## section 14.3

rb <- function(th,k=2) { ## Rosenbrock function

v <- k\*(th[2]-th[1]^2)^2 + (1-th[1])^2

g <- c(-2\*(1-th[1])-k\*4\*th[1]\*(th[2]-th[1]^2),k\*2\*(th[2]-th[1]^2))

h <- matrix(0,2,2)

h[1,1] <- 2-k\*2\*(2\*(th[2]-th[1]^2) - 4\*th[1]^2)

h[2,2] <- 2\*k;h[1,2] <- h[2,1] <- -4\*k\*th[1]

list(v=v,g=g,H=h)

}

th <- c(-1,2)

r <- rb(th)

for (i in 1:100) {

eh <- eigen(r$H); eh$values <- abs(eh$values) ## ensure Hessian +ve def

d <- -drop(eh$vectors %\*% ((t(eh$vectors) %\*% r$g)/eh$values)) ## step = -H^{-1}g

for (j in 1:30) {

r0 <- rb(th+d)

if (r0$v<=r$v) {

th <- th + d

r <- r0

break

} else d <- d/2

}

if (r0$v>r$v) {

cat("Step failure"); break

}

if (max(abs(r$g))<1e-8) break

}

th; r$g

## Section 14.4

nll <- function(theta,t,y) {

## -ve log likelihood for AIDS model y\_i ~ Poi(alpha\*exp(beta\*t\_i))

## theta = (alpha,beta)

mu <- theta[1] \* exp(theta[2] \* t) ## mu = E(y)

-sum(dpois(y,mu,log=TRUE)) ## the negative log likelihood

} ## nll

gll <- function(theta,t,y) {

## grad of -ve log lik of Poisson AIDS early epidemic model

alpha <- theta[1];beta <- theta[2] ## enhances readability

ebt <- exp(beta\*t) ## avoid computing twice

-c(sum(y)/alpha - sum(ebt), ## -dl/dalpha

sum(y\*t) - alpha\*sum(t\*ebt)) ## -dl/dbeta

} ## gll

hll <- function(theta,t,y) {

## Hessian of -ve log lik of Poisson AIDS early epidemic model

alpha <- theta[1];beta <- theta[2] ## enhances readability

ebt <- exp(beta\*t) ## avoid computing twice

H <- matrix(0,2,2) ## matrix for Hessian of -ve ll

H[1,1] <- sum(y)/alpha^2

H[2,2] <- alpha\*sum(t^2\*ebt)

H[1,2] <- H[2,1] <- sum(t\*ebt)

H

} ## hll

nll2 <- function(theta,t,y) {

## wrapper function for nll and its grad and Hessian,

## suitable for optimization by nlm

z <- nll(theta,t,y) ## the objective

attr(z,"gradient") <- gll(theta,t,y)

attr(z,"hessian") <- hll(theta,t,y)

z

} ## nll2

t80 <- 1:13 ## years since 1980

y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240) ## AIDS cases

th0 <- c(1,1)

fit <- nlm(nll2,th0,y=y,t=t80)

plot(t80+1980,y,xlab="year",ylab="AIDS cases")

mu <- fit$estimate[1]\*exp(fit$estimate[2]\*t80)

lines(t80+1980,mu)

## slightly more efficient version of nll2...

nll1 <- function(theta,t,y) {

alpha <- theta[1]; beta <- theta[2]

ebt <- exp(beta\*t)

mu <- alpha \* ebt ## mu = E(y)

nll <- -sum(dpois(y,mu,log=TRUE)) ## the negative log likelihood

sum.y <- sum(y)

sum.tebt <- sum(t\*ebt)

attr(nll,"gradient") <- -c(sum.y/alpha - sum(ebt), ## -dl/dalpha

sum(y\*t) - alpha\*sum.tebt) ## -dl/dbeta

H <- matrix(0,2,2) ## matrix for Hessian of -ve ll

H[1,1] <- sum.y/alpha^2

H[2,2] <- alpha\*sum(t^2\*ebt)

H[1,2] <- H[2,1] <- sum.tebt

attr(nll,"hessian") <- H

nll

} ## nll1

## unit test...

test <- nll1(th0,t80,y)

true <- nll2(th0,t80,y)

all.equal(test,true)

## Section 14.6

nlli <- deriv( ## what to differentiate...

expression(-y\*(log(alpha)+beta\*t)+alpha\*exp(beta\*t)+lgamma(y+1)),

c("alpha","beta"), ## differentiate w.r.t. these

function.arg=c("alpha","beta","t","y"))## return function - with these args

nllo <- function(th,t,y) { ## evaluate -ve log lik

sum(nlli(th[1],th[2],t,y))

} ## nllo

nllg <- function(th,t,y) { ## evaluate grad

colSums(attr(nlli(th[1],th[2],t,y),"gradient"))

}

fit1 <- optim(th0,nllo,nllg,y=y,t=t80,method="BFGS")

nllg(fit1$par,t80,y)

## 16.2 Belgian AIDS data

y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)

t <- -6:6 ## year zero is 1987

## MH sampler for model Y ~ Poi(N\_0 exp(r t))

ns <- 20000 ## samples

th <- matrix(0,3,ns)

thp <- th[,1] <- c(48,.23,0) ## guess initial by data inspection

ll <- sum(dpois(y,thp[1]\*exp(thp[2]\*t+thp[3]\*t^2),log=TRUE))

rsd <- c(.01,.005)

u <- runif(ns); accept <- 0

for (i in 2:ns) {

thp <- th[,i-1] + c(sample(-4:4,1),rnorm(2)\*rsd)

llp <- sum(dpois(y,thp[1]\*exp(thp[2]\*t+thp[3]\*t^2),log=TRUE))

if (u[i]<exp(llp-ll)\*(thp[1]>0)) {

accept <- accept + 1

th[,i] <- thp

ll <- llp

} else th[,i] <- th[,i-1]

}

accept/ns

## check...

par(mfrow=c(3,1))

for (i in 1:3) plot(th[i,],type="l")

th <- th[,-(1:1000)] ## discard burn-in

thm <- rowMeans(th) ## posterior mean of params

## plot...

par(mfrow=c(1,1))

plot(t,y); lines(t,thm[1]\*exp(thm[2]\*t+thm[3]\*t^2))

## CI for expected rate...

mu <- apply(th,2,function(x,t) x[1]\*exp(x[2]\*t+x[3]\*t^2),t=t)

mul <- apply(mu,1,quantile,prob=c(.025,.975))

plot(t+1987,y,ylim=range(rbind(mul,y)),ylab="AIDS case rate",xlab="year");

lines(t+1987,thm[1]\*exp(thm[2]\*t+thm[3]\*t^2))

lines(t+1987,mul[1,],lty=2);lines(t+1987,mul[2,],lty=2)

## 16.3 Same again in JAGS...

y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)

t <- -6:6 ## year zero is 1987

setwd("~sw283/lnotes/StatProg/sp-notes/solutions") ## EDIT: where is JAGS file?

jaid <- jags.model("aids.jags",data=list(t=t,y=y))

sam <- coda.samples(jaid,c("th","mu"),n.iter=20000)

effectiveSize(sam);autocorr.plot(sam)

plot(sam[,1:4]) ## etc

mum <- as.matrix(sam)[,1:13]

mul <- apply(mum,2,quantile,prob=c(.025,.5,.975))

plot(t+1987,y,ylim=range(rbind(mul,y)),ylab="AIDS case rate",xlab="year");

lines(t+1987,mul[2,])

lines(t+1987,mul[1,],lty=2);lines(t+1987,mul[3,],lty=2)

## 16.1.1

n <- 60

nrep <- 1000

nb <- 1000

q.true <- diff(qnorm(c(.25,.75)))

pcp <- bcp <- 0

qb <- rep(0,nb)

for (j in 1:nrep) {

y <- rnorm(n) ## simulated sample

q <- as.numeric(diff(quantile(y,c(.25,.75)))) ## sample IQR

for (i in 1:nb) { ## bootstrap loop, computing BS IQR

qb[i] <- as.numeric(diff(quantile(sample(y,n,replace=TRUE),c(.25,.75))))

}

pci <- quantile(qb,c(.025,.975)) ## percentile interval

b <- q - pci[1]; c <- pci[2] - q ## get upper and lower interval margins

if (q-c < q.true && q+b > q.true) bcp <- bcp + 1 ## basic interval ok?

if (pci[1] < q.true && pci[2] > q.true) pcp <- pcp + 1 ## percentile ok?

}

pcp/nrep;bcp/nrep ## coverage probabilities

## So neither is great here - the percentile interval exludes the truth only

## half as often as it should, and the basic interval excludes the truth twice

## as often as it should. But given that the intervals are the same length,

## the percentile version is clearly preferable in this case.

## 16.1.2...

n <- 100; x <- runif(n); mu <- .2+x/2+2\*x^2;

n.reps <- 200; bt <- c(.2,.5,2)

cp0 <- cp <- rep(0,3)

for (r in 1:n.reps) {

y <- rpois(n,mu)

dat <- data.frame(x=x,y=y)

b <- lm(y~x+I(x^2),data=dat)

bc <- summary(b)$coefficients

ci0 <- rbind(bc[,1]-1.96\*bc[,2],bc[,1]+1.96\*bc[,2])

cp0 <- cp0 + as.numeric(ci0[1,]<= bt&ci0[2,]>=bt)

nb <- 1000; bs <- matrix(0,3,nb)

for (i in 1:nb) {

bs[,i] <- coef(lm(y~x+I(x^2),data=dat[sample(1:n,n,replace=TRUE),]))

}

ci <- apply(bs,1,quantile,probs=c(.025,.975))

cp <- cp + as.numeric(ci[1,]<= bt&ci[2,]>=bt)

}

cp/n.reps;cp0/n.reps

## Note that the standard intervals for the mean are far too wide here

## overcovering substantially