray")  
par(fig=c(0.5, 1, 0, 0.5), new=TRUE)  
plot(gaus\_fit, what="contour", xmarg=2, ymarg=3, draw.legend=FALSE, fg="gray")  
par(fig=c(0,1,0,1))

##### Issues of high parametrization and scaling

#### Subsection 9.3.4: Relationship between -means and mixture models

### Section 9.4: Discriminant analysis

#### Subsection 9.4.1: Example – plant architecture

leafshape17 <- DAAG::leafshape17  
plot(bladelen ~ bladewid, data=leafshape17, pch=c(1,3)[arch+1])  
## For panel B, specify log="xy" in the call to plot()

##### Logistic regression, versus linear discriminant analysis

#### Subsection 9.4.2: Logistic regression

## Fit logistic regression model  
leafshape17 <- DAAG::leafshape17  
leaf17.glm <- glm(arch ~ logwid + loglen, family=binomial(link=logit),  
data=leafshape17)  
print(DAAG::sumry(leaf17.glm)$coef, digits=2)

##### Predictive accuracy

set.seed(29)  
leaf17.cv <- DAAG::CVbinary(leaf17.glm)  
tCV <- table(DAAG::leafshape17[["arch"]], round(leaf17.cv$cvhat))  
rownames(tCV) <- colnames(tCV) <- c("0=Plagiotropic","1=Orthotropic")  
cbind(tCV, "Proportion correct"=c(tCV[1,1], tCV[2,2])/(tCV[,1]+tCV[,2]))

round(unlist(leaf17.cv[c("acc.training","acc.cv")]),3)

#### Subsection 9.4.3: Linear discriminant analysis

suppressPackageStartupMessages(library(MASS))  
## Discriminant analysis; data frame leafshape17 (DAAG)  
leaf17.lda <- lda(arch ~ logwid+loglen, data=DAAG::leafshape17)  
print(leaf17.lda)

##### Assessments of predictive accuracy

set.seed(29)  
leaf17cv.lda <- lda(arch ~ logwid+loglen, data=leafshape17, CV=TRUE)  
## the list element 'class' gives the predicted class  
## The list element 'posterior' holds posterior probabilities  
tab <- table(leafshape17$arch, leaf17cv.lda$class)  
rownames(tab) <- colnames(tab) <- c("0=Plagiotropic","1=Orthotropic")  
cbind(tab, "Proportion correct"=c(tCV[1,1], tCV[2,2])/(tCV[,1]+tCV[,2]))  
cbind(tab, c(tab[1,1], class.acc=tab[2,2])/(tab[,1]+tab[,2]))  
cat("Overall proportion correct =", sum(tab[row(tab)==col(tab)])/sum(tab), "\n")

##### The function qda(), and other alternatives to lda()

#### Subsection 9.4.4: An example with more than two groups

## Linear discriminant calculations for possum data  
possum <- DAAG::possum  
possum.lda <- lda(site ~ hdlngth + skullw + totlngth + taill + footlgth +  
 earconch + eye + chest + belly, data=na.omit(possum))  
# na.omit() omits any rows that have one or more missing values

plot(possum.lda, dimen=3, col=1:9)  
# Scatterplot matrix - scores on 1st 3 canonical variates  
# See `?plot.lda` for details of the generic lda plot function

## Linear discriminant calculations for possum data  
print(possum.lda, digits=3)

### Section 9.5: \*High-dimensional data — RNA-Seq gene expression

##### Setup for installing and using Bioconductor packages

## For latest details, see: https://www.bioconductor.org/install/  
if (!require("BiocManager", quietly = TRUE))  
 install.packages("BiocManager")  
BiocManager::install()  
BiocManager::install('limma','multtest')

##### \*Brief note on mRNA technical issues

#### Subsection 9.5.1: Data and design matrix setup

counts <- DAAGbio::plantStressCounts  
colSums(counts)

## Require at least 3 counts per million that are > 1  
keep <- rowSums(counts)>=3  
counts <- counts[keep,]

treatment <- factor(rep(c("CTL", "L", "D"), rep(3,3)))  
design <- model.matrix(~0+treatment)  
colnames(design) <- levels(treatment)

##### A two-dimensional representation

library(limma)  
v <- voom(counts, design, plot=TRUE)

par(oma=c(0,0,1,0))  
library(limma)  
v <- voom(counts, design, plot=TRUE)  
firstchar <- substring(colnames(counts),1,1)  
plotMDS(counts, labels=paste0(firstchar, rep(1:3,3)), cex=0.8)  
box(col="gray")  
mtext(side=3, line=0.4, adj=0, "MDS summary plot")  
mtext(side=3, line=-0.25, adj=0.105, "A", outer=TRUE)  
mtext(side=3, line=-0.25, adj=0.605, "B", outer=TRUE)

##### Fitting the model

fit <- lmFit(v, design)

contrs <- c("D-CTL", "L-CTL", "L-D")  
contr.matrix <- makeContrasts(contrasts=contrs,  
levels=levels(treatment))  
fit2 <- contrasts.fit(fit, contr.matrix)  
efit2 <- eBayes(fit2)

#### Subsection 9.5.2: From -values to false discovery rate (FDR)

## First contrast only; Drought-CTL  
print(round(topTable(efit2, coef=1, number=4),15), digits=3)

round(sort(p.adjust(p=efit2$p.value[,1], method="BH"))[1:4], 15) # Not run

round(topTable(efit2, number=4), 15)

head(decideTests(fit2),5)

summary(decideTests(fit2))  
## Try also  
## summary(decideTests(fit2, p.value=0.001))

### Section 9.6: High dimensional data from expression arrays

#### Subsection 9.6.1: Molecular classification of cancer — an older technology

##### Breakdown of ALL B-type data, with one observation excluded

library(hddplot)  
data(golubInfo)  
with(golubInfo, table(cancer, tissue.mf))

## Identify allB samples that are BM:f or BM:m or PB:m  
subsetB <- with(golubInfo,  
cancer=="allB" & tissue.mf%in%c("BM:f","BM:m","PB:m"))  
## Separate off the relevant columns of the matrix Golub  
## NB: variables (rows) by cases (columns)  
GolubB <- with(golubInfo, Golub[, subsetB])  
## Form vector that identifies these as BM:f or BM:m or PB:m  
tissue.mfB <- with(golubInfo, tissue.mf[subsetB, drop=TRUE])  
## Change the level names to leave out the colons  
levels(tissue.mfB) <- list("b\_f"="BM:f", "b\_m"="BM:m", "PBm"="PB:m")

#### Subsection 9.6.2: Classifications and associated graphs

##### Preliminary data manipulation

## Display distributions for the first 20 observations  
boxplot(data.frame(GolubB[, 1:20])) # First 20 columns (observations)  
## Random selection of 20 rows (features)  
boxplot(data.frame(GolubB[sample(1:7129, 20), ]))

##### Flawed graphs

colr <- c("red","blue","gray40", "magenta")  
tissue.mf <- golubInfo[, "tissue.mf"]  
cancer <- golubInfo[, "cancer"]  
G.PBf <- Golub[, tissue.mf=="PB:f" & cancer=="allB", drop=FALSE]  
set.seed(41)  
rGolubB <- matrix(rnorm(prod(dim(GolubB))), nrow=dim(GolubB)[1])  
rownames(rGolubB) <- rownames(Golub)  
rG.PBf <- matrix(rnorm(prod(dim(G.PBf))), nrow=dim(G.PBf)[1])  
plot2 <- function(x = GolubB, cl=tissue.mfB, x.omit=Golub.PBf, cl.omit="PBf",   
 ncol = length(cl), nfeatures=12, device = "", seed = 37,  
 pretext="", colr=1:3, levnames = NULL,  
 ylab="2nd discriminant function"){  
 cl <- factor(cl)  
 if(!is.null(levnames))levels(cl) <- levnames  
 ord15 <- orderFeatures(x, cl=cl)[1:nfeatures]  
 dfB <- t(x[ord15, ])  
 dfB.lda <- lda(dfB, grouping=cl)  
 scores <- predict(dfB.lda, dimen=2)$x  
 df.PBf <- data.frame(t(x.omit[ord15, drop=FALSE]))  
 colnames(df.PBf) <- colnames(dfB)  
 scores.other <- predict(dfB.lda, newdata=df.PBf)$x  
 scoreplot(list(scores=scores, cl=cl, other=scores.other, cl.other=cl.omit, nfeatures=nfeatures), prefix.title=pretext, adj.title=0,   
 fg="gray", params=list(other=list(pch=4, cex=1.5)),  
 xlab="1st discriminant function", ylab=ylab)  
}  
plot2(x = GolubB, cl = tissue.mfB, x.omit=G.PBf, cl.omit="PBf",  
 nfeatures=15, device = "", seed = 37, ylab="2nd discriminant function",  
 colr=colr, pretext="A: ALL B-cell:")  
plot2(x = rGolubB, cl = tissue.mfB, x.omit=rG.PBf, cl.omit="Gp 4",   
 device = "", seed = 37, colr=colr, levnames = c("Gp 1", "Gp 2", "Gp 3"),  
 pretext="B: Random data:", ylab="")

## Uses orderFeatures() (hddplot); see below  
ord15 <- orderFeatures(GolubB, cl=tissue.mfB)[1:15]

## Panel A: Take 1st 15 features & transpose to observations by features  
dfB15 <- data.frame(t(GolubB[ord15, ]))  
dfB15.lda <- MASS::lda(dfB15, grouping=tissue.mfB)  
scores <- predict(dfB15.lda, dimen=2)$x  
## Scores for the single PB:f observation  
chooseCols <- with(golubInfo, tissue.mf=="PB:f"& cancer=="allB")  
df.PBf <- data.frame(t(Golub[ord15, chooseCols, drop=FALSE]))  
scores.PBf <- predict(dfB15.lda, newdata=df.PBf, dimen=2)$x  
## Warning! The plot that now follows may be misleading!  
## Use hddplot::scoreplot()  
scoreplot(list(scores=scores, cl=tissue.mfB, other=scores.PBf, cl.other="PB:f"),  
 fg="gray")

## Panel B: Repeat plot, now with random normal data  
simscores <- simulateScores(nrow=7129, cl=rep(1:3, c(19,10,2)),  
cl.other=4, nfeatures=15, seed=41)  
# Returns list elements: scores, cl, scores.other & cl.other  
scoreplot(simscores)

#### Subsection 9.6.3: The mean-variance relationship

par(oma=c(0,0,1,0))  
designG <- model.matrix(~0+tissue.mfB)  
colnames(designG) <- levels(tissue.mfB)  
vG <- vooma(GolubB, designG, plot=TRUE) # Panel A  
plotMDS(vG, pch=unclass(tissue.mfB), cex=0.8) # Panel B  
leglabs <- c("BM:female","BM:male","PB:female")  
legend(x="bottomright", bty="n", legend=leglabs, pch=1:3)  
mtext(side=3, line=0.4, adj=0, "MDS summary plot")  
mtext(side=3, line=-0.275, adj=0.085, "A", outer=TRUE)  
mtext(side=3, line=-0.275, adj=0.585, "B", outer=TRUE)

##### Cross-validation to determine the optimum number of features

##### Cross-validation for a range of choices of number of features

## Cross-validation to determine the optimum number of features  
## 10-fold (x4). Warning messages are omitted.  
## Accuracy measure will be: tissue.mfB.cv$acc.cv  
tissue.mfB.cv <- cvdisc(GolubB, cl=tissue.mfB, nfeatures=1:23,  
nfold=c(10,4), print.progress=FALSE)  
## Defective measures will be in acc.resub (resubstitution)  
## and acc.sel1 (select features prior to cross-validation)  
tissue.mfB.badcv <- defectiveCVdisc(GolubB, cl=tissue.mfB,  
foldids=tissue.mfB.cv$folds, nfeatures=1:23)  
##  
## Calculations for random normal data:  
set.seed(43)  
rGolubB <- matrix(rnorm(prod(dim(GolubB))), nrow=nrow(GolubB))  
rtissue.mfB.cv <- cvdisc(rGolubB, cl=tissue.mfB, nfeatures=1:23,  
nfold=c(10,4), print.progress=FALSE)  
rtissue.mfB.badcv <- defectiveCVdisc(rGolubB, cl=tissue.mfB,  
nfeatures=1:23,  
foldids=rtissue.mfB.cv$folds)

##### Which features?

genelist <- matrix(tissue.mfB.cv$genelist[1:3, ,], nrow=3)  
tab <- table(genelist, row(genelist))  
ord <- order(tab[,1], tab[,2], tab[,3], decreasing=TRUE)  
tab[ord,]

#### Subsection 9.6.4: Graphs derived from the cross-validation process

## Uses tissue.mfB.acc from above  
tissue.mfB.scores <-  
cvscores(cvlist = tissue.mfB.cv, nfeatures = 3, cl.other = NULL)  
scoreplot(scorelist = tissue.mfB.scores, cl.circle=NULL,  
prefix="B-cell subset -", fg='gray')

##### The key role of cross-validation

#### Subsection 9.6.5: Estimating contrasts, and calculating False Discovery Rates

fitG <- lmFit(vG, designG)  
contrs <- c("b\_f-b\_m", "b\_f-PBm", "b\_m-PBm")  
contr.matrix <- makeContrasts(contrasts=contrs,  
levels=levels(tissue.mfB))  
fit2 <- contrasts.fit(fitG, contr.matrix)  
fit2 <- eBayes(fit2)

##### From -values to false discovery rate (FDR)

print(topTable(fit2, number=5), digits=2)

summary(decideTests(fit2))  
## Try also  
## summary(decideTests(fit2, p.value=0.001))

##### Distributional extremes

### Section 9.7: Causal inference from observational data — balance and matching

#### Subsection 9.7.1: Tools for the task

library(DAAG)  
## Columns 4:7 are factors; columns 9:10 (re75 & re78) are continuous  
propmat <- matrix(0, ncol=6, nrow=8)  
dimnames(propmat) <- list(c("psid1", "psid2", "psid3", "cps1", "cps2", "cps3",  
"nsw-ctl", "nsw-trt"), names(nswdemo)[c(4:7, 9:10)])  
for(k in 1:8){  
 dframe <- switch(k, psid1, psid2, psid3, cps1, cps2, cps3,  
 subset(nswdemo, trt==0), subset(nswdemo, trt==1))  
 propmat[k,] <- c(sapply(dframe[,4:7], function(x){  
 z <- table(x); z[2]/sum(z)}),  
 sapply(dframe[,9:10], function(x)sum(x>0)/sum(!is.na(x))))  
}

PGtheme <- DAAG::DAAGtheme(color=TRUE)  
library(DAAG)  
if(!require(grid))return("Package 'grid' is not installed -- cannot proceed")  
dsetnames <- c("nsw-ctl", "nsw-trt", "psid1", "psid2", "psid3",  
 "cps1", "cps2", "cps3")  
colrs <- c("gray","black", PGtheme$superpose.line$col[1:3])  
lty <- c(1,2,1,1,1)  
lwd <- c(1,0.75,0.75,0.75,0.75)  
denplot <-  
 function(sel=c(1:2,6:8), yvar="re75", offset=30, ylim=c(0,1.75),  
 from=NULL, at=c(.5,1,1.5), labels=paste(at), bw="nrd0",  
 ylab="Density", takelog=TRUE, col.axis="black"){  
 nzre <- unlist(lapply(list(subset(nswdemo, trt==0),  
 subset(nswdemo, trt==1),  
 psid1, psid2, psid3, cps1, cps2, cps3)[sel],  
 function(x){z <- x[,yvar]; z[z>0]}))  
num <- unlist(lapply(list(subset(nswdemo, trt==0), subset(nswdemo, trt==1),  
 psid1, psid2, psid3, cps1, cps2, cps3),  
 function(x){z <- x[,yvar]; sum(z>0)}))  
xy <- data.frame(nzre=nzre, fac = factor(rep(dsetnames[sel], num[sel]),  
 levels=dsetnames[sel]))  
if(takelog) {  
y <- log(xy$nzre+offset)  
xlab <- paste("log(", yvar, "+", offset, ")", sep="")} else   
 {  
 y <- xy$nzre  
 xlab <- yvar  
}  
densityplot(~ y, groups=fac, data=xy, bw=bw, from=from,  
 scales=list(y=list(at=at, labels=labels, col=col.axis), tck=0.25),  
 plot.points=FALSE, col=colrs[1:5], lwd=lwd, lty=lty,   
 key=list(x=0.01, y=0.99, text=list(dsetnames[sel[3:5]]), col=colrs[3:5],  
 cex=0.75, lines=list(lwd=rep(1.5,3)), between=1),  
 par.settings=list(col=colrs, lty=lty, cex=0.75, lwd=lwd,   
 fontsize=list(text=9, points=5)),  
 fg="gray", ylim=ylim, ylab=ylab, xlab=xlab)  
}  
## Plot base graph; overlay with lattice graphs on same page  
par(fig=c(0,1,0,1), mar=c(0,0,0,0))  
plot(0:1,0:1, axes=FALSE, type="n", bty="n", xlab="", ylab="")  
legend(x="top",legend=dsetnames[1:2], lty=1:2, lwd=c(1,0.75),  
 col=colrs[1:2], bty="n", ncol=2, yjust=0.75)  
print(denplot(), position=c(0, 0, 0.32, 0.505), newpage=FALSE)  
print(denplot(1:5, ylab=" ", col.axis="white"),  
 position=c(0.21, 0, .53, 0.505), newpage=FALSE)  
print(denplot(ylab=" ", yvar="re78", col.axis="white"),  
 position=c(0.47, 0, 0.79, 0.505), newpage=FALSE)  
print(denplot(1:5, ylab=" ", yvar="re78", col.axis="white"),  
 position=c(0.68, 0, 1, 0.505), newpage=FALSE)  
## Age  
print(denplot(yvar="age", takelog=FALSE, ylim=c(0,0.1), from=16,  
 at=c(.02,.04,.06,.08), labels=c(".02",".04",".06",".08")),  
 position=c(0, 0.475, 0.32, .98), newpage=FALSE)  
print(denplot(1:5, yvar="age", takelog=FALSE, ylim=c(0,0.1), from=16,  
 at=c(.02,.04,.06,.08), labels=c(".02",".04",".06",".08"),  
 ylab=" ", col.axis="white"),  
 position=c(0.21, 0.475, .53, .98), newpage=FALSE)  
## educ  
print(denplot(1:5, yvar="educ", takelog=FALSE, ylim=c(0,0.5), bw=0.5,  
 at=c(.1,.2,.3,.4), ylab=" "),  
 position=c(0.47, 0.475, .79, .98), newpage=FALSE)  
print(denplot(yvar="educ", takelog=FALSE, ylim=c(0,0.75), bw=0.5,  
 at=c(.1,.2,.3,.4), ylab=" ", col.axis="white"),  
 position=c(0.68, 0.475, 1, .98), newpage=FALSE)

addControl <-  
function(control, offset=30){  
 nam <- deparse(substitute(control))  
 if(nam=="nswdemo")nsw0 <- nswdemo else  
 nsw0 <- rbind(control, subset(DAAG::nswdemo, trt==1))  
 nsw0$z75 <- factor(nsw0$re75==0, labels=c("0",">0"))  
 nsw0$ethnicid <- factor(with(nsw0, ifelse(black==1, "black",  
 ifelse(hisp==1, "hisp", "other"))), levels=c("other","black","hisp"))  
nsw0 <- nsw0[, -match(c("black","hisp"), names(nsw0))]  
nsw0  
}

## Create dataset that will be used for later analyses  
nsw <- addControl(psid1)  
nsw <- within(nsw, {re75log <- log(re75+30);  
 re78log <- log(re78+30);  
 trt <- factor(trt, labels=c("Control","Treat"))})  
## A treated values only dataset will be required below  
trtdat <- subset(nsw, trt=="Treat")  
trtdat$pres74 <- factor(!is.na(trtdat$re74), labels=c("<NA>","pres"))  
table(trtdat$pres74)

with(trtdat, table(pres74,z75))

#### Subsection 9.7.2: Regression comparisons

##### Regression calculations

nsw.gam <- gam(log(re78+30)~ trt + ethnicid + z75 + nodeg + s(age) +  
 s(educ) + log(re75+30), data=nsw)

#### Subsection 9.7.3: The use of scores to replace covariates

#### Subsection 9.7.4: Two-dimensional representation using randomForest proximities

suppressPackageStartupMessages(library(randomForest))  
form <- trt ~ age + educ + ethnicid + marr + nodeg + z75 + re75log  
nsw.rf <- randomForest(form, data=nsw, sampsize=c(297,297))  
p.rf <- predict(nsw.rf,type="prob")[,2]  
sc.rf <- log((p.rf+0.001)/(1-p.rf+0.001))

omitn <- match(c("PropScore","weights","subclass"), names(dat2RF), nomatch=0)  
matchISO.rf <-matchit(trt ~ age + educ + ethnicid + marr + nodeg + z75 +  
 re75log, ratio=1, data=dat2RF[,-omitn], distance=isoScores[,1])  
## summary(match.rf,un=F,improvement=F)  
## summary(match.rf, un=F, interactions=T, improvement=F)$sum.matched[,1:4]  
## In the first place, look only at the first 4 columns

dat1RF <- match.data(matchISO.rf, distance="PropScore")  
dat1RF.lm <- lm(re78log ~ trt, data = dat1RF, weights = weights)  
library(sandwich) # Allows use of `vcovCL()` from the `sandwich` package  
lmtest::coeftest(dat1RF.lm, vcov. = vcovCL, cluster = ~subclass)  
## Check for increase in number with non-zero earnings  
dat1RF.glm <- glm(I(re78>0) ~ trt, data = dat1RF, weights = weights,   
 family=binomial)  
lmtest::coeftest(dat1RF.glm, vcov. = vcovCL, cluster = ~subclass)

##### Derivation and investigation of scores

library(mgcv)  
formG <- trt ~ ethnicid + marr+ z75 + s(age) + s(educ) + s(re75log)  
nsw.gam <- gam(formG, family=binomial(link="logit"), data=nsw)  
pred <- predict(nsw.gam, type='response')  
table(nsw$trt, round(pred))  
## Alternative  
library(splines) ## Fit normal cubic splines using splines::ns()  
formNS <- trt ~ ethnicid + marr+ z75 + ns(age,2) +  
ns(educ) + ns(re75log,3)  
nsw.glm <- glm(formNS, family=binomial(link="logit"), data=nsw)  
pred <- predict(nsw.glm, type='response')  
table(nsw$trt, round(pred))  
cbind(AIC(nsw.glm,nsw.gam), BIC(nsw.glm, nsw.gam))

## Include factor by factor and variable interactions with ethnicid  
## and marr (Result not shown)  
formGx <- trt ~ (ethnicid+marr+z75)^2 + s(age, by=ethnicid)+  
 s(educ, by=ethnicid) + s(re75log,by=ethnicid)+  
 s(age, by=marr)+ s(educ, by=marr) + s(re75log,by=marr)  
nswx.gam <- gam(formula = formGx, data = nsw, family=binomial(link = "logit"))  
predx <- predict(nswx.gam, type='response')  
table(nsw$trt, round(predx))  
AIC(nsw.glm,nsw.gam,nswx.gam)

library(MatchIt)  
## Use data frame that omits re74. Otherwise matchit() will generate NAs  
## where they occur in re74, even though re74 is not in the model formula.  
nswG <- nsw[, c("trt","age","educ","ethnicid", "marr","nodeg","z75",  
 "re75log","re78log","re78")]  
formG <- trt ~ ethnicid + marr+ z75 + s(age) + s(educ) + s(re75log)  
match.gam <- matchit(formula = formG, data = nswG, method = "nearest",  
 distance = "gam", link = "logit", reestimate=TRUE)  
datG <- match.data(match.gam, distance="PropScore")  
## Summary information  
match.gam  
## summary(match.gam,un=F,improvement=F)  
## summary(match.gam, un=F, interactions=T, improvement=F)$sum.matched[,1:4]  
## In the first place, look only at the first 4 columns

suppressPackageStartupMessages(library(gridExtra))  
suppressPackageStartupMessages(library(ggplot2))  
suppressPackageStartupMessages(library(cobalt))  
gg1<- cobalt::love.plot(match.gam, position="bottom", grid=TRUE,  
 star.char="",stars='raw') +  
 ggtitle("A: Differences from balance") +  
 theme(plot.title = element\_text(hjust=0, vjust=0.5, size=11),  
 plot.margin=unit(c(9,15,0,9), 'pt'))  
sub <- match(with(subset(datG, trt=="Control"),subclass),  
 with(subset(datG, trt=="Treat"),subclass))  
datGpaired <- cbind(subset(datG, trt=="Treat"),  
 with(subset(datG, trt=="Control")[sub,],  
 cbind("Cre78log"=re78log,"CPropScore"=PropScore)))  
gg2 <- ggplot(datGpaired)+  
 geom\_point(aes(PropScore,I(re78log-Cre78log)), size=1)+  
 geom\_smooth(aes(PropScore,I(re78log-Cre78log)), method = "gam",   
 formula = y ~ s(x, bs = "cs")) +  
 xlab("Propensity score for treated")+  
 ylab("Treatment vs control differences") +  
 ggtitle("B: Treatment vs control differences") +  
 theme(plot.title = element\_text(hjust=0, vjust=0.5, size=11),  
 plot.margin=unit(c(9,9,0,15), 'pt'))  
grid.arrange(gg1, gg2, ncol=2)

library(sandwich)  
datG.lm <- lm(re78log ~ trt, data = datG, weights = weights)  
## With 1:1 matching, the weights argument is not really needed  
## Print first two coefficients only.   
lmtest::coeftest(datG.lm, vcov. = vcovCL, cluster = ~subclass)[1:2,]  
## Check number whose income was greater than 0  
datG.glm <- glm(I(re78>0) ~ trt, data = datG, weights = weights, family=binomial)  
lmtest::coeftest(datG.glm, vcov. = vcovCL, cluster = ~subclass)[1:2,]

##### Alternative matching approaches

#### Subsection 9.7.5: Coarsened exact matching

form <- trt ~ age + educ + ethnicid + marr + nodeg + z75 + re75log  
match5.cem <- matchit(formula=form, data=nswG, method="cem", cutpoints=5)  
datcem5 <- match.data(match5.cem)  
match6.cem <- matchit(formula=form, data=nswG, method="cem", cutpoints=6)  
datcem6 <- match.data(match6.cem)  
## Show the effect of adding another cutpoint  
match5.cem  
match6.cem

library(sandwich)  
datcem5.lm <- lm(re78log ~ trt, data = datcem5, weights = weights)  
## The function vcovHC() provides cluster robust standard errors  
lmtest::coeftest(datcem5.lm, vcov. = vcovHC)  
## Estimate treatment effect on number with some earnings:  
datcem6.glm <- glm(I(re78>0) ~ trt, data = datcem6, weights = weights,  
family=binomial)  
lmtest::coeftest(datcem6.glm, vcov. = vcovHC)

### Section 9.8: Multiple imputation

suppressPackageStartupMessages(library(mice))  
Boys <- with(subset(mice::boys, age>=9),   
 data.frame(age=age, loghgt=log(hgt), logbmi=log(bmi), loghc=log(hc)))  
(Pattern <- md.pattern(Boys, plot=F))

set.seed(31) # Set to reproduce result shown  
PatternB <- rbind(Pattern[-c(1,nrow(Pattern)), -ncol(Pattern)],  
 c(0,1,1,1), c(0,1,0,0), c(0,0,1,0))  
boys <- rbind(ic(Boys),   
 ampute(cc(Boys), pattern=PatternB, freq=c(.3,.15,.15,.2,.1,.1),   
 prop=0.75)$amp)  
md.pattern(boys, plot=FALSE)

set.seed(17) # Set to reproduce result shown  
out <- capture.output( # Evaluate; send screen output to text string  
 boys.mids <- mice(boys, method='pmm', m=8) )  
impDFs <- complete(boys.mids, action='all') # Returns a list of m=8 dataframes  
## Average over imputed dataframes (use for exploratory purposes only)  
impArray <- sapply(impDFs, function(x)as.matrix(x), simplify='array')  
boysAv <- as.data.frame(apply(impArray, 1:2, mean))

fits <- with(boys.mids, lm(logbmi~age+loghgt))  
pool.coef <- summary(pool(fits)) # Include in table below

## 2) Regression that leaves out rows with NAs  
omitNArows.coef <- coef(summary(lm(logbmi~age+loghgt, data=boys)))  
## 3) Regression fit to average over data frames after imputation  
boysAv.coef <- coef(summary(lm(logbmi~age+loghgt, data=boysAv)))  
## 4) Fit to original data, with 36 rows had missing data  
Orig.coef <- coef(summary(lm(logbmi ~ age+loghgt, data=Boys)))

ctab <- cbind(summary(pool(fits))[,2:3], omitNArows.coef[,1:2], boysAv.coef[,1:2],   
 Orig.coef[,1:2])  
tab <- setNames(cbind(ctab[,c(1,3,5,7)], ctab[,c(2,4,6,8)]),  
 paste0(rep(c('Est','SE'), c(4,4)), rep(1:4, 2)))  
round(tab,3)

##### Time series cross-sectional data – an example

airquality <- datasets::airquality  
airq <- cbind(airquality[, 1:4], day=1:nrow(airquality))  
 # 'day' (starting May 1) replaces columns 'Month' & 'Day')  
## Replace `Ozone` with `rt4ozone`:  
airq <- cbind(rt4ozone=airq$Ozone^0.25, airq[,-1])

## Generate the scatterplot matrix, now with `rt4ozone` replacing `Ozone`  
smoothPars <- list(col.smooth='red', lty.smooth=2, spread=0)  
car::spm(airq, cex.labels=1.2, regLine=FALSE, col='blue',   
 oma=c(1.95,3,4, 3), gap=.25, smooth=smoothPars)

airq.imp <- mice(airq, m=20, print=FALSE)  
 ## 20 imputations shows up issues of concern very clearly

## Code for figure  
out <- micemd::overimpute(airq.imp)

##### Some further points

### Section 9.9: Further reading

##### Data with more variables than observations

##### Causal inference

##### Multiple imputation

### Section 9.10: Exercises

9.3

library(DAAG)  
oz::oz(sections=c(3:5, 11:16))  
names(possumsites)[1:2] <- c("long", "lat")  
with(possumsites, {  
points(long, lat);  
text(long, lat, row.names(possumsites), pos=c(2,4,2,2,4,2,2))  
})

9.7

data(wine, package='gclus')  
mat <- with(wine,   
 round(1-cor(cbind(Alcohol, Malic, Magnesium, Phenols, Flavanoids)),2))  
colnames(mat) <- rownames(mat) <- 1:5  
print(mat)

9.9a

`confusion` <-  
function(actual, predicted, digits=4){  
 tab <- table(actual, predicted)  
 confuse <- apply(tab, 1, function(x)x/sum(x))  
 print(round(confuse, digits))  
 acc <- sum(tab[row(tab)==col(tab)])/sum(tab)  
 invisible(print(c("Overall accuracy" = round(acc,digits))))  
}  
data(Vehicle, package="mlbench")  
lhat <- MASS::lda(Class ~ ., data=Vehicle, CV=TRUE)$class  
qhat <- MASS::qda(Class ~ ., data=Vehicle, CV=TRUE)$class  
DAAG::confusion(Vehicle$Class, lhat)  
DAAG::confusion(Vehicle$Class, qhat)  
randomForest::randomForest(Class ~ ., data=Vehicle, CV=TRUE)

9.9c

Vehicle.lda <- MASS::lda(Class ~ ., data=Vehicle)  
twoD <- predict(Vehicle.lda)$x  
ggplot2::quickplot(twoD[,1], twoD[,2], color=Vehicle$Class,  
 geom=c("point","density2d"))

9.10

library(ape); library(MASS)  
library(DAAGbio)  
primates.dna <- as.DNAbin(primateDNA)  
primates.dist <- dist.dna(primates.dna, model="K80")  
primates.cmd <- cmdscale(primates.dist)  
eqscplot(primates.cmd)  
rtleft <- c(4,2,4,2)[unclass(cut(primates.cmd[,1], breaks=4))]  
text(primates.cmd, labels=row.names(primates.cmd), pos=rtleft)

d <- dist(primates.cmd)  
sum((d-primates.dist)^2)/sum(primates.dist^2)

9.11

library(DAAG)  
pacific.dist <- dist(x = as.matrix(rockArt[-c(47,54,60,63,92),28:641]),   
 method = "binary")  
sum(pacific.dist==1)/length(pacific.dist)  
plot(density(pacific.dist, to = 1))  
## Check that all columns have at least one distance < 1  
symmat <- as.matrix(pacific.dist)  
table(apply(symmat, 2, function(x) sum(x<1)))  
pacific.cmd <- cmdscale(pacific.dist)  
pacific.sam <- sammon(pacific.dist)

9.15

Wine <- setNames(cbind(stack(wine, select=2:14), rep(wine[,-1], 13)),  
 c("value", "measure", "Class"))  
bwplot(measure ~ value, data=Wine)

wine.pr <- prcomp(wine[,-1], scale=TRUE)  
round(wine.pr$sdev,2)  
t(round(wine.pr$rotation[,1:2],2))  
scores <- as.data.frame(cbind(predict(wine.pr), Class=wine[,1]))  
xyplot(PC2 ~ PC1, groups=Class, data=scores, aspect='iso',   
 par.settings=simpleTheme(pch=16), auto.key=list(columns=3))

library(MASS)  
wine.lda <- lda(Class ~ ., data=wine)  
wineCV.lda <- lda(Class ~ ., data=wine, CV=T)  
t(round(wine.lda$scaling,2))  
tab <- table(wine$Class, wineCV.lda$class,   
 dnn=c('Actual', 'Predicted'))   
tab  
setNames(round(1-sum(diag(tab))/sum(tab),4), "CV error rate")  
scores <- as.data.frame(cbind(predict(wine.lda)$x, Class=wine[,1]))  
xyplot(LD2 ~ LD1, groups=Class, data=scores, aspect='iso',   
 par.settings=simpleTheme(pch=16), auto.key=list(columns=3))

wine$Class <- factor(Wine$Class)  
wine.rf <- randomForest(x=wine[,-1], y=wine$Class)

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/ch9.R")  
}

# 7. Appendix A: The R System – A Brief Overview

On options for working with the code see the vignettes [Ch1-Learning](PGRcode/Ch1-Learning.html) and [UsingCode](PGRcode/UsingCode.html).

##### Packages required (plus any dependencies)

DAAG dplyr tidyr tibble MASS gplots plotrix latticeExtra RColorBrewer

Additionally, knitr and Hmisc are required in order to process the qmd source file.

## The R System – A Brief Overview

### Section 10.1: Getting started with R

#### Subsection 10.1.1: Learn by typing code at the command line

> ## Arithmetic calculations. See the help page `?Arithmetic` {-}  
> 2\*3+10 # The symbol `\*` denotes 'multiply'

> ## Use the `c()` function to join results into a numeric vector  
> c(sqrt(10), 2\*3+10, sqrt(10), 2^3) # 2^3 is 2 to the power of 3  
> ## R knows about pi  
> 2\*pi\*6378 # Approximate circumference of earth at equator (km)

?help # Get information on the use of `help()`  
?sqrt # Or, type help(sqrt)  
?Arithmetic # See, in similar vein ?Syntax  
?'<' # `?Comparison` finds the same help page

## Two commands on one line; Use ';' as separator  
2\*3\*4+10; sqrt(10) ## Try also `cat(2\*3\*4+10, sqrt(10), sep='n')  
## Convert CO2 carbon emissions from tonnes of carbon to tonnes of CO2   
3.664\*c(.53, 2.56, 9.62) ## Data are for 1900, 1960 & 2020

## Use `cat()` to print several items, with control of formatting  
cat(2\*3\*4+10, sqrt(10), '\n')

##### Assignment

## Convert from amounts of carbon to amounts of CO2 (billions of tonnes)  
## and assign result to a named object  
fossilCO2vals <- c(.53, 2.56, 9.62)\*3.664 # Amounts in 1900, 1960, and 2020  
 # Equivalently `fossilCO2vals <- c(.53, 2.56, 9.62)\*rep(3.664,3)`  
## To assign and print, enclose in round brackets  
(fossilCO2vals <- c(.53, 2.56, 9.62)\*3.664)

3.664\*c(.53,2.56, 9.62) -> fossilCO2vals

##### Entry of data at the command line, a graphics formula, and a graph

Year <- c(1900, 1920, 1940, 1960, 1980, 2000, 2020)  
CO2 <- c(.53,.96,1.32,2.56,5.32,6.95,9.62)\*3.664  
## Now plot Carbon Dioxide emissions as a function of Year  
plot(CO2 ~ Year, pch=16, fg="gray")

##### Grouping vectors togeher into data frames

CO2byYear <- data.frame(year=Year, co2gas=CO2)  
CO2byYear # Display the contents of the data frame.  
rm(Year, CO2) # Optionally, remove `Year` and `Carbon` from the workspace  
plot(co2gas ~ year, data=CO2byYear, pch=16)

sqrt(10) # Number of digits is determined by current seting  
options(digits=2) # Change until further notice,  
sqrt(10)

##### Wide-ranging information access and searches

### Section 10.2: R data structures

#### Subsection 10.2.1: Vectors, dates, and arrays

vehicles <- c("Compact", "Large", "Midsize", "Small", "Sporty", "Van")  
c(T, F, F, F, T, T, F) # A logical vector, assuming F=FALSE and T=TRUE

## Character vector  
mammals <- c("Rat", "Pig", "Rat", "Mouse", "Pig")  
## Logical vector  
rodent <- c("TRUE", "FALSE", "TRUE", "FALSE", "TRUE", "FALSE")  
## From character vector `mammals`, create factor  
mfac <- factor(mammals)  
levels(mfac)   
table(mfac)

##### Dates

day1 <- as.Date(c("2022-01-01", "2022-02-01", "2022-03-01"))  
as.numeric(day1) # Days since 1 January 1970  
day1[3] - day1[2]

##### The use of square brackets to extract subsets of vectors

## Specify the indices of the elements that are to be extracted  
x <- c(3, 11, 8, 15, 12,18)  
x[c(1,4:6)] # Elements in positions 1, 4, 5, and 6  
## Use negative indices to identify elements for omission  
x[-c(2,3)] # Positive and negative indices cannot be mixed  
## Specify a vector of logical values.   
x > 10 # This generates a vector of logical values  
x[x > 10]

bodywt <- c(Cow=465, Goat=28, Donkey=187, Pig=192)  
bodywt[c("Goat", "Pig")]

##### Matrices and arrays

arr123 <- array(1:24, dim=c(2,4,3))  
## This prints as three 2 by 4 matrices. Print just the first of the three.  
arr123[, 2, 1] # Column 2 and index 1 of 3rd dimension  
attributes(arr123)

#### Subsection 10.2.2: Factors

gender <- c(rep("male",691), rep("female",692))  
gender <- factor(gender) # From character vector, create factor  
levels(gender) # Notice that `female` comes first

Gender <- factor(gender, levels=c("male", "female"))

mf1 <- factor(rep(c('male','female'),c(2,3)), labels=c("f", "m"))  
## The following has the same result  
mf2 <- factor(rep(c('male','female'), c(2,3)))  
levels(mf2) <- c("f","m") # Assign new levels  
if(all(mf1==mf2))print(mf1)

sum(gender=="male")

table(chickwts$feed) # feed is a factor  
source <- chickwts$feed   
levels(source) <- c("milk","plant","plant","meat","plant","plant")  
table(source)

##### Ordered factors

stress <- rep(c("low","medium","high"), 2)  
ord.stress <- ordered(stress, levels=c("low", "medium", "high"))  
ord.stress  
ord.stress >= "medium"

##### Missing values in values of factors

#### Subsection 10.2.3: Operations with data frames

Cars93sum <- DAAG::Cars93.summary # Create copy in workspace  
Cars93sum

Cars93sum[4:6, 2:3] # Extract rows 4 to 6 and columns 2 and 3  
Cars93sum[6:4, ] # Extract rows in the order 6, 5, 4  
Cars93sum[, 2:3] # Extract columns 2 and 3  
## Or, use negative integers to specify rows and/or columns to be omitted  
Cars93sum[-(1:3), -c(1,4)] # In each case, numbers must be all +ve or all -ve  
## Specify row and/or column names  
Cars93sum[c("Small","Sporty","Van"), c("Max.passengers","No.of.cars")]

##### Data frames vs matrices

names(Cars93sum)[3] <- "numCars"  
names(Cars9sum) <- c("minPass","maxPass","numCars","code")

##### Using a data frame as a database – with() and within()

## trees (datasets) has data on Black Cherry Trees  
with(trees, round(c(mean(Girth), median(Girth), sd(Girth)),1))

with(DAAG::pair65, # stretch of rubber bands  
 {lenchange = heated-ambient  
 c(mean(lenchange), median(lenchange))  
})

## Add variables `mph` and `gradient` to `DAAG::nihills`  
nihr <- within(DAAG::nihills, {mph <- dist/time; gradient <- climb/dist})

##### Extracting rows from data frames

unlist(Cars93sum[1, ])

#### Subsection 10.2.4: Data manipulation functions used in earlier chapters

## For columns of `DAAG::jobs`, show the range of values  
sapply(DAAG::jobs, range)  
## Split egg lengths by species, calculate mean, sd, and number for each  
with(DAAG::cuckoos, sapply(split(length,species),   
 function(x)c(av=mean(x), sd=sd(x), nobs=length(x))))

apply(UCBAdmissions, 3, function(x)(x[1,2]/(x[1,2]+x[2,2]))\*100) # Females  
apply(UCBAdmissions, 3, function(x)(x[1,1]/(x[1,1]+x[2,1]))\*100) # Males

UCBAdmissions[, , 1]

DAAG::cricketer |> dplyr::count(year, left, name="Freq") -> handYear  
names(handYear)[2] <- "hand"  
byYear <- tidyr::pivot\_wider(handYear, names\_from='hand', values\_from="Freq")

#### Subsection 10.2.5: Writing data to a file, and reading data from a file

CO2byYear <- data.frame(year = seq(from=1900, to=2020, by=20),  
 co2gas = c(1.94, 3.52, 4.84, 9.38, 19.49, 25.46, 35.25))  
write.table(CO2byYear, file='gas.txt') # Write data frame to file  
CO2byYear <- read.table(file="gas.txt") # Read data back in  
write.csv(CO2byYear, file='gas.csv') # Write data frame  
CO2byYear <- read.csv(file="gas.csv", row.names=1) # Read data back in

##### Data input from the RStudio menu — data frames vs tibbles

#### Subsection 10.2.6: Issues for working with data frames and tibbles

##### Extraction of columns from data frames and tibbles

sites <- DAAG::possumsites # sites is then a data frame  
sites[,3] # returns a vector  
sites[,3, drop=FALSE] # returns a 1-column data frame

dplyr::as\_tibble(sites)[,3] # returns a 1-column tibble  
dplyr::as\_tibble(sites)[[3]] # returns a vector  
sites[[3]] # returns a vector

##### Conversion between data frames and tibbles

attributes(DAAG::possumsites)[['row.names']]

possumSites <- tibble::as\_tibble(DAAG::possumsites, rownames="Site")  
possumSites

#### Subsection 10.2.7: Lists

## Summary statistics for 31 felled black cherry tree  
## Median (middle value), range, number, units  
htstats <- list(med=76, range=c(low=63,high=87), n=31, units="ft")  
htstats[1:2] # Show first two list elements only

## The following are alternative ways to extract the second list element  
htstats[2] # First list element (Can replace `2` by 'range')  
htstats[2][1] # A subset of a list is a list

htstats[[2]]; htstats$range; htstats[["range"]]

unlist(htstats[2]) # Contents of second list element, with composite names  
unlist(htstats[2], use.names=F) # Elements have no names

tstats <- with(MASS::shoes, t.test(B, A, paired=TRUE))  
names(tstats) ## Names of list elements. See `?t.test` for details.  
tstats[1] ## Type tstats[1] to see the first list element  
## Compact listing of contents list elements 1 to 5, which are all numeric  
unlist(tstats[1:5]) ## With `unlist(tstats)` all elements become character

### Section 10.3: Functions and operators

#### Subsection 10.3.1: Common useful built-in functions

## Data indices  
length() # number of elements in a vector or a list  
order() # x[order(x)] sorts x (by default, NAs are last)  
which() # which indices of a logical vector are `TRUE`  
which.max() # locates (first) maximum (NB, also: `which.min()`)

## Data manipulation  
c() # join together (`concatenate`) elements or vectors or lists  
diff() # vector of first differences  
sort() # sort elements into order, by default omitting NAs  
rev() # reverse the order of vector elements  
t() # transpose matrix or data frame   
 # (a data frame is first coerced to a matrixwith()   
with() # do computation using columns of specified data frame

## Data summary  
mean() # mean of the elements of a vector  
median() # median of the elements of a vector  
range() # minimum and maximum value elements of vector  
unique() # form the vector of distinct values  
## List function arguments  
args() # information on the arguments to a function

## Obtain details  
head() # display first few rows (by default 6) of object  
ls() # list names of objects in the workspace

## Print multiple objects  
cat() # prints multiple objects, one after the other

## Functions that return TRUE or FALSE?  
all() # returns TRUE if all values are TRUE  
any() # returns TRUE if any values are TRUE  
is.factor() # returns TRUE if the argument is a factor  
is.na() # returns TRUE if the argument is an NA  
 # NB also is.logical(), etc.

seq(from =1, by=2, length.out=3) # Unabbeviated arguments  
seq(from =1, by=2, length=3) # Abbreviate `length.out` to `length`

#### Subsection 10.3.2: User-written functions

distance <- c(148,182,173,166,109,141,166)  
mean.and.sd(distance)

## Execute the function with the default argument:  
mean.and.sd()

## Thus, to return the mean, SD and name of the input vector  
## replace c(mean=av, SD=sdev) by  
list(mean=av, SD=sdev, dataset = deparse(substitute(x)))

#### Subsection 10.3.3: Generic functions, and the class of an object

#### Subsection 10.3.4: Pipes — a “Do this, then this, . . .” syntax

mean(rnorm(20, sd=2))  
20 |> rnorm(sd=2) |> mean()

logmammals <- MASS::mammals |> log() |> setNames(c("logbody","logbrain"))  
## Alternatively, use the ability to reverse the assignment operator.  
MASS::mammals |> log() |> setNames(c("logbody","logbrain")) -> logmammals   
 ## This last is more in the spirit of pipes.

MASS::mammals |>  
 log() |>  
 setNames(c("logbody","logbrain")) |>  
 (\(d)lm(logbrain ~ logbody, data=d))() |>  
 coef()

#### Subsection 10.3.5: Operators

## Multiple of divisor that leaves smallest non-negative remainder  
c("Multiple of divisor" = 24 %/% 9, "Remainder after division" = 6)

### Section 10.4: Calculations with matrices, arrays, lists, and data frames

##### Calculations in parallel across all elements of a vector

x <- 1:6  
log(x) # Natural logarithm of 1, 2, ... 6  
log(x, base=10) # Common logarithm (base 10)  
log(64, base=c(2,10)) # Apply different bases to one number  
log(matrix(1:6, nrow=2), base=2) # Take logarithms of all matrix elements

##### Patterned data

seq(from=5, to=22, by=3) # The first value is 5.  
rep(c(2,3,5), 4) # Repeat the sequence (2, 3, 5) four times over  
rep(c("female", "male"), c(2,3)) # Use syntax with a character vector

#### Subsection 10.4.1: Missing values

nbranch <- subset(DAAG::rainforest, species=="Acacia mabellae")$branch  
nbranch # Number of small branches (2cm or less)

mean(nbranch, na.rm=TRUE)

nbranch == NA # This always equals `NA`

is.na(nbranch) # Use to check for NAs

nbranch[is.na(nbranch)] <- -999  
 # `mean(nbranch)` will then be a nonsense value

##### NAs in modeling functions

options()$na.action # Version 3.2.2, following startup

##### Counting and identifying NAs – the use of table()

with(DAAG::nswdemo, table(trt, re74>0, useNA="ifany"))

##### Infinities and NaNs

summary(DAAG::primates)

primates <- DAAG::primates

gplots::plotCI() # `plotCI() function in package `gplots`  
plotrix::plotCI() # `plotCI() function in package `plotrix`

sessionInfo()[['basePkgs']]

## List just the workspace and the first eight packages on the search list:  
search()[1:9]

data(package="datasets")

### Section 10.5: Brief notes on R graphics packages and functions

#### Subsection 10.5.1: Lattice graphics — a step beyond base graphics

grog <- DAAG::grog  
chr <- with(grog, match(Country, c('Australia', 'NewZealand')))   
 # Australia: 1; matches 1st element of c('Australia', 'NewZealand')  
 # NewZealand: 2; matches 2nd element  
plot(Beer ~ Year, data=grog, ylim=c(0, max(Beer)\*1.1), pch = chr)  
with(grog, points(Wine ~ Year, pch=chr, col='red'))  
legend("bottomright", legend=c("Australia", "New Zealand"), pch=1:2)  
title(main="Beer consumption (l, pure alcohol)", line=1)

library(latticeExtra) ## Loads both lattice and the add-on latticeExtra  
gph <- xyplot(Beer+Wine ~ Year, groups=Country, data=grog)  
update(gph, par.settings=simpleTheme(pch=19), auto.key=list(columns=2))

## Or, condition on `Country`  
xyplot(Beer+Wine+Spirit ~ Year | Country, data=grog,  
 par.settings=simpleTheme(pch=19), auto.key=list(columns=3))

tinting <- DAAG::tinting  
xyplot(csoa~it | tint\*target, groups=agegp, data=tinting, auto.key=list(columns=2))

cuckoos <- DAAG::cuckoos  
av <- with(cuckoos, aggregate(length, list(species=species), FUN=mean))  
gph <- dotplot(species ~ length, data=cuckoos, alpha=0.4) +  
 as.layer(dotplot(species ~ x, pch=3, cex=1.4, col="black", data=av))  
update(gph, xlab="Length of egg (mm)")

## Alternatives, using `layer()` or `as.layer()`  
avg <- with(cuckoos, data.frame(nspec=1:nlevels(species),   
 av=sapply(split(length,species),mean)))  
dotplot(species ~ length, data=cuckoos) +   
 layer(lpoints(nspec~av, pch=3, cex=1.25, col="black"), data=avg)

dotplot(species ~ length, data=cuckoos) +   
 as.layer(dotplot(nspec~av, data=avg, pch=3, cex=1.25, col="black"))

## Specify panel function  
dotplot(species ~ length, data=cuckoos,   
 panel=function(x,y,...){panel.dotplot(x, y, pch=1, col="gray40")  
 avg <- data.frame(nspec=1:nlevels(y), av=sapply(split(x,y),mean))  
 with(avg, lpoints(nspec~av, pch=3, cex=1.25, col="black")) })

##### Combining separately created graphics objects

cuckoos <- DAAG::cuckoos  
## Panel A: Dotplot without species means added  
gphA <- dotplot(species ~ length, data=cuckoos)   
## Panel B: Box and whisker plot  
gphB <- bwplot(species ~ length, data=cuckoos)  
update(c("A: Dotplot"=gphA, "B: Boxplot"=gphB), between=list(x=0.4),  
 xlab="Length of egg (mm)", layout=c(2,1))  
 ## `latticeExtra::c()` joins compatible plots together.   
 ## `layout=c(2,1)` : join horizontally; `layout=c(1,2)` : join vertically

#### Subsection 10.5.2: Dynamic graphics – the rgl package

vignette('plot3D', package='plot3D')

### Section 10.6: Plotting characters, symbols, line types and colors

ycol <- -2.1 - (0:9) \* 2.1  
ftype <- c("plain", "bold", "italic", "bold italic", "symbol")  
yline <- 4.2  
ypmax <- 20  
farleft <- -7.8  
plot(c(-4, 31), c(4.25, ypmax), type = "n", xlab = "", ylab = "",  
axes = F)  
chh <- par()$cxy[2]  
text(0:25, rep(ypmax + 0.8 \* chh, 26), paste(0:25), srt = 90,  
cex = 0.75, xpd = T)  
text(-1.5, ypmax + 0.8 \* chh, "pch = ", cex = 0.75, xpd = T)  
points(0:25, rep(ypmax, 26), pch = 0:25, lwd=0.8)  
letterfont <- function(ypos = ypmax, font = 2) {  
par(font = font)  
text(-1.35, ypos, "64-76", cex = 0.75, adj = 1, xpd = TRUE)  
text(19 - 1.35, ypos, "96-108", cex = 0.75, adj = 1)  
points(c(0:12), rep(ypos, 13), pch = 64:76)  
points(19:31, rep(ypos, 13), pch = 96:108)  
text(farleft, ypos, paste(font), xpd = T)  
text(farleft, ypos - 0.5, ftype[font], cex = 0.75)  
}  
plotfont <- function(xpos = 0:31, ypos = ypmax, font = 1,  
sel32 = 2:4, showfont = TRUE) {  
par(font = font)  
i <- 0  
for (j in sel32) {  
i <- i + 1  
maxval <- j \* 32 - 1  
if(j==4)maxval <- maxval-1  
text(-1.35, ypos - i + 1, paste((j - 1) \* 32, "-",  
maxval, sep = ""), cex = 0.75, adj = 1, xpd = TRUE)  
if(j!=4)  
points(xpos, rep(ypos - i + 1, 32), pch = (j - 1) \*  
32 + (0:31))  
else  
points(xpos[-32], rep(ypos - i + 1, 31), pch = (j - 1) \*  
32 + (0:30))  
}  
lines(rep(-1.05, 2), c(ypos - length(sel32) + 1, ypos) +  
c(-0.4, 0.4), xpd = T, col = "grey40")  
if (showfont) {  
text(farleft, ypos, paste("font =", font, " "), xpd = T)  
text(farleft, ypos - 0.5, ftype[font], cex = 0.75,  
xpd = T)  
}  
}  
plotfont(ypos = ypmax - 1.5, font = 1, sel32 = 2:4)  
for (j in 2:4) letterfont(ypos = ypmax - 2.1 - 1.4 \* j, font = j)  
plotfont(ypos = ypmax - 9.1, font = 5, sel32 = 3)  
plotfont(xpos = c(-0.5, 1:31), ypos = ypmax - 10.1, font = 5,  
sel32 = 4, showfont = FALSE)  
par(font = 1)  
ltypes <- c("blank", "solid", "dashed", "dotted", "dotdash",  
"longdash", "twodash")  
lcode <- c("", "", "44", "13", "1343", "73", "2262")  
for (i in 0:6) {  
lines(c(4, 31), c(yline + 4.5 - 0.8 \* i, yline + 4.5 -  
0.8 \* i), lty = i, lwd = 2, xpd = T)  
if (i == 0)  
numchar <- paste("lty =", i, " ")  
else numchar <- i  
text(farleft, yline + 4.5 - 0.8 \* i, numchar, xpd = TRUE)  
text(farleft + 3.5, yline + 4.5 - 0.8 \* i, ltypes[i +  
1], cex = 0.85, xpd = TRUE)  
text(farleft + 7.5, yline + 4.5 - 0.8 \* i, lcode[i +  
1], cex = 0.85, xpd = TRUE)  
}

ycol <- -2.1 - (0:9) \* 2.1  
ftype <- c("plain", "bold", "italic", "bold italic", "symbol")  
yline <- 4.2  
ypmax <- 20  
farleft <- -7.8  
plot(c(-4, 31), c(4.25, ypmax), type = "n", xlab = "", ylab = "",  
axes = F)  
chh <- par()$cxy[2]  
text(0:25, rep(ypmax + 0.8 \* chh, 26), paste(0:25), srt = 90,  
cex = 0.75, xpd = T)  
text(-1.5, ypmax + 0.8 \* chh, "pch = ", cex = 0.75, xpd = T)  
points(0:25, rep(ypmax, 26), pch = 0:25, lwd=0.8)  
letterfont <- function(ypos = ypmax, font = 2) {  
par(font = font)  
text(-1.35, ypos, "64-76", cex = 0.75, adj = 1, xpd = TRUE)  
text(19 - 1.35, ypos, "96-108", cex = 0.75, adj = 1)  
points(c(0:12), rep(ypos, 13), pch = 64:76)  
points(19:31, rep(ypos, 13), pch = 96:108)  
text(farleft, ypos, paste(font), xpd = T)  
text(farleft, ypos - 0.5, ftype[font], cex = 0.75)  
}  
plotfont <- function(xpos = 0:31, ypos = ypmax, font = 1,  
sel32 = 2:4, showfont = TRUE) {  
par(font = font)  
i <- 0  
for (j in sel32) {  
i <- i + 1  
maxval <- j \* 32 - 1  
if(j==4)maxval <- maxval-1  
text(-1.35, ypos - i + 1, paste((j - 1) \* 32, "-",  
maxval, sep = ""), cex = 0.75, adj = 1, xpd = TRUE)  
if(j!=4)  
points(xpos, rep(ypos - i + 1, 32), pch = (j - 1) \*  
32 + (0:31))  
else  
points(xpos[-32], rep(ypos - i + 1, 31), pch = (j - 1) \*  
32 + (0:30))  
}  
lines(rep(-1.05, 2), c(ypos - length(sel32) + 1, ypos) +  
c(-0.4, 0.4), xpd = T, col = "grey40")  
if (showfont) {  
text(farleft, ypos, paste("font =", font, " "), xpd = T)  
text(farleft, ypos - 0.5, ftype[font], cex = 0.75,  
xpd = T)  
}  
}  
plotfont(ypos = ypmax - 1.5, font = 1, sel32 = 2:4)  
for (j in 2:4) letterfont(ypos = ypmax - 2.1 - 1.4 \* j, font = j)  
plotfont(ypos = ypmax - 9.1, font = 5, sel32 = 3)  
plotfont(xpos = c(-0.5, 1:31), ypos = ypmax - 10.1, font = 5,  
sel32 = 4, showfont = FALSE)  
par(font = 1)  
ltypes <- c("blank", "solid", "dashed", "dotted", "dotdash",  
"longdash", "twodash")  
lcode <- c("", "", "44", "13", "1343", "73", "2262")  
for (i in 0:6) {  
lines(c(4, 31), c(yline + 4.5 - 0.8 \* i, yline + 4.5 -  
0.8 \* i), lty = i, lwd = 2, xpd = T)  
if (i == 0)  
numchar <- paste("lty =", i, " ")  
else numchar <- i  
text(farleft, yline + 4.5 - 0.8 \* i, numchar, xpd = TRUE)  
text(farleft + 3.5, yline + 4.5 - 0.8 \* i, ltypes[i +  
1], cex = 0.85, xpd = TRUE)  
text(farleft + 7.5, yline + 4.5 - 0.8 \* i, lcode[i +  
1], cex = 0.85, xpd = TRUE)  
}

##### Font families

##### Colors

library(RColorBrewer)  
palette(brewer.pal(12, "Set3"))

if(file.exists("/Users/johnm1/pkgs/PGRcode/inst/doc/")){  
code <- knitr::knit\_code$get()  
txt <- paste0("\n## ", names(code),"\n", sapply(code, paste, collapse='\n'))  
writeLines(txt, con="/Users/johnm1/pkgs/PGRcode/inst/doc/Appendix.R")  
}