Package ‘gamair’

October 13, 2022

Version 1.0-2

Author Simon Wood <`simon.wood@r-project.org`>

Maintainer Simon Wood <`simon.wood@r-project.org`>

Title Data for 'GAMs: An Introduction with R'

Description Data sets and scripts used in the book 'Generalized Additive Models: An Introduction with R', Wood (2006,2017) CRC.

Depends R (>= 2.10)

Suggests mgcv, lattice, MASS, nlme, lme4, geoR, survival

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-23 12:40:02 UTC

R topics documented:

`gamair-package` *Data and scripts for 'Generalized Additive Models: An Introduction with R'*

Description

This package contains the data sets used in the book *Generalized Additive Models: An Introduction with R*, which covers linear and generalized linear models, GAMs as implemented in package `mgcv` and mixed model extensions of these.

There are help files containing the R code for each chapter and its exercise solutions, for the second edition of the book.

The script files for the first edition of the book can be found in the 'scripts' folder of the 'inst' folder of the source package. They have been modified slightly to work with recent versions of mgcv (e.g. >= 1.7-0).

Details

Each dataset has its own help page, which describes the dataset, and gives the original source and associated references. All datasets have been reformatted into standard R data frames. Some smaller datasets from the book have not been included. Datasets from other R packages have not been included, with the exception of a distillation of one set from the `NMMAPSdata` package.

Index:

aral	Aral sea chlorophyll
aral.bnd	Aral sea boundary
bird	Bird distribution data from Portugal
blowfly	Nicholson's Blowfly data
bone	Bone marrow treatment survival data.
brain	Brain scan data
cairo	Daily temperature data for Cairo
CanWeather	Canadian annual temperature curves
chicago	Chicago air pollution and death rate data
chl	Chlorophyll data
co2s	Atmospheric CO2 at South Pole
coast	European coastline from -11 to 0 East and from 43 to 59 North.
engine	Engine wear versus size data
german.polys.rda	Polygons defining german local regions
gamair	Generalized Additive Models: An Introduction With R
harrier	Hen Harriers Eating Grouse
hubble	Hubble Space Telescope Data
ipo	Initial Public Offering Data
larynx	Cancer of the larynx in Germany
mack	Egg data from 1992 mackerel survey
mackp	Prediction grid data for 1992 mackerel egg model
med	2010 mackerel egg survey data
meh	2010 horse mackerel egg survey data
prostate	Protein mass spectra for prostate diagnosis
sitka	Sitka spruce growth and ozone data
sole	Sole Eggs in the Bristol Channel
sperm.comp1	Sperm competition data I
sperm.comp2	Sperm competition data II
swer	Swiss extreme ranfall data
stomata	Artifical stomatal area data
wesdr	Diabetic retinopathy data
wine	Bordeaux Wines

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2006,2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`

Examples

```
library(help=gamair)
```

`aral` *Aral sea remote sensed chlorophyll data*

Description

SeaWifs satellite chlorophyll measurements for the 38th 8-day observation period of the year in the Aral sea, averaged over 1998-2002, along with an Aral sea boundary file.

Usage

```
data(aral)
data(aral.bnd)
```

Format

The `aral` data frame has the following columns

lon longitude of pixel or boundary vertex.

lat latitude of pixel or boundary vertex.

chl chlorophyll measurement

exra The highest rainfall observed in any 12 hour period in that year, in mm.

Details

Trying to smooth the data with a conventional smoother, such as a thin plate spline, leads to linkage between the two arms of the sea, which is clearly an artefact. A soap film smoother avoids this problem.

Source

`https://seawifs.gsfc.nasa.gov/`

Examples

```
require(gamair);require(mgcv)
data(aral); data(aral.bnd)

## define some knots...
knt <- list(lon=c(58.55,59.09,59.36,59.64,59.91,60.18,58.27,58.55,59.09,
59.36,59.64,59.91,60.18,60.45,58.27,58.55,58.82,59.09,59.36,59.64,59.91,
60.18,60.45,58.27,58.55,59.36,59.64,59.91,60.18,58.55,59.36,59.64,59.91,
60.18,58.55,58.82,59.36,59.64,59.91,60.18,60.45,58.82,59.09,59.64,59.91,
60.18,59.64),
lat=c(44.27,44.27,44.27,44.27,44.27,44.27,44.55,44.55,44.55,44.55,44.55,
44.55,44.55,44.55,44.82,44.82,44.82,44.82,44.82,44.82,44.82,44.82,44.82,
45.09,45.09,45.09,45.09,45.09,45.09,45.36,45.36,45.36,45.36,45.36,45.64,
45.64,45.64,45.64,45.64,45.64,45.64,45.91,45.91,45.91,45.91,45.91,46.18))

## fit soap film...
b <-  gam(chl~s(lon,lat,k=30,bs="so",xt=list(bnd=list(aral.bnd),
                nmax=150)),knots=knt,data=aral)

## plot results...
plot(b)
```

`bird` *Bird distribution data from Portugal*

Description

Data from the compilation of the Portuguese Atlas of Breeding Birds.

Usage

```
data(bird)
```

Format

A data frame with 6 columns and 25100 rows. Each row refers to one 2km by 2km square (tetrad). The columns are:

QUADRICULA An identifier for the 10km by 10km square that this tetrad belongs to.

TET Which tetrad the observation is from.

crestlark Were crested lark (or possibly thekla lark!) found (1), not found (0) breading in this tetrad, or was the tetrad not visited (`NA`).

linnet As `crestlark`, but for linnet.

x location of tetrad (km east of an origin).

y location of tetrad (km north of an origin).

Details

At least 6 tetrads from each 10km square were visited, to establish whether each species was breeding there, or not. Each Tetrad was visited twice for one hour each visit. These data are not definitive: at time of writing the fieldwork was not quite complete.

The data were kindly supplied by Jose Pedro Granadeiro.

Source

The Atlas of the Portuguese Breeding Birds.

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R

Examples

```
data(bird)
species <- "crestlark"
op<-par(bg="white",mfrow=c(1,1),mar=c(5,5,1,1))
ind <- bird[[species]]==0&!is.na(bird[[species]])
plot(bird$y[ind]/1000,1000-bird$x[ind]/1000,pch=19,cex=.3,col="white",
   ylab="km west",xlab="km north",cex.lab=1.4,cex.axis=1.3,type="n")
polygon(c(4000,4700,4700,4000),c(250,250,600,600),col="grey",border="black")
points(bird$y[ind]/1000,1000-bird$x[ind]/1000,pch=19,cex=.3,col="white")
ind <- bird[[species]]==1&!is.na(bird[[species]])
with(bird,points(y[ind]/1000,1000-x[ind]/1000,pch=19,cex=.3))
par(op)
```

`blowfly` *Nicholson's Blowfly data*

Description

Data on a laboratory population of Blowfies, from the classic ecological studies of Nicholson.

Usage

`data(blowfly)`

Format

A data frame with 2 columns and 180 rows. The columns are:

pop Counts (!) of the population of adult Blowflies in one of the experiments.

day Day of experiment.

Details

The population counts are actually obtained by counting dead blowflies and back calculating.

References

Nicholson, A.J. (1954a) Compensatory reactions of populations to stresses and their evolutionary significance. Australian Journal of Zoology 2, 1-8.

Nicholson, A.J. (1954b) An outline of the dynamics of animal populations. Australian Journal of Zoology 2, 9-65.

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R

Examples

```
data(blowfly)
with(blowfly,plot(day,pop,type="l"))
```

`bone` *Bone marrow treatemtn survival data*

Description

Data from Klein and Moeschberger (2003), for 23 patients with non-Hodgkin's lymphoma.

Usage

```
data(bone)
```

Format

A data frame with 3 columns and 23 rows. Each row refers to one patient. The columns are:

t Time of death, relapse or last follow up after treatment, in days.

d 1 for death or relapse. 0 otherwise.

trt 2 level factor. `allo` or `auto` depending on treatment recieved.

Details

The data were collected at the Ohio State University bone marrow transplant unit. The `allo` treatment is bone marrow transplant from a matched sibling donor. The `auto` treatment consists of bone marrow removal and replacement after chemotherapy.

Source

Klein and Moeschberger (2003).

References

Klein and Moeschberger (2003) Survival Analysis: techniques for censored and truncated data.

Wood, S.N. (2017) Generalized Additive Models: An Introduction with R

Examples

```
## example of fitting a Cox PH model as a Poisson GLM...
## First a function to convert data frame of raw data
## to data frame of artificial data...

psurv <- function(surv,time="t",censor="d",event="z") {
## create data frame to fit Cox PH as Poisson model.
## surv[[censor]] should be 1 for event or zero for censored.
  if (event %in% names(surv)) warning("event name already in use in data frame")
  surv <- as.data.frame(surv)[order(surv[[time]]),] ## ascending time order
  et <- unique(surv[[time]][surv[[censor]]==1]) ## unique times requiring record
  es <- match(et,surv[[time]]) ## starts of risk sets in surv
  n <- nrow(surv); t <- rep(et,1+n-es) ## times for risk sets
  st <- cbind(0,surv[unlist(apply(matrix(es),1,function(x,n) x:n,n=n)),])
  st[st[[time]]==t&st[[censor]]!=0,1] <- 1 ## signal events
  st[[time]] <- t ## reset time field to time for this risk set
  names(st)[1] <- event
  st
} ## psurv

## Now fit the model...
require(gamair)
data(bone);bone$id <- 1:nrow(bone)
pb <- psurv(bone); pb$tf <- factor(pb$t)
## Note that time factor tf should go first to ensure
## it has no contrasts applied...
b <- glm(z ~ tf + trt - 1,poisson,pb)
drop1(b,test="Chisq") ## test for effect - no evidence

## martingale and deviance residuals
chaz <- tapply(fitted(b),pb$id,sum) ## cum haz by subject
mrsd <- bone$d - chaz
drsd <- sign(mrsd)*sqrt(-2*(mrsd + bone$d*log(chaz)))

## Estimate and plot survivor functions and standard
## errors for the two groups...

te <- sort(unique(bone$t[bone$d==1])) ## event times

## predict survivor function for "allo"...
pd <- data.frame(tf=factor(te),trt=bone$trt[1])
fv <- predict(b,pd)
H <- cumsum(exp(fv)) ## cumulative hazard
plot(stepfun(te,c(1,exp(-H))),do.points=FALSE,ylim=c(0,1),xlim=c(0,550),
     main="black allo, grey auto",ylab="S(t)",xlab="t (days)")
## add s.e. bands...
X <- model.matrix(~tf+trt-1,pd)
J <- apply(exp(fv)*X,2,cumsum)
se <- diag(J%*%vcov(b)%*%t(J))^.5
lines(stepfun(te,c(1,exp(-H+se))),do.points=FALSE,lty=2)
lines(stepfun(te,c(1,exp(-H-se))),do.points=FALSE,lty=2)
```

```
## same for "auto"...
pd <- data.frame(tf=factor(te),trt=bone$trt[23])
fv <- predict(b,pd); H <- cumsum(exp(fv))
lines(stepfun(te,c(1,exp(-H))),col="grey",lwd=2,do.points=FALSE)
X <- model.matrix(~tf+trt-1,pd)
J <- apply(exp(fv)*X,2,cumsum)
se <- diag(J%*%vcov(b)%*%t(J))^.5
lines(stepfun(te,c(1,exp(-H+se))),do.points=FALSE,lty=2,col="grey",lwd=2)
lines(stepfun(te,c(1,exp(-H-se))),do.points=FALSE,lty=2,col="grey",lwd=2)
```

`brain` *Brain scan data*

Description

Functional magnetic resonance imaging measurements for a human brain subject to a particular experimental stimulus. One slice of the image is provided, described as a near-axial slice through the dorsal cerebral cortex.

Usage

```
data(brain)
```

Format

A data frame with 5 columns and 1567 rows. Each row refers to one 'voxel' of the image. The columns are:

X voxel position on horizontal axis.

Y voxel position on vertical axis.

medFPQ median of three replicate 'Fundamental Power Quotient' values at the voxel: this is the main measurement of brain activivity.

region code indicating which of several regions of the brain the voxel belongs to. The regions are defined by the experimenters. `0` is the base region; `1` is the region of interest; `2` is the region activated by the experimental stimulus; `NA` denotes a voxel with no allocation.

meanTheta mean phase shift at the Voxel, over three measurments.

Details

See the source article for fuller details.

Source

S. Landau et al (2003) 'Tests for a difference in timing of physiological response between two brain regions measured by using functional magnetic resonance imaging'. Journal of the Royal Statistical Society, Series C, Applied Statistics, 53(1):63-82

`cairo` *Daily temperature data for Cairo*

Description

The average air temperature (F) in Cairo from Jan 1st 1995.

Usage

```
data(cairo)
```

Format

A data frame with 6 columns and 3780 rows. The columns are:

month month of year from 1 to 12.

day.of.month day of month, from 1 to 31.

year Year, starting 1995.

temp Average temperature (F).

day.of.year Day of year from 1 to 366.

time Number of days since 1st Jan 1995.

Source

http://academic.udayton.edu/kissock/http/Weather/citylistWorld.htm

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R

Examples

```
data(cairo)
with(cairo,plot(time,temp,type="l"))
```

`CanWeather` *Canadian Weather data*

Description

Data on temperature throughout the year at 35 Canadian locations, originally form the `fda` package.

Usage

```
data(canWeather)
```

Format

The `CanWeather` data frame has the following 5 columns

time Day of year from 1 to 365.

T Mean temperature for that day in centigrade.

region A four level factor classifiying locations as `Arctic`, `Atlantic`, `Continental` or `Pacific`.

latitude Degrees north of the equator.

place A factor with 35 levels: the names of each locagtion.

Details

The data provide quite a nice application of function on scalar regression. Note that the data are for a single year, so will not generally be cyclic.

Source

Data are from the `fda` package.

https://cran.r-project.org/package=fda

References

Ramsay J.O. and B.W. Silverman (2006) Functional data analysis (2nd ed). Springer

Examples

```
require(gamair);require(mgcv)
data(canWeather)
reg <- unique(CanWeather$region)
place <- unique(CanWeather$place)
col <- 1:4;names(col) <- reg
for (k in 1:35) {
  if (k==1) plot(1:365,CanWeather$T[CanWeather$place==place[k]],
            ylim=range(CanWeather$T),type="l",
    col=col[CanWeather$region],xlab="day",ylab="temperature") else
    lines(1:365,CanWeather$T[CanWeather$place==place[k]],
            col=col[CanWeather$region[CanWeather$place==place[k]]])
```

```
}

## Function on scalar regression.
## T(t) = f_r(t) + f(t)*latitude + e(t)
## where e(t) is AR1 Gaussian and f_r is
## a smooth for region r.
## 'rho' chosen to minimize AIC or (-ve) REML score.

b <- bam(T ~ region + s(time,k=20,bs="cr",by=region) +
         s(time,k=40,bs="cr",by=latitude),
         data=CanWeather,AR.start=time==1,rho=.97)

## Note: 'discrete==TRUE' option even faster.

par(mfrow=c(2,3))
plot(b,scale=0,scheme=1)
acf(b$std)
```

`ch1` *Code for Chapter 1: Linear Models*

Description

R code from Chapter 1 of the second edition of 'Generalized Additive Models: An Introduction with R' is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch1.solutions`

Examples

```
library(gamair); library(mgcv)

## 1.1.2
data(hubble)
hub.mod <- lm(y ~ x - 1, data=hubble)
summary(hub.mod)
plot(fitted(hub.mod),residuals(hub.mod),xlab="fitted values",
```

```
     ylab="residuals")
hub.mod1 <- lm(y ~ x - 1,data=hubble[-c(3,15),])
summary(hub.mod1)
plot(fitted(hub.mod1),residuals(hub.mod1),
     xlab="fitted values",ylab="residuals")
hubble.const <- c(coef(hub.mod),coef(hub.mod1))/3.09e19
age <- 1/hubble.const
age/(60^2*24*365)

## 1.1.3
cs.hubble <- 163000000
t.stat <- (coef(hub.mod1)-cs.hubble)/vcov(hub.mod1)[1,1]^0.5
pt(t.stat,df=21)*2 # 2 because of |T| in p-value defn.

sigb <- summary(hub.mod1)$coefficients[2]
h.ci <- coef(hub.mod1)+qt(c(0.025,0.975),df=21)*sigb
h.ci
h.ci <- h.ci*60^2*24*365.25/3.09e19 # convert to 1/years
sort(1/h.ci)

## 1.5.1
data(sperm.comp1)
pairs(sperm.comp1[,-1])
sc.mod1 <- lm(count~time.ipc+prop.partner,sperm.comp1)
model.matrix(sc.mod1)
par(mfrow=c(2,2))  # split the graphics device into 4 panels
plot(sc.mod1)      # (uses plot.lm as sc.mod1 is class `lm')
sperm.comp1[9,]
sc.mod1
sc.mod2 <- lm(count~time.ipc+I(prop.partner*time.ipc),
              sperm.comp1)

## 1.5.2
summary(sc.mod1)

## 1.5.3
sc.mod3 <- lm(count~prop.partner,sperm.comp1)
summary(sc.mod3)
sc.mod4 <- lm(count~1,sperm.comp1) # null model
AIC(sc.mod1,sc.mod3,sc.mod4)

## 1.5.4
data(sperm.comp2)
sc2.mod1 <- lm(count~f.age+f.height+f.weight+m.age+m.height+
               m.weight+m.vol,sperm.comp2)
plot(sc2.mod1)
summary(sc2.mod1)
sc2.mod2 <- lm(count~f.age+f.height+f.weight+m.height+
               m.weight+m.vol,sperm.comp2)
summary(sc2.mod2)
sc2.mod7 <- lm(count~f.weight,sperm.comp2)
summary(sc2.mod7)
sc <- sperm.comp2[-19,]
```

```
sc3.mod1 <- lm(count~f.age+f.height+f.weight+m.age+m.height+
               m.weight+m.vol,sc)
summary(sc3.mod1)
sperm.comp1$m.vol <-
   sperm.comp2$m.vol[sperm.comp2$pair%in%sperm.comp1$subject]
sc1.mod1 <- lm(count~m.vol,sperm.comp1)
summary(sc1.mod1)

## 1.5.5
sc.c <- summary(sc1.mod1)$coefficients
sc.c   # check info extracted from summary
sc.c[2,1]+qt(c(.025,.975),6)*sc.c[2,2]

## 1.5.6
df <- data.frame(m.vol=c(10,15,20,25))
predict(sc1.mod1,df,se=TRUE)

## 1.5.7
set.seed(1); n <- 100; x <- runif(n)
z <- x + rnorm(n)*.05
y <- 2 + 3 * x + rnorm(n)
summary(lm(y~z))
summary(lm(y~x+z))

## 1.6.4
z <- c(1,1,1,2,2,1,3,3,3,3,4)
z
z <- as.factor(z)
z
x <- c("A","A","C","C","C","er","er")
x
x <- factor(x)
x
PlantGrowth$group
# PlantGrowth$group <- as.factor(PlantGrowth$group)
pgm.1 <- lm(weight ~ group,data=PlantGrowth)
plot(pgm.1)
summary(pgm.1)
pgm.0 <- lm(weight~1,data=PlantGrowth)
anova(pgm.0,pgm.1)
```

`ch1.solutions` *Solution code for Chapter 1: Linear Models*

Description

R code for Chapter 1 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch1`

Examples

```
library(gamair); library(mgcv)

## Q.8 Rubber
## a)
library(MASS)
m1 <- lm(loss~hard+tens+I(hard*tens)+I(hard^2)+I(tens^2)+
I(hard^2*tens)+I(tens^2*hard)+I(tens^3)+I(hard^3),Rubber)
plot(m1)    ## residuals OK
summary(m1) ## p-values => drop I(tens^2*hard)
m2 <- update(m1,.~.-I(tens^2*hard))
summary(m2)
m3 <- update(m2,.~.-hard)
summary(m3)
m4 <- update(m3,.~.-1)
summary(m4)
m5 <- update(m4,.~.-I(hard^2))
summary(m5) ## p-values => keep all remaining
plot(m5)    ## residuals OK

## b)
AIC(m1,m2,m3,m4,m5)
m6 <- step(m1)

## c)
m <- 40;attach(Rubber)
mt <- seq(min(tens),max(tens),length=m)
mh <- seq(min(hard),max(hard),length=m)
lp <- predict(m6,data.frame(hard=rep(mh,rep(m,m)),
                            tens=rep(mt,m)))
contour(mt,mh,matrix(lp,m,m),xlab="tens",ylab="hard")
points(tens,hard)
detach(Rubber)

## Q.9 warpbreaks
wm <- lm(breaks~wool*tension,warpbreaks)
par(mfrow=c(2,2))
plot(wm)    # residuals OK
```

```
anova(wm)
## ... so there is evidence for a wool:tension interaction.
par(mfrow=c(1,1))
with(warpbreaks,interaction.plot(tension,wool,breaks))

## Q.10 cars
## a)
cm1 <- lm(dist ~ speed + I(speed^2),cars)
summary(cm1)
## Intercept has very high p-value, so drop it
cm2 <- lm(dist ~ speed + I(speed^2)-1,cars)
summary(cm2)
## both terms now significant, but try the alternative of
## dropping `speed'
cm3 <- lm(dist ~ I(speed^2),cars)
AIC(cm1,cm2,cm3)
plot(cm2)
# Clearly cm2, with speed and speed squared terms, is to be preferred,
# but note that variance seems to be increasing with mean a little:
# perhaps a GLM, better?

## b)
# In seconds, the answer is obtained as follows..
b <- coef(cm2)
5280/(b[1]*60^2)
# This is a long time, but would have a rather wide associated confidence
# interval.

## Q.11 QR
# The following is a version of the function that you should end up with.

fitlm <- function(y,X)
{ qrx <- qr(X)                ## get QR decomposition
  y <- qr.qty(qrx,y)          ## form Q'y efficiently
  R <- qr.R(qrx)              ## extract R
  p <- ncol(R);n <- length(y) ## get dimensions
  f <- y[1:p]; r <- y[(p+1):n]## partition Q'y
  beta <- backsolve(R,f)      ## parameter estimates (a)
  sig2 <- sum(r^2)/(n-p)      ## resid variance estimate (c)
  Ri <- backsolve(R,diag(ncol(R))) ## inverse of R matrix
  Vb <- Ri%*%t(Ri)*sig2       ## covariance matrix
  se <- diag(Vb)^.5           ## standard errors (c)
  F.ratio <- f^2/sig2         ## sequential F-ratios
  seq.p.val <- 1-pf(F.ratio,1,n-p) ## seq. p-values (e)
  list(beta=beta,se=se,sig2=sig2,seq.p.val=seq.p.val,df=n-p)
} ## fitlm

# The following code uses the function to answer some of the question parts.

## get example X ...
X <- model.matrix(dist ~ speed + I(speed^2),cars)
cm <- fitlm(cars$dist,X) # used fitting function
cm$beta;cm$se            # print estimates and s.e.s (a,c)
```

```
cm1<-lm(dist ~ speed + I(speed^2),cars) # equiv. lm call
summary(cm1)             # check estimates and s.e.s (b,c)
t.ratio <- cm$beta/cm$se # form t-ratios
p.val <- pt(-abs(t.ratio),df=cm$df)*2
p.val                    # print evaluated p-values (d)
## print sequential ANOVA p-values, and check them (e)
cm$seq.p.val
anova(cm1)

## Q.12 InsectSprays
X <- model.matrix(~spray-1,InsectSprays)
X <- cbind(rep(1,nrow(X)),X)   # redundant model matrix
C <- matrix(c(0,rep(1,6)),1,7) # constraints
qrc <- qr(t(C))                # QR decomp. of C'
## use fact that Q=[D:Z] and XQ = (Q'X')' to form XZ ...
XZ <- t(qr.qty(qrc,t(X)))[,2:7]
m1 <- lm(InsectSprays$count~XZ-1) # fit model
bz <- coef(m1) # estimates in constrained parameterization
## form b = Z b_z, using fact that Q=[D:Z], again
b <- c(0,bz)
b <- qr.qy(qrc,b)
sum(b[2:7])

## Q.13 trees
## a)
EV.func <- function(b,g,h)
{ mu <- b[1]*g^b[2]*h^b[3]
  J <- cbind(g^b[2]*h^b[3],mu*log(g),mu*log(h))
  list(mu=mu,J=J)
}

## b)
attach(trees)
b <- c(.002,2,1);b.old <- 100*b+100
while (sum(abs(b-b.old))>1e-7*sum(abs(b.old))) {
   EV <- EV.func(b,Girth,Height)
   z <- (Volume-EV$mu) + EV$J%*%b
   b.old <- b
   b <- coef(lm(z~EV$J-1))
}
b

## c)
sig2 <- sum((Volume - EV$mu)^2)/(nrow(trees)-3)
Vb <- solve(t(EV$J)%*%EV$J)*sig2
se <- diag(Vb)^.5;se
```

---

ch2 *Code for Chapter 2: Linear Mixed Models*

Description

R code from Chapter 2 of the second edition of 'Generalized Additive Models: An Introduction with R' is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch2.solutions`

Examples

```
library(gamair); library(mgcv)

## 2.1.1
data(stomata)
m1 <- lm(area ~ CO2 + tree,stomata)
m0 <- lm(area ~ CO2,stomata)
anova(m0,m1)
m2 <- lm(area ~ tree,stomata)
anova(m2,m1)
st <- aggregate(data.matrix(stomata),
                by=list(tree=stomata$tree),mean)
st$CO2 <- as.factor(st$CO2);st
m3 <- lm(area~CO2,st)
anova(m3)
summary(m3)$sigma^2 - summary(m1)$sigma^2/4

## 2.1.3
library(nlme)  # load nlme `library', which contains data
data(Rail)     # load data
Rail
m1 <- lm(travel ~ Rail,Rail)
anova(m1)
rt <- aggregate(data.matrix(Rail),by=list(Rail$Rail),mean)
rt
m0 <- lm(travel ~ 1,rt)   # fit model to aggregated data
sigb <- (summary(m0)$sigma^2-summary(m1)$sigma^2/3)^0.5
# sigb^2 is variance component for rail
sig <- summary(m1)$sigma # sig^2 is resid. var. component
sigb
sig
summary(m0)
```

```
## 2.1.4
library(nlme)
data(Machines)
names(Machines)
attach(Machines)  # make data available without `Machines$'
interaction.plot(Machine,Worker,score)
m1 <- lm(score ~ Worker*Machine,Machines)
m0 <- lm(score ~ Worker + Machine,Machines)
anova(m0,m1)
summary(m1)$sigma^2
Mach <- aggregate(data.matrix(Machines),by=
        list(Machines$Worker,Machines$Machine),mean)
Mach$Worker <- as.factor(Mach$Worker)
Mach$Machine <- as.factor(Mach$Machine)
m0 <- lm(score ~ Worker + Machine,Mach)
anova(m0)
summary(m0)$sigma^2 - summary(m1)$sigma^2/3
M <- aggregate(data.matrix(Mach),by=list(Mach$Worker),mean)
m00 <- lm(score ~ 1,M)
summary(m00)$sigma^2 - (summary(m0)$sigma^2)/3

## 2.4.4
llm <- function(theta,X,Z,y) {
  ## untransform parameters...
  sigma.b <- exp(theta[1])
  sigma <- exp(theta[2])
  ## extract dimensions...
  n <- length(y); pr <- ncol(Z); pf <- ncol(X)
  ## obtain \hat \beta, \hat b...
  X1 <- cbind(X,Z)
  ipsi <- c(rep(0,pf),rep(1/sigma.b^2,pr))
  b1 <- solve(crossprod(X1)/sigma^2+diag(ipsi),
            t(X1)%*%y/sigma^2)
  ## compute log|Z'Z/sigma^2 + I/sigma.b^2|...
  ldet <- sum(log(diag(chol(crossprod(Z)/sigma^2 +
            diag(ipsi[-(1:pf)])))))
  ## compute log profile likelihood...
  l <- (-sum((y-X1%*%b1)^2)/sigma^2 - sum(b1^2*ipsi) -
  n*log(sigma^2) - pr*log(sigma.b^2) - 2*ldet - n*log(2*pi))/2
  attr(l,"b") <- as.numeric(b1) ## return \hat beta and \hat b
  -l
}
library(nlme) ## for Rail data
options(contrasts=c("contr.treatment","contr.treatment"))
Z <- model.matrix(~Rail$Rail-1) ## r.e. model matrix
X <- matrix(1,18,1)             ## fixed model matrix
## fit the model...
rail.mod <- optim(c(0,0),llm,hessian=TRUE,
                           X=X,Z=Z,y=Rail$travel)
exp(rail.mod$par) ## variance components
solve(rail.mod$hessian) ## approx cov matrix for theta
attr(llm(rail.mod$par,X,Z,Rail$travel),"b")
```

```
## 2.5.1
library(nlme)
lme(travel~1,Rail,list(Rail=~1))

## 2.5.2

Loblolly$age <- Loblolly$age - mean(Loblolly$age)
lmc <- lmeControl(niterEM=500,msMaxIter=100)
m0 <- lme(height ~ age + I(age^2) + I(age^3),Loblolly,
          random=list(Seed=~age+I(age^2)+I(age^3)),
          correlation=corAR1(form=~age|Seed),control=lmc)
plot(m0)
m1 <- lme(height ~ age+I(age^2)+I(age^3)+I(age^4),Loblolly,
          list(Seed=~age+I(age^2)+I(age^3)),
          cor=corAR1(form=~age|Seed),control=lmc)
plot(m1)
m2 <- lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
          Loblolly,list(Seed=~age+I(age^2)+I(age^3)),
          cor=corAR1(form=~age|Seed),control=lmc)
plot(m2)
plot(m2,Seed~resid(.))
qqnorm(m2,~resid(.))
qqnorm(m2,~ranef(.))

m3 <- lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
          Loblolly,list(Seed=~age+I(age^2)+I(age^3)),control=lmc)
anova(m3,m2)
m4 <- lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
          Loblolly,list(Seed=~age+I(age^2)),
          correlation=corAR1(form=~age|Seed),control=lmc)
anova(m4,m2)
m5 <- lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
          Loblolly,list(Seed=pdDiag(~age+I(age^2)+I(age^3))),
          correlation=corAR1(form=~age|Seed),control=lmc)
anova(m2,m5)
plot(augPred(m2))

## 2.5.3
lme(score~Machine,Machines,list(Worker=~1,Machine=~1))

## 2.5.4
library(lme4)
a1 <- lmer(score~Machine+(1|Worker)+(1|Worker:Machine),
           data=Machines)
a1
a2 <- lmer(score~Machine+(1|Worker)+(Machine-1|Worker),
           data=Machines)
AIC(a1,a2)
anova(a1,a2)

## 2.5.5
library(mgcv)
b1 <- gam(score~ Machine + s(Worker,bs="re") +
```

```
      s(Machine,Worker,bs="re"),data=Machines,method="REML")
gam.vcomp(b1)
b2 <- gam(score~ Machine + s(Worker,bs="re") +
      s(Worker,bs="re",by=Machine),data=Machines,method="REML")
gam.vcomp(b2)
```

`ch2.solutions` *Solution code for Chapter 2: Linear Mixed Models*

Description

R code for Chapter 2 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch2`

Examples

```
library(gamair); library(mgcv)

## Q.6
## c)
library(nlme)
options(contrasts=c("contr.treatment",
                    "contr.treatment"))
m1 <- lme(Thickness~Site+Source,Oxide,~1|Lot/Wafer)
plot(m1)               # check resids vs. fitted vals
qqnorm(residuals(m1))  # check resids for normality
abline(0,sd(resid(m1)))# adding a "reference line"
qqnorm(m1,~ranef(.,level=1)) # check normality of b_k
qqnorm(m1,~ranef(.,level=2)) # check normality of c_(k)l
m2 <- lme(Thickness~Site+Source,Oxide,~1|Lot)
anova(m1,m2)
anova(m1)
intervals(m1)

## Q.7
```

```
library(nlme)
attach(Machines)
interaction.plot(Machine,Worker,score) # note 6B
## base model
m1<-lme(score~Machine,Machines,~1|Worker/Machine)
## check it...
plot(m1)
plot(m1,Machine~resid(.),abline=0)
plot(m1,Worker~resid(.),abline=0)
qqnorm(m1,~resid(.))
qqnorm(m1,~ranef(.,level=1))
qqnorm(m1,~ranef(.,level=2)) ## note outlier
## try more general r.e. structure
m2<-lme(score~Machine,Machines,~Machine|Worker)
## check it...
qqnorm(m2,~resid(.))
qqnorm(m2,~ranef(.,level=1)) ## still an outlier
## simplified model...
m0 <- lme(score~Machine,Machines,~1|Worker)
## formal comparison
anova(m0,m1,m2) ## m1 most appropriate
anova(m1)     ## significant Machine effect
intervals(m1) ## Machines B and C better than A
## remove problematic worker 6, machine B
Machines <- Machines[-(34:36),]
## re-running improves plots, but conclusions same.

## It seems that (2.6) is the most appropriate model of those tried,
## and broadly the same conclusions are reached with or without
## worker 6, on Machine B, which causes outliers on several checking
## plots. See next question for comparison of machines B and C.

## Q.8
# Using the data without worker 6 machine B:

intervals(m1,level=1-0.05/3,which="fixed")
levels(Machines$Machine)
Machines$Machine <- relevel(Machines$Machine,"B")
m1a <- lme(score ~ Machine, Machines, ~ 1|Worker/Machine)
intervals(m1a,level=1-0.05/3,which="fixed")

# So, there is evidence for differences between machine A and the
# other 2, but not between B and C, at the 5% level. However,
# depending on how economically significant the point estimate of
# the B-C difference is, it might be worth conducting a study with
# more workers in order to check whether the possible small
# difference might actually be real.

## Q.9
## c)
library(nlme);data(Gun)
options(contrasts=c("contr.treatment","contr.treatment"))
with(Gun,plot(Method,rounds))
```

```
with(Gun,plot(Physique,rounds))
m1 <- lme(rounds~Method+Physique,Gun,~1|Team)
plot(m1)              # fitted vs. resid plot
qqnorm(residuals(m1))
abline(0,m1$sigma)  # add line of "perfect Normality"
anova(m1)  # balanced data: don't need type="marginal"
m2 <- lme(rounds~Method,Gun,~1|Team)
intervals(m2)
```

`ch3` *Code for Chapter 3: Generalized Linear Models*

Description

R code from Chapter 3 of the second edition of 'Generalized Additive Models: An Introduction with R' is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch3.solutions`

Examples

```
library(gamair); library(mgcv)

## 3.2.2
x <- c(.6,1.5); y <- c(.02,.9)
ms <- exp(-x*4)   # set initial values at lower left
glm(y ~ I(-x)-1,family=gaussian(link=log),mustart=ms)
ms <- exp(-x*0.1)  # set initial values at upper right
glm(y ~ I(-x)-1,family=gaussian(link=log),mustart=ms)

## 3.3.1

heart <- data.frame(ck = 0:11*40+20,
ha=c(2,13,30,30,21,19,18,13,19,15,7,8),
ok=c(88,26,8,5,0,1,1,1,1,0,0,0))

p <- heart$ha/(heart$ha+heart$ok)
plot(heart$ck,p,xlab="Creatinine kinase level",
```

```
     ylab="Proportion Heart Attack")
mod.0 <- glm(cbind(ha,ok) ~ ck, family=binomial(link=logit),
             data=heart)
mod.0 <- glm(cbind(ha,ok) ~ ck, family=binomial, data=heart)
mod.0
(271.7-36.93)/271.7
1-pchisq(36.93,10)
par(mfrow=c(2,2))
plot(mod.0)
plot(heart$ck, p, xlab="Creatinine kinase level",
     ylab="Proportion Heart Attack")
lines(heart$ck, fitted(mod.0))
mod.2 <- glm(cbind(ha,ok)~ck+I(ck^2)+I(ck^3),family=binomial,
             data=heart)
mod.2
par(mfrow=c(2,2))
plot(mod.2)
par(mfrow=c(1,1))
plot(heart$ck,p,xlab="Creatinine kinase level",
     ylab="Proportion Heart Attack")
lines(heart$ck,fitted(mod.2))
anova(mod.0,mod.2,test="Chisq")

## 3.3.2
y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t <- 1:13
plot(t+1980,y,xlab="Year",ylab="New AIDS cases",ylim=c(0,280))
m0 <- glm(y~t,poisson)
m0
par(mfrow=c(2,2))
plot(m0)
m1 <- glm(y~t+I(t^2),poisson)
plot(m1)
summary(m1)
anova(m0,m1,test="Chisq")
beta.1 <- summary(m1)$coefficients[2,]
ci <- c(beta.1[1]-1.96*beta.1[2],beta.1[1]+1.96*beta.1[2])
ci ## print 95% CI for beta_1
new.t <- seq(1,13,length=100)
fv <- predict(m1,data.frame(t=new.t),se=TRUE)
par(mfrow=c(1,1))
plot(t+1980,y,xlab="Year",ylab="New AIDS cases",ylim=c(0,280))
lines(new.t+1980,exp(fv$fit))
lines(new.t+1980,exp(fv$fit+2*fv$se.fit),lty=2)
lines(new.t+1980,exp(fv$fit-2*fv$se.fit),lty=2)

## 3.3.3
psurv <- function(surv,time="t",censor="d",event="z") {
## create data frame to fit Cox PH as Poisson model.
## surv[[censor]] should be 1 for event or zero for censored.
  if (event %in% names(surv)) warning("event name clashes")
  surv <- as.data.frame(surv)[order(surv[[time]]),] # t order
  et <- unique(surv[[time]][surv[[censor]]==1]) # unique times
```

```
  es <- match(et,surv[[time]]) # starts of risk sets in surv
  n <- nrow(surv); t <- rep(et,1+n-es) # times for risk sets
  st <- cbind(0,
     surv[unlist(apply(matrix(es),1,function(x,n) x:n,n=n)),])
  st[st[[time]]==t&st[[censor]]!=0,1] <- 1 # signal events
  st[[time]] <- t ## reset event time to risk set time
  names(st)[1] <- event
  st
} ## psurv

require(gamair); data(bone); bone$id <- 1:nrow(bone)
pb <- psurv(bone); pb$tf <- factor(pb$t)
b <- glm(z ~ tf + trt - 1,poisson,pb)

chaz <- tapply(fitted(b),pb$id,sum) ## by subject cum. hazard
mrsd <- bone$d - chaz ## Martingale residuals

drop1(b,test="Chisq") ## test for effect - no evidence

te <- sort(unique(bone$t[bone$d==1])) ## event times
## predict survivor function for "allo"...
pd <- data.frame(tf=factor(te),trt=bone$trt[1])
fv <- predict(b,pd)
H <- cumsum(exp(fv)) ## cumulative hazard
plot(stepfun(te,c(1,exp(-H))),do.points=FALSE,ylim=c(0,1),
     xlim=c(0,550),main="",ylab="S(t)",xlab="t (days)")
## add s.e. bands...
X <- model.matrix(~tf+trt-1,pd)
J <- apply(exp(fv)*X,2,cumsum)
se <- diag(J%*%vcov(b)%*%t(J))^.5
lines(stepfun(te,c(1,exp(-H+se))),do.points=FALSE,lty=2)
lines(stepfun(te,c(1,exp(-H-se))),do.points=FALSE,lty=2)

## 3.3.4
al <- data.frame(y=c(435,147,375,134),gender=
   as.factor(c("F","F","M","M")),faith=as.factor(c(1,0,1,0)))
al
mod.0 <- glm(y ~ gender + faith, data=al, family=poisson)
model.matrix(mod.0)
mod.0
fitted(mod.0)
mod.1 <- glm(y~gender*faith,data=al,family=poisson)
model.matrix(mod.1)
mod.1
anova(mod.0,mod.1,test="Chisq")

## 3.3.5
data(sole)
sole$off <- log(sole$a.1-sole$a.0)# model offset term
sole$a<-(sole$a.1+sole$a.0)/2     # mean stage age
solr<-sole                        # make copy for rescaling
solr$t<-solr$t-mean(sole$t)
solr$t<-solr$t/var(sole$t)^0.5
```

```
solr$la<-solr$la-mean(sole$la)
solr$lo<-solr$lo-mean(sole$lo)
b <- glm(eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+I(la^2)
          +I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+I(t^3)+
          I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+I(lo^2*t)+
          I(la^2*t)+I(la*t^2)+I(lo*t^2)+ a +I(a*t)+I(t^2*a),
          family=quasi(link=log,variance="mu"),data=solr)
summary(b)
b1 <- update(b, ~ . - I(lo*t))
b4 <- update(b1, ~ . - I(lo*la*t) - I(lo*t^2) - I(lo^2*t))
anova(b,b4,test="F")
par(mfrow=c(1,2)) # split graph window into 2 panels
plot(fitted(b4)^0.5,solr$eggs^0.5) # fitted vs. data plot
plot(fitted(b4)^0.5,residuals(b4)) # resids vs. sqrt(fitted)

## 3.5.1
rf <- residuals(b4,type="d") # extract deviance residuals
## create an identifier for each sampling station
solr$station <- factor(with(solr,paste(-la,-lo,-t,sep="")))
## is there evidence of a station effect in the residuals?
solr$rf <-rf
rm <- lme(rf~1,solr,random=~1|station)
rm0 <- lm(rf~1,solr)
anova(rm,rm0)
## following is slow...
## Not run:
library(MASS)
form <- eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
            I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+
            I(t^3)+I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+I(lo^2*t)+
            I(la^2*t)+I(la*t^2)+I(lo*t^2)+ # end log spawn
            a +I(a*t)+I(t^2*a)
b <- glmmPQL(form,random=list(station=~1),
            family=quasi(link=log,variance="mu"),data=solr)

summary(b)

form4 <- eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
            I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+
            I(t^3)+I(lo^2*la)+I(lo*la^2)+
            I(la^2*t)+I(lo*t^2)+ # end log spawn
            a +I(a*t)+I(t^2*a)

b4 <- glmmPQL(form4,random=list(station=~1),
            family=quasi(link=log,variance="mu"),data=solr)

fv <- exp(fitted(b4)+solr$off) # note need to add offset
resid <- solr$egg-fv          # raw residuals
plot(fv^.5,solr$eggs^.5)
abline(0,1,lwd=2)
plot(fv^.5,resid/fv^.5)
plot(fv^.5,resid)
fl<-sort(fv^.5)
```

```
## add 1 s.d. and 2 s.d. reference lines
lines(fl,fl);lines(fl,-fl);lines(fl,2*fl,lty=2)
lines(fl,-2*fl,lty=2)

intervals(b4,which="var-cov")

## 3.5.2

form5 <- eggs ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
            I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+
            I(t^3)+I(lo^2*la)+I(lo*la^2)+
            I(la^2*t)+I(lo*t^2)+ # end log spawn
            a +I(a*t)+I(t^2*a) + s(station,bs="re")

b <- gam(form5,family=quasi(link=log,variance="mu"),data=solr,
         method="REML")

## 3.5.3
library(lme4)
solr$egg1 <- round(solr$egg * 5)
form <- egg1 ~ offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
            I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+I(la^3)+
            I(t^3)+I(lo*la*t)+I(lo^2*la)+I(lo*la^2)+I(lo^2*t)+
            I(la^2*t)+I(la*t^2)+I(lo*t^2)+ # end log spawn
            a +I(a*t)+I(t^2*a) + (1|station)

glmer(form,family=poisson,data=solr)

## End(Not run)
```

`ch3.solutions` *Solution code for Chapter 3: Generalized Linear Models*

Description

R code for Chapter 3 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch3`

Examples

```
library(gamair); library(mgcv)

## Q.2 Residuals

n <- 100; m <- 10
x <- runif(n)
lp <- 3*x-1
mu <- binomial()$linkinv(lp)
y <- rbinom(1:n,m,mu)
par(mfrow=c(2,2))
plot(glm(y/m ~ x,family=binomial,weights=rep(m,n)))

## example glm fit...
b <- glm(y/m ~ x,family=binomial,weights=rep(m,n))
reps <- 200;mu <- fitted(b)
rsd <- matrix(0,reps,n) # array for simulated resids
runs <- rep(0,reps)  # array for simulated run counts
for (i in 1:reps) {  # simulation loop
  ys <- rbinom(1:n,m,mu) # simulate from fitted model
  ## refit model to simulated data
  br <- glm(ys/m ~ x,family=binomial,weights=rep(m,n))
  rs <- residuals(br) # simulated resids (meet assumptions)
  rsd[i,] <- sort(rs) # store sorted residuals
  fv.sort <- sort(fitted(br),index.return=TRUE)
  rs <- rs[fv.sort$ix] # order resids by sorted fit values
  rs <- rs > 0         # check runs of +ve, -ve resids
  runs[i] <- sum(rs[1:(n-1)]!=rs[2:n])
}
# plot original ordered residuals, and simulation envelope
for (i in 1:n) rsd[,i] <- sort(rsd[,i])
par(mfrow=c(1,1))
plot(sort(residuals(b)),(1:n-.5)/n) # original
## plot 95% envelope ....
lines(rsd[5,],(1:n-.5)/n);lines(rsd[reps-5,],(1:n-.5)/n)

# compare original runs to distribution under independence
rs <- residuals(b)
fv.sort <- sort(fitted(b),index.return=TRUE)
rs <- rs[fv.sort$ix]
rs <- rs > 0
obs.runs <- sum(rs[1:(n-1)]!=rs[2:n])
sum(runs>obs.runs)

## Q.3 Death penalty

## read in data...
count <- c(53,414,11,37,0,16,4,139)
death <- factor(c(1,0,1,0,1,0,1,0))
defendant <- factor(c(0,0,1,1,0,0,1,1))
victim <- factor(c(0,0,0,0,1,1,1,1))
levels(death) <- c("no","yes")
```

```
levels(defendant) <- c("white","black")
levels(victim) <- c("white","black")

## a)
sum(count[death=="yes"&defendant=="black"])/
    sum(count[defendant=="black"])
sum(count[death=="yes"&defendant=="white"])/
    sum(count[defendant=="white"])

## b)
dm <- glm(count~death*victim+death*defendant+
          victim*defendant,family=poisson(link=log))
summary(dm)
dm0 <- glm(count~death*victim+victim*defendant,
           family=poisson(link=log))
anova(dm0,dm,test="Chisq")

## Q.7 IRLS
y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t <-1:13
X <- cbind(rep(1,13),t,t^2) # model matrix
mu <- y;eta <- log(mu) # initial values
ok <- TRUE
while (ok) {
 ## evaluate pseudodata and weights
 z <- (y-mu)/mu + eta
 w <- as.numeric(mu)
 ## fit weighted working linear model
 z <- sqrt(w)*z; WX <- sqrt(w)*X
 beta <- coef(lm(z~WX-1))
 ## evaluate new eta and mu
 eta.old <- eta
 eta <- X%*%beta
 mu <- exp(eta)
 ## test for convergence...
 if (max(abs(eta-eta.old))<1e-7*max(abs(eta))) ok <- FALSE
}
plot(t,y);lines(t,mu) # plot fit

## Q.8
## b)
data(harrier)
m <- 1
b <- glm(Consumption.Rate~I(1/Grouse.Density^m),
     family=quasi(link=inverse,variance=mu),data=harrier)

## c)
plot(harrier$Grouse.Density,residuals(b))
## clear pattern if $m=1$, and the parameter estimates lead to a rather odd curve.

## d)
## search leads to...
m <- 3.25
```

```
b <- glm(Consumption.Rate~I(1/Grouse.Density^m),
     family=quasi(link=inverse,variance=mu),data=harrier)

## e)
pd <- data.frame(Grouse.Density = seq(0,130,length=200))
pr <- predict(b,newdata=pd,se=TRUE)
with(harrier,plot(Grouse.Density,Consumption.Rate))
lines(pd$Grouse.Density,1/pr$fit,col=2)
lines(pd$Grouse.Density,1/(pr$fit-pr$se*2),col=3)
lines(pd$Grouse.Density,1/(pr$fit+pr$se*2),col=3)

## f)
ll <- function(b,cr,d)
## evalates -ve quasi-log likelihood of model
## b is parameters, cr is consumption, d is density
{ ## get expected consumption...
  dm <- d^b[3]
  Ec <- exp(b[1])*dm/(1+exp(b[1])*exp(b[2])*dm)
  ## appropriate quasi-likelihood...
  ind <- cr>0    ## have to deal with cr==0 case
  ql <- cr - Ec
  ql[ind] <- ql[ind] + cr[ind]*log(Ec[ind]/cr[ind])
  -sum(ql)
}
## Now fit model ...
fit <- optim(c(log(.4),log(10),3),ll,method="L-BFGS-B",
             hessian=TRUE,cr=harrier$Consumption.Rate,
             d=harrier$Grouse.Density)
## and plot results ...
b <- fit$par
d <- seq(0,130,length=200); dm <- d^b[3]
Ec <- exp(b[1])*dm/(1+exp(b[1])*exp(b[2])*dm)
with(harrier,plot(Grouse.Density,Consumption.Rate))
lines(d,Ec,col=2)

## Q.9
death <- as.numeric(ldeaths)
month <- rep(1:12,6)
time <- 1:72
ldm <- glm(death ~ sin(month/12*2*pi)+cos(month/12*2*pi),
           family=poisson(link=identity))
plot(time,death,type="l");lines(time,fitted(ldm),col=2)
summary(ldm)
plot(ldm)

## Q.10
y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t <- 1:13
b <- glm(y ~ t + I(t^2),family=poisson)
log.lik <- b1 <- seq(.4,.7,length=100)
for (i in 1:100)
{ log.lik[i] <- logLik(glm(y~offset(b1[i]*t)+I(t^2),
                           family=poisson))
```

```
}
plot(b1,log.lik,type=”l”)
points(coef(b)[2],logLik(b),pch=19)
abline(logLik(b)[1]-qchisq(.95,df=1),0,lty=2)

## Q.11 Soybean
## a)
library(nlme)
attach(Soybean)
lmc <- lmeControl(niterEM=300) ## needed for convergence
m1<-lme(weight~Variety*Time+Variety*I(Time^2)+
        Variety*I(Time^3),Soybean,~Time|Plot,control=lmc)
plot(m1) ## clear increasing variance with mean

## b)
library(MASS)
m2<-glmmPQL(weight~Variety*Time+Variety*I(Time^2)+
    Variety*I(Time^3),data=Soybean,random=~Time|Plot,
    family=Gamma(link=log),control=lmc)
plot(m2) ## much better

## c)
m0<-glmmPQL(weight~Variety*Time+Variety*I(Time^2)+
    Variety*I(Time^3),data=Soybean,random=~1|Plot,
    family=Gamma(link=log),control=lmc) # simpler r.e.'s
m3<-glmmPQL(weight~Variety*Time+Variety*I(Time^2)+
    Variety*I(Time^3),data=Soybean,random=~Time+
    I(Time^2)|Plot,family=Gamma(link=log),control=lmc)
## ... m3 has more complex r.e. structure
## Following not strictly valid, but gives a rough
## guide. Suggests m2 is best...
AIC(m0,m2,m3)
summary(m2) ## drop Variety:Time
m4<-glmmPQL(weight~Variety+Time+Variety*I(Time^2)+
    Variety*I(Time^3),data=Soybean,random=~Time|Plot,
    family=Gamma(link=log),control=lmc)
summary(m4) ## perhaps drop Variety:I(Time^3)?
m5<-glmmPQL(weight~Variety+Time+Variety*I(Time^2)+
    I(Time^3),data=Soybean,random=~Time|Plot,
    family=Gamma(link=log),control=lmc)
summary(m5) ## don't drop any more
AIC(m2,m4,m5) ## supports m4
intervals(m5,which=”fixed”)

## So m4 or m5 are probably the best models to use, and
## both suggest that variety P has a higher weight on average.
```

---

ch4 *Code for Chapter 4: Introducing GAMs*

Description

R code from Chapter 4 of the second edition of 'Generalized Additive Models: An Introduction with R' is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch4.solutions`

Examples

```
library(gamair); library(mgcv)

## 4.2.1
data(engine); attach(engine)

plot(size,wear,xlab="Engine capacity",ylab="Wear index")

tf <- function(x,xj,j) {
## generate jth tent function from set defined by knots xj
  dj <- xj*0;dj[j] <- 1
  approx(xj,dj,x)$y
}

tf.X <- function(x,xj) {
## tent function basis matrix given data x
## and knot sequence xj
  nk <- length(xj); n <- length(x)
  X <- matrix(NA,n,nk)
  for (j in 1:nk) X[,j] <- tf(x,xj,j)
  X
}

sj <- seq(min(size),max(size),length=6) ## generate knots
X <- tf.X(size,sj)                      ## get model matrix
b <- lm(wear~X-1)                       ## fit model
s <- seq(min(size),max(size),length=200)## prediction data
Xp <- tf.X(s,sj)                        ## prediction matrix
plot(size,wear)                         ## plot data
lines(s,Xp%*%coef(b))                   ## overlay estimated f

## 4.2.2
prs.fit <- function(y,x,xj,sp) {
```

```
  X <- tf.X(x,xj)       ## model matrix
  D <- diff(diag(length(xj)),differences=2) ## sqrt penalty
  X <- rbind(X,sqrt(sp)*D) ## augmented model matrix
  y <- c(y,rep(0,nrow(D))) ## augmented data
  lm(y~X-1)   ## penalized least squares fit
}

sj <- seq(min(size),max(size),length=20) ## knots
b <- prs.fit(wear,size,sj,2) ## penalized fit
plot(size,wear)   ## plot data
Xp <- tf.X(s,sj)  ## prediction matrix
lines(s,Xp%*%coef(b)) ## plot the smooth

## 4.2.3
rho = seq(-9,11,length=90)
n <- length(wear)
V <- rep(NA,90)
for (i in 1:90) { ## loop through smoothing params
  b <- prs.fit(wear,size,sj,exp(rho[i])) ## fit model
  trF <- sum(influence(b)$hat[1:n])      ## extract EDF
  rss <- sum((wear-fitted(b)[1:n])^2)    ## residual SS
  V[i] <- n*rss/(n-trF)^2                ## GCV score
}

plot(rho,V,type="l",xlab=expression(log(lambda)),
                    main="GCV score")
sp <- exp(rho[V==min(V)])     ## extract optimal sp
b <- prs.fit(wear,size,sj,sp) ## re-fit
plot(size,wear,main="GCV optimal fit")
lines(s,Xp%*%coef(b))

## 4.2.3 mixed model connection

## copy of llm from 2.2.4...
llm <- function(theta,X,Z,y) {
  ## untransform parameters...
  sigma.b <- exp(theta[1])
  sigma <- exp(theta[2])
  ## extract dimensions...
  n <- length(y); pr <- ncol(Z); pf <- ncol(X)
  ## obtain \hat \beta, \hat b...
  X1 <- cbind(X,Z)
  ipsi <- c(rep(0,pf),rep(1/sigma.b^2,pr))
  b1 <- solve(crossprod(X1)/sigma^2+diag(ipsi),
              t(X1)%*%y/sigma^2)
  ## compute log|Z'Z/sigma^2 + I/sigma.b^2|...
  ldet <- sum(log(diag(chol(crossprod(Z)/sigma^2 +
              diag(ipsi[-(1:pf)])))))
  ## compute log profile likelihood...
  l <- (-sum((y-X1%*%b1)^2)/sigma^2 - sum(b1^2*ipsi) -
  n*log(sigma^2) - pr*log(sigma.b^2) - 2*ldet - n*log(2*pi))/2
  attr(l,"b") <- as.numeric(b1) ## return \hat beta and \hat b
  -l
```

```
}

X0 <- tf.X(size,sj)            ## X in original parameterization
D <- rbind(0,0,diff(diag(20),difference=2))
diag(D) <- 1                   ## augmented D
X <- t(backsolve(t(D),t(X0)))  ## re-parameterized X
Z <- X[,-c(1,2)]; X <- X[,1:2] ## mixed model matrices
## estimate smoothing and variance parameters...
m <- optim(c(0,0),llm,method="BFGS",X=X,Z=Z,y=wear)
b <- attr(llm(m$par,X,Z,wear),"b") ## extract coefficients
## plot results...
plot(size,wear)
Xp1 <- t(backsolve(t(D),t(Xp))) ## re-parameterized pred. mat.
lines(s,Xp1%*%as.numeric(b),col="grey",lwd=2)

library(nlme)
g <- factor(rep(1,nrow(X)))        ## dummy factor
m <- lme(wear~X-1,random=list(g=pdIdent(~Z-1)))
lines(s,Xp1%*%as.numeric(coef(m))) ## and to plot


## 4.3.1 Additive

tf.XD <- function(x,xk,cmx=NULL,m=2) {
## get X and D subject to constraint
  nk <- length(xk)
  X <- tf.X(x,xk)[,-nk]                   ## basis matrix
  D <- diff(diag(nk),differences=m)[,-nk] ## root penalty
  if (is.null(cmx)) cmx <- colMeans(X)
  X <- sweep(X,2,cmx)        ## subtract cmx from columns
  list(X=X,D=D,cmx=cmx)
} ## tf.XD

am.fit <- function(y,x,v,sp,k=10) {
  ## setup bases and penalties...
  xk <- seq(min(x),max(x),length=k)
  xdx <- tf.XD(x,xk)
  vk <- seq(min(v),max(v),length=k)
  xdv <- tf.XD(v,vk)
  ## create augmented model matrix and response...
  nD <- nrow(xdx$D)*2
  sp <- sqrt(sp)
  X <- cbind(c(rep(1,nrow(xdx$X)),rep(0,nD)),
             rbind(xdx$X,sp[1]*xdx$D,xdv$D*0),
             rbind(xdv$X,xdx$D*0,sp[2]*xdv$D))
  y1 <- c(y,rep(0,nD))
  ## fit model..
  b <- lm(y1~X-1)
  ## compute some useful quantities...
  n <- length(y)
  trA <- sum(influence(b)$hat[1:n]) ## EDF
  rsd <- y-fitted(b)[1:n] ## residuals
  rss <- sum(rsd^2)       ## residual SS
```

```
  sig.hat <- rss/(n-trA)       ## residual variance
  gcv <- sig.hat*n/(n-trA)     ## GCV score
  Vb <- vcov(b)*sig.hat/summary(b)$sigma^2 ## coeff cov matrix
  ## return fitted model...
  list(b=coef(b),Vb=Vb,edf=trA,gcv=gcv,fitted=fitted(b)[1:n],
       rsd=rsd,xk=list(xk,vk),cmx=list(xdx$cmx,xdv$cmx))
} ## am.fit

am.gcv <- function(lsp,y,x,v,k) {
## function suitable for GCV optimization by optim
  am.fit(y,x,v,exp(lsp),k)$gcv
}

## find GCV optimal smoothing parameters...
fit <- optim(c(0,0), am.gcv, y=trees$Volume, x=trees$Girth,
             v=trees$Height,k=10)
sp <- exp(fit$par) ## best fit smoothing parameters
## Get fit at GCV optimal smoothing parameters...
fit <- am.fit(trees$Volume,trees$Girth,trees$Height,sp,k=10)

am.plot <- function(fit,xlab,ylab) {
## produces effect plots for simple 2 term
## additive model
  start <- 2 ## where smooth coeffs start in beta
  for (i in 1:2) {
    ## sequence of values at which to predict...
    x <- seq(min(fit$xk[[i]]),max(fit$xk[[i]]),length=200)
    ## get prediction matrix for this smooth...
    Xp <- tf.XD(x,fit$xk[[i]],fit$cmx[[i]])$X
    ## extract coefficients and cov matrix for this smooth
    stop <- start + ncol(Xp)-1; ind <- start:stop
    b <- fit$b[ind];Vb <- fit$Vb[ind,ind]
    ## values for smooth at x...
    fv <- Xp%*%b
    ## standard errors of smooth at x....
    se <- rowSums((Xp%*%Vb)*Xp)^.5
    ## 2 s.e. limits for smooth...
    ul <- fv + 2*se;ll <- fv - 2 * se
    ## plot smooth and limits...
    plot(x,fv,type="l",ylim=range(c(ul,ll)),xlab=xlab[i],
         ylab=ylab[i])
    lines(x,ul,lty=2);lines(x,ll,lty=2)
    start <- stop + 1
  }
} ## am.plot

par(mfrow=c(1,3))
plot(fit$fitted,trees$Vol,xlab="fitted volume ",
     ylab="observed volume")
am.plot(fit,xlab=c("Girth","Height"),
        ylab=c("s(Girth)","s(Height)"))

## 4.4 Generalized additive
```

```
gam.fit <- function(y,x,v,sp,k=10) {
## gamma error log link 2 term gam fit...
  eta <- log(y) ## get initial eta
  not.converged <- TRUE
  old.gcv <- -100 ## don't converge immediately
  while (not.converged) {
    mu <- exp(eta)  ## current mu estimate
    z <- (y - mu)/mu + eta ## pseudodata
    fit <- am.fit(z,x,v,sp,k) ## penalized least squares
    if (abs(fit$gcv-old.gcv)<1e-5*fit$gcv) {
      not.converged <- FALSE
    }
    old.gcv <- fit$gcv
    eta <- fit$fitted ## updated eta
  }
  fit$fitted <- exp(fit$fitted) ## mu
  fit
} ## gam.fit

gam.gcv <- function(lsp,y,x,v,k=10) {
  gam.fit(y,x,v,exp(lsp),k=k)$gcv
}

fit <- optim(c(0,0),gam.gcv,y=trees$Volume,x=trees$Girth,
             v=trees$Height,k=10)
sp <- exp(fit$par)
fit <- gam.fit(trees$Volume,trees$Girth,trees$Height,sp)
par(mfrow=c(1,3))
plot(fit$fitted,trees$Vol,xlab="fitted volume ",
     ylab="observed volume")
am.plot(fit,xlab=c("Girth","Height"),
        ylab=c("s(Girth)","s(Height)"))

## 4.6 mgcv

library(mgcv)   ## load the package
library(gamair) ## load the data package
data(trees)
ct1 <- gam(Volume~s(Height)+s(Girth),
           family=Gamma(link=log),data=trees)
ct1
plot(ct1,residuals=TRUE)

## 4.6.1
ct2 <- gam(Volume~s(Height,bs="cr")+s(Girth,bs="cr"),
           family=Gamma(link=log),data=trees)
ct2
ct3 <- gam(Volume ~ s(Height) + s(Girth,bs="cr",k=20),
           family=Gamma(link=log),data=trees)
ct3

ct4 <- gam(Volume ~ s(Height) + s(Girth),
```

```
           family=Gamma(link=log),data=trees,gamma=1.4)
ct4
plot(ct4,residuals=TRUE)

## 4.6.2

ct5 <- gam(Volume ~ s(Height,Girth,k=25),
           family=Gamma(link=log),data=trees)
ct5
plot(ct5,too.far=0.15)
ct6 <- gam(Volume ~ te(Height,Girth,k=5),
           family=Gamma(link=log),data=trees)
ct6
plot(ct6,too.far=0.15)

## 4.6.3

gam(Volume~Height+s(Girth),family=Gamma(link=log),data=trees)

trees$Hclass <- factor(floor(trees$Height/10)-5,
                labels=c("small","medium","large"))

ct7 <- gam(Volume ~ Hclass+s(Girth),
           family=Gamma(link=log),data=trees)
par(mfrow=c(1,2))
plot(ct7,all.terms=TRUE)
anova(ct7)
AIC(ct7)
summary(ct7)
```

`ch4.solutions` *Solution code for Chapter 4: Introducing GAMs*

Description

R code for Chapter 4 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch4`

Examples

```
library(gamair); library(mgcv)
## Q.1
set.seed(1)
x <- sort(runif(40)*10)^.5
y <- sort(runif(40))^0.1
## polynomial fits ...
xx <- seq(min(x),max(x),length=200)
plot(x,y)
b<-lm(y~poly(x,5))
lines(xx,predict(b,data.frame(x=xx)))
b<-lm(y~poly(x,10))
lines(xx,predict(b,data.frame(x=xx)),col=2)
## spline fits ...
sb <- function(x,xk) { abs(x-xk)^3}
q<-11
xk<-((1:(q-2)/(q-1))*10)^.5
## lazy person's formula construction ...
form<-paste("sb(x,xk[",1:(q-2),"])",sep="",collapse="+")
form <- paste("y~x+",form)
b<-lm(formula(form))
lines(xx,predict(b,data.frame(x=xx)),col=3)

## Q.2
## x,y, and xx from previous question
b1 <- lm(form)
plot(x,y)
lines(xx,predict(b1,data.frame(x=xx)),col=4)
X <- model.matrix(b1)   # extract model matrix
beta <- solve(t(X)%*%X,t(X)%*%y,tol=0)
b1$coefficients <- beta # trick for simple prediction
lines(xx,predict(b1,data.frame(x=xx)),col=5)
## ... upping the basis dimension to 11 makes the
## normal equations estimates perform very badly.

## Q.8 Additive model as a mixed model

## from 4.2.1 and 4.3.1...
tf <- function(x,xj,j) {
## generate jth tent function from set defined by knots xj
  dj <- xj*0;dj[j] <- 1
  approx(xj,dj,x)$y
}

tf.X <- function(x,xj) {
## tent function basis matrix given data x
## and knot sequence xj
  nk <- length(xj); n <- length(x)
  X <- matrix(NA,n,nk)
  for (j in 1:nk) X[,j] <- tf(x,xj,j)
  X
}
```

```
tf.XD <- function(x,xk,cmx=NULL,m=2) {
## get X and D subject to constraint
  nk <- length(xk)
  X <- tf.X(x,xk)[,-nk]                   ## basis matrix
  D <- diff(diag(nk),differences=m)[,-nk] ## root penalty
  if (is.null(cmx)) cmx <- colMeans(X)
  X <- sweep(X,2,cmx)        ## subtract cmx from columns
  list(X=X,D=D,cmx=cmx)
} ## tf.XD

## Solution code...
## a)

XZmixed <- function(x,xk=NULL,k=10,sep=TRUE) {
## Get re-parameterized model matrix/matrices...
  if (is.null(xk)) xk <- seq(min(x),max(x),length=k)
  xd <- tf.XD(x,xk)
  D <- rbind(0,xd$D); D[1,1] <- 1
  X <- t(solve(t(D),t(xd$X)))
  if (sep) list(X=X[,1,drop=FALSE],Z=X[,-1],xk=xk)
  else list(X=X,xk=xk)
} ## XZmixed

## b)
## get components of smooths for Height and Girth...
xh <- XZmixed(trees$Height)
xg <- XZmixed(trees$Girth)

## Fit as mixed model...
X <- cbind(1,xh$X,xg$X)
Zg <- xg$Z; Zh <- xh$Z
g1 <- g <- factor(rep(1,nrow(X)))
vol <- trees$Volume
b <- lme(vol~X-1,random=list(g=pdIdent(~Zh-1),
         g1=pdIdent(~Zg-1)))

## c)
## raw vs. fitted and residual plot
par(mfrow=c(1,2))
plot(fitted(b),vol)
rsd <- vol - fitted(b)
plot(fitted(b),rsd)

## extract coefs for each smooth...
bh <- as.numeric(coef(b)[c(2,4:11)]) ## coefs for s(Height)
bg <- as.numeric(coef(b)[c(3,12:19)]) ## coefs for s(Height)

## get smooth specific prediction matrices...
Xh <- XZmixed(trees$Height,xk=xh$xk,sep=FALSE)$X
Xg <- XZmixed(trees$Girth,xk=xg$xk,sep=FALSE)$X

## d)
```

```
## plot smooths over partial residuals...
sh <- Xh%*%bh
sg <- Xg%*%bg

par(mfrow=c(1,2))
plot(trees$Girth,sg+rsd,pch=19,col="grey",
     xlab="Girth",ylab="s(Girth)")
lines(trees$Girth,sg)
plot(trees$Height,sh+rsd,pch=19,col="grey",
     xlab="Height",ylab="s(Height)")
lines(trees$Height,sh)

## Q.9 Generalized version of 8 by PQL
## a)

gamm.fit <- function(y,X,Zh,Zg) {
## gamma error log link 2 term gam fit via PQL...
  eta <- log(y) ## get initial eta
  g <- g1 <- factor(rep(1,nrow(X)))
  not.converged <- TRUE
  old.reml <- 1e100 ## don't converge immediately
  while (not.converged) {
    mu <- exp(eta)  ## current mu estimate
    z <- (y - mu)/mu + eta ## pseudodata
    fit <- lme(z~X-1,random=list(g=pdIdent(~Zh-1),g1=pdIdent(~Zg-1)))
    if (abs(logLik(fit)-old.reml)<1e-5*abs(logLik(fit))) {
      not.converged <- FALSE
    }
    old.reml <- logLik(fit)
    eta <- fitted(fit) ## updated eta
  }
  fit
} ## gamm.fit

## b) re-using arguments from Q.8...
m <- gamm.fit(vol,X,Zh,Zg)

## c)
rsd <- residuals(m)
par(mfrow=c(1,2))
plot(exp(fitted(m)),vol);abline(0,1)
plot(fitted(m),rsd)

## d)
bh <- as.numeric(coef(m)[c(2,4:11)]) ## coefs for s(Height)
bg <- as.numeric(coef(m)[c(3,12:19)]) ## coefs for s(Height)

sh <- Xh%*%bh
sg <- Xg%*%bg

par(mfrow=c(1,2))
plot(trees$Girth,sg+rsd,pch=19,col="grey",
     xlab="Girth",ylab="s(Girth)")
```

```
lines(trees$Girth,sg)
plot(trees$Height,sh+rsd,pch=19,col="grey",
     xlab="Height",ylab="s(Height)")
lines(trees$Height,sh)
```

---

ch5 *Code for Chapter 5: Smoothers*

Description

R code from Chapter 5 of the second edition of ‘Generalized Additive Models: An Introduction with R’ is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch5.solutions`

Examples

```
library(gamair); library(mgcv)

## 5.3.3 P-splines

bspline <- function(x,k,i,m=2)
# evaluate ith b-spline basis function of order m at the values
# in x, given knot locations in k
{ if (m==-1) # base of recursion
  { res <- as.numeric(x<k[i+1]&x>=k[i])
  } else     # construct from call to lower order basis
  { z0 <- (x-k[i])/(k[i+m+1]-k[i])
    z1 <- (k[i+m+2]-x)/(k[i+m+2]-k[i+1])
    res <- z0*bspline(x,k,i,m-1)+ z1*bspline(x,k,i+1,m-1)
  }
  res
} ## bspline

k<-6                              # example basis dimension
P <- diff(diag(k),differences=1)  # sqrt of penalty matrix
```

```
S <- t(P)%*%P

## 5.3.6 SCOP-splines
x <- 0:200/200
set.seed(32)
y <- binomial()$linkinv((x-.5)*10) + rnorm(length(x))*.1
plot(x,y)
k <- 7
ssp <- s(x,bs="ps",k=k); ssp$mono <- 1
sm <- smoothCon(ssp,data.frame(x))[[1]]
X <- sm$X; XX <- crossprod(X); sp <- .005
gamma <- rep(0,k); S <- sm$S[[1]]
for (i in 1:20) {
  gt <- c(gamma[1],exp(gamma[2:k]))
  dg <- c(1,gt[2:k])
  g <- -dg*(t(X)%*%(y-X%*%gt)) + sp*S%*%gamma
  H <- dg*t(dg*XX)
  gamma <- gamma - solve(H+sp*S,g)
}
lines(x,X%*%gt)
```

`ch5.solutions` *Solution code for Chapter 5: Smoothers*

Description

R code for Chapter 5 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch5`

Examples

```
library(gamair); library(mgcv)
## Q.4 P-spline
## a)
library(splines)
pspline.XB <- function(x,q=10,m=2,p.m=2)
```

```
# Get model matrix and sqrt Penalty matrix for P-spline
{ # first make knot sequence, k
  k <- seq(min(x),max(x),length=q-m)
  dk <- k[2]-k[1]
  k <- c(k[1]-dk*((m+1):1),k,k[q-m]+dk*(1:(m+1)))
  # now get model matrix and root penalty
  X <- splineDesign(k,x,ord=m+2)
  B <- diff(diag(q),difference=p.m)
  list(X=X,B=B)
} ## pspline.XB

## b)
n<-100
x <- sort(runif(n))
ps <- pspline.XB(x,q=9,m=2,p.m=2)
par(mfrow=c(3,3)) # plot the original basis functions
for (i in 1:9) plot(x,ps$X[,i],type="l")

## c)
S <-  t(ps$B)%*%ps$B
es <- eigen(S);U <- es$vectors
XU <- ps$X%*%U # last p.m cols are penalty null space
par(mfrow=c(3,3)) # plot penalty eigenbasis functions
for (i in 1:9) plot(x,XU[,i],type="l")

## d)
qrx <- qr(ps$X) # QR of X
R <- qr.R(qrx)
RSR <- solve(t(R),S);RSR <- t(solve(t(R),t(RSR)))
ersr <- eigen(RSR)
U <- ersr$vectors
Q <- qr.Q(qrx)
QU <- Q%*%U
par(mfrow=c(3,3)) # plot the natural basis functions
for (i in 1:9) plot(x,QU[,i],type="l")

## Q.5

test1<-function(x,z,sx=0.3,sz=0.4)
{ 1.2*exp(-(x-0.2)^2/sx^2-(z-0.3)^2/sz^2)+
  0.8*exp(-(x-0.7)^2/sx^2-(z-0.8)^2/sz^2)
}
n <- 200
x <- matrix(runif(2*n),n,2)
f <- test1(x[,1],x[,2])
y <- f + rnorm(n)*.1

eta <- function(r)
{ # thin plate spline basis functions
  ind <- r<=0
  eta <- r
  eta[!ind] <- r[!ind]^2*log(r[!ind])/(8*pi)
  eta[ind] <- 0
```

```
  eta
} ## eta

XSC <- function(x,xk=x)
{ # set up t.p.s., given covariates, x, and knots, xk
  n <- nrow(x);k <- nrow(xk)
  X <- matrix(1,n,k+3)  # tps model matrix
  for (j in 1:k) {
    r <- sqrt((x[,1]-xk[j,1])^2+(x[,2]-xk[j,2])^2)
    X[,j] <- eta(r)
  }
  X[,j+2] <- x[,1];X[,j+3] <- x[,2]
  C <- matrix(0,3,k+3)  # tps constraint matrix
  S <- matrix(0,k+3,k+3)# tps penalty matrix
  for (i in 1:k) {
   C[1,i]<-1;C[2,i] <- xk[i,1];C[3,i] <- xk[i,2]
   for (j in i:k) S[j,i]<-S[i,j] <-
               eta(sqrt(sum((xk[i,]-xk[j,])^2)))
  }
  list(X=X,S=S,C=C)
} ## XSC

absorb.con <- function(X,S,C)
{ # get constraint null space, Z...
  qrc <- qr(t(C)) # QR=C', Q=[Y,Z]
  m <- nrow(C);k <- ncol(X)
  X <- t(qr.qty(qrc,t(X)))[,(m+1):k] # form XZ
  # now form Z'SZ ...
  S <- qr.qty(qrc,t(qr.qty(qrc,t(S))))[(m+1):k,(m+1):k]
  list(X=X,S=S,qrc=qrc)
} ## absorb.con

fit.tps <- function(y,x,xk=x,lambda=0)
{ tp <- XSC(x,xk)                  # get tps matrices
  tp <- absorb.con(tp$X,tp$S,tp$C) # make unconstrained
  ev <- eigen(tp$S,symmetric=TRUE) # get sqrt penalty, rS
  rS <- ev$vectors%*%(ev$values^.5*t(ev$vectors))
  X <- rbind(tp$X,rS*sqrt(lambda)) # augmented model matrix
  z <- c(y,rep(0,ncol(rS)))        # augmented data
  beta <- coef(lm(z~X-1))          # fit model
  beta <- qr.qy(tp$qrc,c(0,0,0,beta)) # backtransform beta
} ## fit.tps

eval.tps <- function(x,beta,xk)
{ # evaluate tps at x, given parameters, beta, and knots, xk.
  k <- nrow(xk);n <- nrow(x)
  f <- rep(beta[k+1],n)
  for (i in 1:k) {
    r <- sqrt((x[,1]-xk[i,1])^2+(x[,2]-xk[i,2])^2)
    f <- f + beta[i]*eta(r)
  }
  f <- f + beta[k+2]*x[,1] + beta[k+3]*x[,2]
} ## eval.tps
```

```
## select some `knots', xk ...
ind <- sample(1:n,100,replace=FALSE)
xk <- x[ind,]
## fit model ...
beta <- fit.tps(y,x,xk=xk,lambda=.01)

## contour truth and fit
par(mfrow=c(1,2))
xp <- matrix(0,900,2)
x1<-seq(0,1,length=30);x2<-seq(0,1,length=30)
xp[,1]<-rep(x1,30);xp[,2]<-rep(x2,rep(30,30))
truth<-matrix(test1(xp[,1],xp[,2]),30,30)
contour(x1,x2,truth)
fit <- matrix(eval.tps(xp,beta,xk),30,30)
contour(x1,x2,fit)

## Q.6 smooth.construct

tf <- function(x,xj,j) {
## generate jth tent function from set defined by knots xj
  dj <- xj*0;dj[j] <- 1
  approx(xj,dj,x)$y
}

tf.X <- function(x,xj) {
## tent function basis matrix given data x
## and knot sequence xj
  nk <- length(xj); n <- length(x)
  X <- matrix(NA,n,nk)
  for (j in 1:nk) X[,j] <- tf(x,xj,j)
  X
}

smooth.construct.pl.smooth.spec<-function(object,data,knots) {
## a piecewise linear smooth constructor method function
  m <- object$p.order[1]
  if (is.na(m)) m <- 2 ## default
  if (m<1) stop("silly m supplied")
  if (object$bs.dim<0) object$bs.dim <- 20 ## default
  x <- data[[object$term]]  ## the data
  k <- knots[[object$term]] ## will be NULL if none supplied
  if (is.null(k)) { # space knots through data
    k <- seq(min(x),max(x),length=object$bs.dim)
  } else {
    if (length(k)!=object$bs.dim) # right number of knots?
    k <- seq(min(k),max(k),length=object$bs.dim)
  }
  object$X <- tf.X(x,k)
  if (!object$fixed) { # create the penalty matrix
    object$S[[1]] <- crossprod(diff(diag(object$bs.dim),difference=m))
  }
  object$rank <- object$bs.dim - m  # penalty rank
```

```
  object$null.space.dim <- m  # dim. of unpenalized space
  ## store "tr" specific stuff ...
  object$knots <- k

  object$df <- ncol(object$X)     # maximum DoF (if unconstrained)

  class(object) <- "pl.smooth"  # Give object a class
  object
}

Predict.matrix.pl.smooth<-function(object,data)
## prediction method function for the `pl' smooth class
{ x <- data[[object$term]]
  X <- tf.X(x,object$knots)
  X # return the prediction matrix
}

# an example, using the new class....
require(mgcv)
set.seed(10)
dat <- gamSim(1,n=400,scale=2)
b <- gam(y~s(x0,bs="pl",m=2)+s(x1,bs="pl",m=2) +
         s(x2,bs="pl",m=3)+s(x3,bs="pl",m=2),
 data=dat,method="REML")
plot(b,pages=1)
```

`ch6` *Code for Chapter 6: GAM Theory*

Description

R code from Chapter 6 of the second edition of ‘Generalized Additive Models: An Introduction with R’ is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch6.solutions`

Examples

```
library(gamair); library(mgcv)

## 6.13.2 backfitting

set.seed(2) ## simulate some data...
dat <- gamSim(1,n=400,dist="normal",scale=2)
edf <- c(3,3,8,3)
y <- dat$y
x <- cbind(dat$x0,dat$x1,dat$x2,dat$x3)
f <- x*0; alpha <- mean(y); ok <- TRUE; rss0 <- 0
while (ok) { # backfitting loop
  for (i in 1:ncol(x)) { # loop through the smooth terms
    ep <- y - rowSums(f[,-i]) - alpha
    b <- smooth.spline(x[,i],ep,df=edf[i])
    f[,i] <- predict(b,x[,i])$y
  }
  rss <- sum((y-rowSums(f))^2)
  if (abs(rss-rss0)<1e-6*rss) ok <- FALSE
  rss0 <- rss
}
par(mfrow=c(2,2))
for (i in 1:ncol(x)) {
  plot(x[,i],y-mean(y),col="grey",pch=19,cex=.3)
  ii <- order(x[,i])
  lines(x[ii,i],f[ii,i],col=2,lwd=2)
}
```

`ch6.solutions` *Solution code for Chapter 6: GAM Theory*

Description

R code for Chapter 6 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch6`

Examples

```
library(gamair); library(mgcv)

## code from Chapter 5 solutions...

## Q.3

pspline.XB <- function(x,q=10,m=2,p.m=2)
# Get model matrix and sqrt Penalty matrix for P-spline
{ # first make knot sequence, k
  k <- seq(min(x),max(x),length=q-m)
  dk <- k[2]-k[1]
  k <- c(k[1]-dk*((m+1):1),k,k[q-m]+dk*(1:(m+1)))
  # now get model matrix and root penalty
  X <- splineDesign(k,x,ord=m+2)
  B <- diff(diag(q),difference=p.m)
  list(X=X,B=B)
} ## pspline.XB

## a) and b)
fit.wPs <- function(y,X,B,lambda=0,w=rep(1,length(y)))
# fit to y by weighted penalized least squares, X is
# model matrix, B is sqrt penalty, lambda is smoothing p.
{ w <- as.numeric(w^.5)
  n <- nrow(X)
  X<-rbind(w*X,sqrt(lambda)*B)
  y<-c(w*y,rep(0,nrow(B)))
  b <- lm(y~X-1) # actually estimate model
  trA <- sum(influence(b)$hat[1:n])
  rss <- sum((y-fitted(b))[1:n]^2) ## not really needed here
  list(trA=trA,rss=rss,b=coef(b))
}

fitPoiPs <- function(y,X,B,lambda=0)
# Fit Poisson model with log-link by P-IRLS
{ mu <- y;mu[mu==0] <- .1
  eta <- log(mu)
  converged <- FALSE
  dev <- ll.sat <- sum(dpois(y,y,log=TRUE))
  while (!converged) {
    z <- (y-mu)/mu + eta
    w <- mu
    fPs <- fit.wPs(z,X,B,lambda,w)
    eta <- X%*%fPs$b
    mu=exp(eta)
    old.dev <- dev
    dev <- 2*(ll.sat-sum(dpois(y,mu,log=TRUE)))
    if (abs(dev-old.dev)<1e-6*dev) converged <- TRUE
  }
  list(dev=dev,rss=fPs$rss,trA=fPs$trA,b=fPs$b,fv=mu)
}
```

```
## c)
## simulate data as in question...
set.seed(1)
f <- function(x) .04*x^11*(10*(1-x))^6+2*(10*x)^3*(1-x)^10
n <- 100;x <- sort(runif(n))
y <- rpois(rep(1,n),exp(f(x)))

## fitting...
library(splines)
ps <- pspline.XB(x,q=10,m=2,p.m=2)
lambda <- 1e-4;reps <- 60
sp <- trA <- gcv <- rep(0,reps)
for (i in 1:reps) { # loop through trial s.p.s
  fps <- fitPoiPs(y,ps$X,ps$B,lambda=lambda)
  trA[i] <- fps$trA;sp[i] <- lambda
  gcv[i] <- n*fps$dev/(n-trA[i])^2
  lambda <- lambda*1.3
}
plot(trA,gcv,type="l")
fps1 <- fitPoiPs(y,ps$X,ps$B,lambda=sp[gcv==min(gcv)])
plot(x,y);lines(x,fps1$fv)

## Q.6 Fellner-Schall for GCV and AIC...

## b)
library(mgcv);library(MASS)
sm <- smoothCon(s(times,k=20),data=mcycle)[[1]]
X <- sm$X; S <- sm$S[[1]]; y <- mcycle$accel
lambda <- 1; n <- length(y)
XX <- crossprod(X);
with(mcycle,plot(times,accel))
for (i in 1:20) {
  R <- chol(XX+lambda*S)
  b <- backsolve(R,forwardsolve(t(R),t(X) %*% y))
  f <- X %*% b
  lines(mcycle$times,f,col="grey")
  HiS <- backsolve(R,forwardsolve(t(R),S))
  HiH <- backsolve(R,forwardsolve(t(R),XX))
  tau <- sum(diag(HiH))
  if (i>1) { ## convergence test
    if (abs(tau-tau0)<1e-5*tau) break
  }
  tau0 <- tau
  dt.dl <- -sum(t(HiH)*HiS)
  db.dl <- -HiS %*% b
  dD.db <- 2*t(X) %*% (f - y)
  lambda <- -sum(2*(y-f)^2)/(n-tau)*dt.dl/sum(db.dl*dD.db) * lambda
}
lines(mcycle$times,f)

## c)
y <- c(12,14,33,50,67,74,123,141,165,204,253,246,240)
t <- 1:13
```

```
sm <- smoothCon(s(t),data=data.frame(t=t,y=y))[[1]]
X <- sm$X; S <- sm$S[[1]]; lambda <- .001; n <- length(y)
plot(t,y)
mu <- y; eta <- log(mu)
for (i in 1:50) {
  w <- mu; z <- (y-mu)/mu + eta
  XWX <- crossprod(sqrt(w)*X)
  R <- chol(XWX+lambda*S)
  b <- backsolve(R,forwardsolve(t(R),t(X) %*% (w*z)))
  eta <- drop(X %*% b);mu <- exp(eta)
  lines(t,mu,col="grey")
  HiS <- backsolve(R,forwardsolve(t(R),S))
  HiH <- backsolve(R,forwardsolve(t(R),XWX))
  tau <- sum(diag(HiH))
  if (i>1) { ## convergence test
    if (abs(tau-tau0)<1e-5*tau) break
  }
  tau0 <- tau
  dt.dl <- -sum(t(HiH)*HiS)
  db.dl <- -HiS %*% b
  dl.db <- t(X) %*% (y-mu) ## especially simple for this case
  lambda <- dt.dl/sum(db.dl*dl.db) * lambda
}
i;tau;lines(t,mu)

## Q.8 log det stabilty (or lack of)

set.seed(1);lam <- 1
A1 <- crossprod(diff(diag(3),diff=1))
A2 <- crossprod(matrix(runif(9),3,3))
A <- matrix(0,5,5);A[1:3,1:3] <- A1
A[3:5,3:5] <- A[3:5,3:5] + lam * A2

ldetA.qr <- ldetA.ev <- ldetA.svd <- ldetA <-
            rho <- seq(-40,-25,length=100)
for (i in 1:length(rho)) {
  lam <- exp(rho[i])
  A <- matrix(0,5,5);A[1:3,1:3] <- A1
  A[3:5,3:5] <- A[3:5,3:5] + lam * A2
  ea1 <- eigen(A1)
  Q <- diag(5);Q[1:3,1:3] <- ea1$vectors
  At <- matrix(0,5,5)
  At[3:5,3:5] <- At[3:5,3:5] + lam * A2
  At <- t(Q)%*%At%*%Q
  diag(At)[1:2] <- diag(At)[1:2]+ea1$values[1:2]

  ldetA[i] <- sum(log(abs(diag(qr.R(qr(At))))))
  ldetA.qr[i] <- sum(log(abs(diag(qr.R(qr(A))))))
  ldetA.ev[i] <- sum(log(abs(eigen(A)$values)))
  ldetA.svd[i] <- sum(log(abs(svd(A)$d)))
}
plot(rho,ldetA,type="l") ## nice and stable
## not...
```

```
lines(rho,ldetA.qr,lty=2)
lines(rho,ldetA.ev,lty=3)
lines(rho,ldetA.svd,lty=4)
```

---

ch7 *Code for Chapter 7: GAMs in Practice: mgcv*

Description

R code from Chapter 7 of the second edition of ‘Generalized Additive Models: An Introduction with R’ is in the examples section below.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch7.solutions`

Examples

```
library(gamair); library(mgcv)
## NOTE: Examples are marked 'Not run' to save CRAN check time

## 7.1.1 using smooth constructors

library(mgcv); library(MASS) ## load for mcycle data.
## set up a smoother...
sm <- smoothCon(s(times,k=10),data=mcycle,knots=NULL)[[1]]
## use it to fit a regression spline model...
beta <- coef(lm(mcycle$accel~sm$X-1))
with(mcycle,plot(times,accel)) ## plot data
times <- seq(0,60,length=200)  ## create prediction times
## Get matrix mapping beta to spline prediction at 'times'
Xp <- PredictMat(sm,data.frame(times=times))
lines(times,Xp%*%beta) ## add smooth to plot

## Not run:
## 7.2 Brain scan
## 7.2.1 preliminary modelling
```

```
require(gamair); require(mgcv); data(brain)
brain <- brain[brain$medFPQ>5e-3,] # exclude 2 outliers
m0 <- gam(medFPQ~s(Y,X,k=100),data=brain)
gam.check(m0)

e <- residuals(m0); fv <- fitted(m0)
lm(log(e^2)~log(fv))

m1<-gam(medFPQ^.25~s(Y,X,k=100),data=brain)
gam.check(m1)
m2<-gam(medFPQ~s(Y,X,k=100),data=brain,family=Gamma(link=log))
mean(fitted(m1)^4);mean(fitted(m2));mean(brain$medFPQ)

m2
vis.gam(m2,plot.type="contour",too.far=0.03,
        color="gray",n.grid=60,zlim=c(-1,2))

## 7.2.2 additive?
m3 <- gam(medFPQ~s(Y,k=30)+s(X,k=30),data=brain,
          family=Gamma(link=log))
m3
AIC(m2,m3)

## 7.2.3 isotropic or tensor
tm <- gam(medFPQ~te(Y,X,k=10),data=brain,family=Gamma(link=log))
tm1 <- gam(medFPQ ~ s(Y,k=10,bs="cr") + s(X,bs="cr",k=10) +
           ti(X,Y,k=10), data=brain, family=Gamma(link=log))
AIC(m2,tm,tm1)
anova(tm1)

## 7.2.4 Detecting symmetry
brain$Xc <- abs(brain$X - 64.5)
brain$right <- as.numeric(brain$X<64.5)
m.sy <- gam(medFPQ~s(Y,Xc,k=100),data=brain,
            family=Gamma(link=log))
m.as <- gam(medFPQ~s(Y,Xc,k=100)+s(Y,Xc,k=100,by=right),
            data=brain,family=Gamma(link=log))
m.sy
m.as

anova(m.as)

vis.gam(m.sy,plot.type="contour",view=c("Xc","Y"),too.far=.03,
        color="gray",n.grid=60,zlim=c(-1,2),main="both sides")
vis.gam(m.as,plot.type="contour",view=c("Xc","Y"),
        cond=list(right=0),too.far=.03,color="gray",n.grid=60,
        zlim=c(-1,2),main="left side")
vis.gam(m.as,plot.type="contour",view=c("Xc","Y"),
        cond=list(right=1),too.far=.03,color="gray",n.grid=60,
        zlim=c(-1,2),main="right side")

## 7.2.5 Comparing surfaces
brain1 <- brain
```

```
mu <- fitted(m2)
n<-length(mu)
ind <- brain1$X<60 & brain1$Y<20
mu[ind] <- mu[ind]/3
set.seed(1)
brain1$medFPQ <- rgamma(rep(1,n),mu/m2$sig2,scale=m2$sig2)

brain2=rbind(brain,brain1)
brain2$sample1 <- c(rep(1,n),rep(0,n))
brain2$sample0 <- 1 - brain2$sample1

m.same<-gam(medFPQ~s(Y,X,k=100),data=brain2,
            family=Gamma(link=log))
m.diff<-gam(medFPQ~s(Y,X,k=100)+s(Y,X,by=sample1,k=100),
            data=brain2,family=Gamma(link=log))
AIC(m.same,m.diff)
anova(m.diff)

## 7.2.6 Prediction
predict(m2)[1:5]
pv <- predict(m2,se=TRUE)
pv$fit[1:5]
pv$se[1:5]

predict(m2,type="response")[1:5]
pv <- predict(m2,type="response",se=TRUE)
pv$se[1:5]

pd <- data.frame(X=c(80.1,68.3),Y=c(41.8,41.8))
predict(m2,newdata=pd)
predict(m2,newdata=pd,type="response",se=TRUE)

predict(m3,newdata=pd,type="terms",se=TRUE)

Xp <- predict(m2,newdata=pd,type="lpmatrix")
fv <- Xp%*%coef(m2)
fv
d <- t(c(1,-1))
d%*%fv
d%*%Xp%*%m2$Vp%*%t(Xp)%*%t(d)

## 7.2.7 Variance of non-linear function

ind <- brain$region==1& ! is.na(brain$region)
Xp <- predict(m2,newdata=brain[ind,],type="lpmatrix")
set.seed(8) ## for repeatability
br <- rmvn(n=1000,coef(m2),vcov(m2)) # simulate from posterior
mean.FPQ<-rep(0,1000)
for (i in 1:1000)
{ lp <- Xp%*%br[i,]  # replicate linear predictor
  mean.FPQ[i] <- mean(exp(lp)) # replicate region 1 mean FPQ
}
mean.FPQ <- colMeans(exp(Xp%*%t(br)))
```

```
## 7.3 Retinopathy
require(gamair); require(mgcv); data(wesdr)
k <- 7
b <- gam(ret ~ s(dur,k=k) + s(gly,k=k) + s(bmi,k=k) +
         ti(dur,gly,k=k) + ti(dur,bmi,k=k) + ti(gly,bmi,k=k),
         select=TRUE, data=wesdr, family=binomial(), method="ML")
b

## 7.4 Air pollution
data(chicago)
ap0 <- gam(death~s(time,bs="cr",k=200)+pm10median+so2median+
           o3median+tmpd,data=chicago,family=poisson)
gam.check(ap0)

par(mfrow=c(2,1))
plot(ap0,n=1000)  # n increased to make plot smooth
plot(ap0,residuals=TRUE,n=1000)

chicago$death[3111:3125]

ap1<-gam(death~s(time,bs="cr",k=200)+s(pm10median,bs="cr")+
     s(so2median,bs="cr")+s(o3median,bs="cr")+s(tmpd,bs="cr"),
     data=chicago,family=poisson)

## 7.4.1 single index

lagard <- function(x,n.lag=6) {
  n <- length(x); X <- matrix(NA,n,n.lag)
  for (i in 1:n.lag) X[i:n,i] <- x[i:n-i+1]
  X
}
dat <- list(lag=matrix(0:5,nrow(chicago),6,byrow=TRUE),
            death=chicago$death,time=chicago$time)
dat$pm10 <- lagard(chicago$pm10median)
dat$tmp <- lagard(chicago$tmpd)
dat$o3 <- lagard(chicago$o3median)

si <- function(theta,dat,opt=TRUE) {
## Return ML if opt==TRUE or fitted gam otherwise.
  alpha <- c(1,theta) ## alpha defined via unconstrained theta
  kk <- sqrt(sum(alpha^2)); alpha <- alpha/kk  ## ||alpha||=1
  o3 <- dat$o3%*%alpha; tmp <- dat$tmp%*%alpha
  pm10 <- dat$pm10%*%alpha ## re-weight lagged covariates
  b<- bam(dat$death~s(dat$time,k=200,bs="cr")+s(pm10,bs="cr")+
          te(o3,tmp,k=8),family=poisson) ## fit model
  cat(".") ## give user something to watch
  if (opt) return(b$gcv.ubre) else {
    b$alpha <- alpha  ## add alpha to model object
    b$J <- outer(alpha,-theta/kk^2) ## get dalpha_i/dtheta_j
    for (j in 1:length(theta)) b$J[j+1,j] <- b$J[j+1,j] + 1/kk
    return(b)
  }
```

```
} ## si

## WARNING: the next line takes around half an hour to run

f1 <- optim(rep(1,5),si,method="BFGS",hessian=TRUE,dat=dat)

apsi <- si(f1$par,dat,opt=FALSE)
apsi$alpha

## 7.4.2 distributed lag...

apl <- bam(death~s(time,bs="cr",k=200)+te(pm10,lag,k=c(10,5))+
       te(o3,tmp,lag,k=c(8,8,5)),family=poisson,data=dat)

## 7.5 Egg survey - less than a minute
## 7.5.1 Model development
data(mack)
mack$log.net.area <- log(mack$net.area)

gmtw <- gam(egg.count ~ s(lon,lat,k=100) + s(I(b.depth^.5))+
       s(c.dist) + s(salinity) + s(temp.surf) + s(temp.20m)+
       offset(log.net.area),data=mack,family=tw,method="REML",
       select=TRUE)
gm2 <- gam(egg.count ~ s(lon,lat,k=100) + s(I(b.depth^.5)) +
         s(c.dist) + s(temp.20m) + offset(log.net.area),
         data=mack,family=tw,method="REML")
gm2

## 7.5.2 model predictions
par(mfrow=c(1,3))
data(mackp); data(coast)
mackp$log.net.area <- rep(0,nrow(mackp))
lon <- seq(-15,-1,1/4); lat <- seq(44,58,1/4)
zz<-array(NA,57*57); zz[mackp$area.index]<-predict(gm2,mackp)
image(lon,lat,matrix(zz,57,57),col=gray(0:32/32),
      cex.lab=1.5,cex.axis=1.4)
contour(lon,lat,matrix(zz,57,57),add=TRUE)
lines(coast$lon,coast$lat,col=1)

set.seed(4) ## make reproducable
br1 <- rmvn(n=1000,coef(gm2),vcov(gm2))
Xp <- predict(gm2,newdata=mackp,type="lpmatrix")
mean.eggs1 <- colMeans(exp(Xp%*%t(br1)))
hist(mean.eggs1)

## 7.5.3 alternative

gmgr <- gam(egg.count ~s(lon,lat,k=100)+s(lon,lat,by=temp.20m)
        +s(lon,lat,by=I(b.depth^.5)) +offset(log.net.area),
        data=mack,family=tw,method="REML")

## 7.6 Larks - about a minute
library(gamair); data(bird)
```

```
bird$n <- bird$y/1000;bird$e <- bird$x/1000
m1 <- gam(crestlark~s(e,n,k=100),data=bird,family=binomial,
          method="REML")
m1

m2 <- gam(crestlark ~ s(e,n,bs="ds",m=c(1,.5),k=100),data=bird,family=binomial,
          method="REML")

REML <- r <- 1:10*10
for (i in 1:length(r)) {
  mt <- gam(crestlark ~ s(e,n,bs="gp",m=c(3,r[i]),k=100),
        data=bird,family=binomial,method="REML")
  REML[i] <- mt$gcv.ubre
  if (i==1||REML[i]==REML0) { m3 <- mt; REML0 <- REML[i]}
}
AIC(m1,m2,m3)

bird$tet.n <- bird$N <- rep(1,nrow(bird))
bird$N[is.na(as.vector(bird$crestlark))] <- NA
ba <- aggregate(data.matrix(bird), by=list(bird$QUADRICULA),
                FUN=sum, na.rm=TRUE)
ba$e <- ba$e/ba$tet.n; ba$n <- ba$n/ba$tet.n

m10 <- gam(cbind(crestlark,N-crestlark) ~ s(e,n,k=100),
           data=ba, family=binomial, method="REML")
library(geoR)
coords<-matrix(0,nrow(ba),2);coords[,1]<-ba$e;coords[,2]<-ba$n
gb<-list(data=residuals(m10,type="d"),coords=coords)
plot(variog(gb,max.dist=100))
plot(fitted(m10),residuals(m10))

## 7.7.1 Sole egg GAMM
## Chapter 3 preliminaries...
data(sole)
sole$off <- log(sole$a.1-sole$a.0)# model offset term
sole$a<-(sole$a.1+sole$a.0)/2     # mean stage age
solr<-sole                        # make copy for rescaling
solr$t<-solr$t-mean(sole$t)
solr$t<-solr$t/var(sole$t)^0.5
solr$la<-solr$la-mean(sole$la)
solr$lo<-solr$lo-mean(sole$lo)

## GAMM fit...
solr$station <- factor(with(solr,paste(-la,-lo,-t,sep="")))
som <- gamm(eggs~te(lo,la,t,bs=c("tp","tp"),k=c(25,5),d=c(2,1))
            +s(t,k=5,by=a)+offset(off), family=quasipoisson,
            data=solr,random=list(station=~1))
som$gam
som1 <- bam(eggs~te(lo,la,t,bs=c("tp","tp"),k=c(25,5),d=c(2,1))
            + s(t,k=5,by=a)+offset(off)+s(station,bs="re"),
        family=quasipoisson,data=solr)
gam.vcomp(som1)
som$lme
```

```
## boundary and knots for soap...
bnd <- list(list(lo=c(-6.74,-5.72,-5.7 ,-5.52,-5.37,-5.21,-5.09,-5.02,
         -4.92,-4.76,-4.64,-4.56,-4.53,-4.3,-4.16,-3.8 ,-3.8,-5.04,-6.76,
  -6.74),
         la=c(50.01,50.02,50.13,50.21,50.24,50.32,50.41,50.54,50.59,50.64,
  50.74,50.86,51.01,51  ,51.2,51.22,51.61,51.7,51.7,50.01)))

knt <- list(lo=c(-4.643,-5.172,-5.638,-6.159,-6.665,-6.158,-5.656,-5.149,
  -4.652,-4.154,-3.901,-4.146,-4.381,-4.9,-5.149,-5.37,-5.866,-6.36,-6.635,
  -6.12,-5.626,-5.117,-4.622,-4.695,-4.875,-5.102,-5.609,-5.652,-5.141,
  -5.354,-5.843,-6.35,-6.628,-6.127,-5.63,-5.154,-5.356,-5.652,-5.853,
  -6.123),
   la=c(51.626,51.61,51.639,51.638,51.376,51.377,51.373,51.374,51.374,
   51.376,51.379,51.226,51.129,51.194,51.083,51.147,51.129,51.151,50.901,
   50.891,50.959,50.958,50.942,50.728,50.676,50.818,50.825,50.684,50.693,
   50.568,50.564,50.626,50.397,50.451,50.443,50.457,50.325,50.193,50.322,
   50.177))

sole$station <- solr$station ## station to sole data

som2 <- bam(eggs ~ te(lo,la,t,bs=c(”sw”,”cr”),k=c(40,5),
            d=c(2,1),xt=list(list(bnd=bnd),NULL)) +
            s(t,k=5,by=a) + offset(off) + s(station,bs=”re”),
            knots=knt, family=quasipoisson, data=sole)

## 7.7.2 Cairo temperature
data(cairo)
ctamm <- gamm(temp~s(day.of.year,bs=”cc”,k=20)+s(time,bs=”cr”),
         data=cairo,correlation=corAR1(form=~1|year))
summary(ctamm$gam)
intervals(ctamm$lme,which=”var-cov”)
ctamm$gam$sig2/ctamm$gam$sp
plot(ctamm$gam,scale=0,pages=1)

REML <- rho <- 0.6+0:20/100
for (i in 1:length(rho)) {
  ctbam <- bam(temp~s(day.of.year,bs=”cc”,k=20)+s(time,bs=”cr”),
             data=cairo,rho=rho[i])
  REML[i] <- ctbam$gcv.ubre
}
rho[REML==min(REML)]

## 7.7.3 Fully Bayesian
## Not currently included (requires editing of JAGS file)

## 7.7.4 Random wiggly curves
data(sitka)
sitka$id.num <- as.factor(sitka$id.num)
b <- gamm(log.size~s(days) + ozone + ozone:days +
         s(days,id.num,bs=”fs”,k=5),data=sitka)
plot(b$gam,pages=1)
```

```
## 7.8 survival
require(survival)
data(pbc) ## loads pbcseq also
pbc$status1 <- as.numeric(pbc$status==2)
pbc$stage <- factor(pbc$stage)
b0 <- gam(time ~ trt+sex+stage+s(sqrt(protime))+s(platelet)+
          s(age)+s(bili)+s(albumin)+s(sqrt(ast))+s(alk.phos),
          weights=status1,family=cox.ph,data=pbc)

b <- gam(time ~ trt+sex+s(sqrt(protime))+s(platelet)+
          s(age)+s(bili)+s(albumin),
          weights=status1,family=cox.ph,data=pbc)

anova(b)
par(mfrow=c(2,3))
plot(b); plot(b$linear.predictors,residuals(b))

par(mfrow=c(1,1))
## create prediction data frame...
np <- 300
newd <- data.frame(matrix(0,np,0))
for (n in names(pbc)) newd[[n]] <- rep(pbc[[n]][25],np)
newd$time <- seq(0,4500,length=np)
## predict and plot the survival function...
fv <- predict(b,newdata=newd,type="response",se=TRUE)
plot(newd$time,fv$fit,type="l",ylim=c(0.,1),xlab="time",
     ylab="survival",lwd=2)
## add crude one s.e. intervals...
lines(newd$time,fv$fit+fv$se.fit,col="grey")
lines(newd$time,fv$fit-fv$se.fit,col="grey")
## and intervals based on cumulative hazard s.e...
se <- fv$se.fit/fv$fit
lines(newd$time,exp(log(fv$fit)+se))
lines(newd$time,exp(log(fv$fit)-se))

## 7.8.1 time dependent
## copy functions from ?cox.pht in mgcv...

app <- function(x,t,to) {
## wrapper to approx for calling from apply...
   y <- if (sum(!is.na(x))<1) rep(NA,length(to)) else
        approx(t,x,to,method="constant",rule=2)$y
   if (is.factor(x)) factor(levels(x)[y],levels=levels(x)) else y
} ## app

tdpois <- function(dat,event="z",et="futime",t="day",
                             status="status1",id="id") {
## dat is data frame. id is patient id; et is event time; t is
## observation time; status is 1 for death 0 otherwise;
## event is name for Poisson response.
   if (event %in% names(dat)) warning("event name in use")
   require(utils) ## for progress bar
   te <- sort(unique(dat[[et]][dat[[status]]==1])) ## event times
```

```
   sid <- unique(dat[[id]])
   prg <- txtProgressBar(min = 0, max = length(sid), initial = 0,
              char = "=",width = NA, title="Progress", style = 3)
   ## create dataframe for poisson model data
   dat[[event]] <- 0; start <- 1
   dap <- dat[rep(1:length(sid),length(te)),]
   for (i in 1:length(sid)) { ## work through patients
     di <- dat[dat[[id]]==sid[i],] ## ith patient's data
     tr <- te[te <= di[[et]][1]] ## times required for this patient
     ## Now do the interpolation of covariates to event times...
     um <- data.frame(lapply(X=di,FUN=app,t=di[[t]],to=tr))
     ## Mark the actual event...
     if (um[[et]][1]==max(tr)&&um[[status]]==1) um[[event]][nrow(um)] <- 1
     um[[et]] <- tr ## reset time to relevant event times
     dap[start:(start-1+nrow(um)),] <- um ## copy to dap
     start <- start + nrow(um)
     setTxtProgressBar(prg, i)
   }
   close(prg)
   dap[1:(start-1),]
} ## tdpois

## model fitting...

data(pbc)
pbcseq$status1 <- as.numeric(pbcseq$status==2) ## deaths
pb <- tdpois(pbcseq) ## conversion
pb$tf <- factor(pb$futime) ## add factor for event time

b <- bam(z ~ tf - 1  +  trt + s(sqrt(protime)) + s(platelet) +
         s(age) + s(bili) + s(albumin) + s(sqrt(ast)),
         family=poisson,data=pb,discrete=TRUE,nthreads=2)

chaz <- tapply(fitted(b),pb$id,sum) ## cum. hazard by subject
d <- tapply(pb$z,pb$id,sum) ## censoring indicator
mrsd <- d - chaz ## Martingale residuals
drsd <- sign(mrsd)*sqrt(-2*(mrsd + d*log(chaz))) ## deviance

te <- sort(unique(pb$futime)) ## event times
di <- pbcseq[pbcseq$id==25,] ## data for subject 25
## interpolate to te using app from ?cox.pht...
pd <- data.frame(lapply(X=di,FUN=app,t=di$day,to=te))
pd$tf <- factor(te)
X <- predict(b,newdata=pd,type="lpmatrix")
eta <- drop(X%*%coef(b)); H <- cumsum(exp(eta))
J <- apply(exp(eta)*X,2,cumsum)
se <- diag(J%*%vcov(b)%*%t(J))^.5
par(mfrow=c(1,2))
plot(stepfun(te,c(1,exp(-H))),do.points=FALSE,ylim=c(0.7,1),
     ylab="S(t)",xlab="t (days)",main="",lwd=2)
lines(stepfun(te,c(1,exp(-H+se))),do.points=FALSE)
lines(stepfun(te,c(1,exp(-H-se))),do.points=FALSE)
rug(pbcseq$day[pbcseq$id==25]) ## measurement times
```

```
er <- pbcseq[pbcseq$id==25,]
plot(er$day,er$ast);lines(te,pd$ast)

## 7.9 Location scale

library(MASS);library(mgcv)
b <- gam(list(accel~s(times,bs="ad"),~s(times,bs="ad")),
         family=gaulss,data=mcycle)

## 7.9.1 Extreme rainfall
library(mgcv);library(gamair);data(swer)
b0 <- gam(list(exra ~ s(nao)+ s(elevation)+ climate.region+
                      te(N,E,year,d=c(2,1),k=c(20,5)),
      ~ s(year)+ s(nao)+ s(elevation)+ climate.region+ s(N,E),
      ~ s(elevation)+ climate.region),family=gevlss,data=swer)

b <- gam(list(exra~ s(nao)+s(elevation)+climate.region+s(N,E),
         ~ s(year)+ s(elevation)+ climate.region+ s(N,E),
        ~ climate.region),family=gevlss,data=swer)
plot(b,scale=0,scheme=c(1,1,3,1,1,3),contour.col="white",pages=1)

mu <- fitted(b)[,1];rho <- fitted(b)[,2]; xi <- fitted(b)[,3]
fv <- mu + exp(rho)*(gamma(1-xi)-1)/xi

Fi.gev <- function(z,mu,sigma,xi) { ## GEV inverse cdf.
  xi[abs(xi)<1e-8] <- 1e-8 ## approximate xi=0, by small xi
  x <- mu + ((-log(z))^-xi-1)*sigma/xi
}
mb <- coef(b);Vb <- vcov(b) ## posterior mean and cov
b1 <- b ## copy fitted model object to modify
n.rep <- 1000; br <- rmvn(n.rep,mb,Vb) ## posterior sim
n <- length(fitted(b))
sim.dat <- cbind(data.frame(rep(0,n*n.rep)),swer$code)
for (i in 1:n.rep) {
  b1$coefficients <- br[i,] ## copy sim coefs to gam object
  X <- predict(b1,type="response");ii <- 1:n + (i-1)*n
  sim.dat[ii,1] <- Fi.gev(runif(n),X[,1],exp(X[,2]),X[,3])
}

stm <- tapply(sim.dat[,1],sim.dat[,2],mean)
st98 <- tapply(sim.dat[,1],sim.dat[,2],quantile,probs=0.98)

## 7.10 Multivariate
library(mgcv); library(gamair); data(mpg)
b <- gam(list(city.mpg ~ fuel +style +drive +s(weight) +s(hp)
                         + s(make,bs="re"),
                hw.mpg ~ fuel +style +drive +s(weight) +s(hp)
                         + s(make,bs="re")),
               family = mvn(d=2) , data = mpg)

b1 <- gam(list(city.mpg ~ fuel +style +drive +s(hp) +s(weight)
                          + s(make,bs="re"),
```

```
               hw.mpg ~ fuel +style +drive +s(make,bs="re"),
                  1+2 ~ s(weight) +s(hp) -1),
             family = mvn(d=2) , data = mpg)

## 7.11 FDA
## 7.11.1 scalar-on-function
data(gas)
b <- gam(octane~s(nm,by=NIR,k=50),data=gas)
par(mfrow=c(1,2))
plot(b,scheme=1,col=1)
plot(fitted(b),gas$octane)

## Prostate...
data(prostate)
b <- gam(type ~ s(MZ,by=intensity,k=100),family=ocat(R=3),
         data=prostate,method="ML")
par(mfrow=c(1,3))
plot(b,rug=FALSE,scheme=1,xlab="Daltons",ylab="f(D)",
     cex.lab=1.6,cex.axis=1.4)
pb <- predict(b,type="response") ## matrix of class probs
plot(factor(prostate$type),pb[,3])
qq.gam(b,rep=100,lev=.95)

prostate$type1 <- prostate$type - 1 ## recode for multinom
b1 <- gam(list(type1 ~ s(MZ,by=intensity,k=100),
               ~ s(MZ,by=intensity,k=100)),
          family=multinom(K=2),data=prostate)
plot(b1,pages=1,scheme=1,rug=FALSE)

## 7.11.2 Canadian weather
require(gamair);require(lattice);data(canWeather)
xyplot(T~time|region,data=CanWeather,type="l",groups=place)

aic <- reml <- rho <- seq(0.9,0.99,by=.01)
for (i in 1:length(rho)) {
  b <- bam(T ~ region + s(time,k=20,bs="cr",by=region) +
           s(time,k=40,bs="cr",by=latitude),
           data=CanWeather,AR.start=time==1,rho=rho[i])
  aic[i] <- AIC(b); reml[i] <- b$gcv.ubre
}

## End(Not run)
```

`ch7.solutions` *Solution code for Chapter 7 GAMs in Practice: mgcv*

Description

R code for Chapter 7 exercise solutions.

Author(s)

Simon Wood <simon@r-project.org>

Maintainer: Simon Wood <simon@r-project.org>

References

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R*, CRC

See Also

`mgcv`, `ch7`

Examples

```
library(gamair); library(mgcv)

## Q.1
## a)
data(hubble)
h1 <- gam(y~s(x),data=hubble)
plot(h1) ## model is curved
h0 <- gam(y~x,data=hubble)
h1;h0
AIC(h1,h0)

## b)
gam.check(h1) # oh dear
h2 <- gam(y~s(x),data=hubble,family=quasi(var=mu))
gam.check(h2) # not great, but better
h2

## Q.2
## a)
library(MASS)
par(mfrow=c(2,2))
mc <- gam(accel~s(times,k=40),data=mcycle)
plot(mc,residuals=TRUE,se=FALSE,pch=1)

## b)
mc1 <- lm(accel~poly(times,11),data=mcycle)
termplot(mc1,partial.resid=TRUE)

## c)
mc2 <- gam(accel~s(times,k=11,fx=TRUE),data=mcycle)
plot(mc2,residuals=TRUE,se=FALSE,pch=1)

## d)
mc3 <- gam(accel~s(times,k=11,fx=TRUE,bs="cr"),data=mcycle)
plot(mc3,residuals=TRUE,se=FALSE,pch=1)

## e)
```

```
par(mfrow=c(1,1))
plot(mcycle$times,residuals(mc))

## f)
mcw <- gam(accel~s(times,k=40),data=mcycle,
           weights=c(rep(400,20),rep(1,113)))
plot(mcw,residuals=TRUE,pch=1)
rsd <- residuals(mcw)
plot(mcycle$times,rsd)
var(rsd[21:133])/var(rsd[1:20])

## g)
gam(accel~s(times,k=40,m=3),data=mcycle,
    weights=c(rep(400,20),rep(1,113)))

## Q.3
## b)
library(MASS)
n <- nrow(mcycle)
A <- matrix(0,n,n)
for (i in 1:n) {
  mcycle$y<-mcycle$accel*0;mcycle$y[i] <- 1
  A[,i] <- fitted(gam(y~s(times,k=40),data=mcycle,sp=mc$sp))
}

## d)
plot(mcycle$times,A[,65],type="l",ylim=c(-0.05,0.15))

## e)
for (i in 1:n) lines(mcycle$times,A[,i])

## f)
par(mfrow=c(2,2))
mcycle$y<-mcycle$accel*0;mcycle$y[65] <- 1
for (k in 1:4) plot(mcycle$times,fitted(
     gam(y~s(times,k=40),data=mcycle,sp=mc$sp*10^(k-1.5))
     ),type="l",ylab="A[65,]",ylim=c(-0.01,0.12))

## Q.4
## a)
par(mfrow=c(1,1))
w <- c(rep(400,20),rep(1,113))
m <- 40;par(mfrow=c(1,1))
sp <- seq(-13,12,length=m) ## trial log(sp)'s
AC1 <- EDF <- rep(0,m)
for (i in 1:m) { ## loop through s.p.'s
 b <- gam(accel~s(times,k=40),data=mcycle,weights=w,
          sp=exp(sp[i]))
 EDF[i] <- sum(b$edf)
 AC1[i] <- acf(residuals(b),plot=FALSE)$acf[2]
}
plot(EDF,AC1,type="l");abline(0,0,col=2)
```

```
## Not run:
## Q.5 - a bit slow - few seconds
## a)
data(co2s)
attach(co2s)
plot(c.month,co2,type="l")

## b)
b<-gam(co2~s(c.month,k=300,bs="cr"))

## c)
pd <- data.frame(c.month=1:(n+36))
fv <- predict(b,pd,se=TRUE)
plot(pd$c.month,fv$fit,type="l")
lines(pd$c.month,fv$fit+2*fv$se,col=2)
lines(pd$c.month,fv$fit-2*fv$se,col=2)

## d)
b2 <- gam(co2~s(month,bs="cc")+s(c.month,bs="cr",k=300),
           knots=list(month=seq(1,13,length=10)))

## e)
pd2 <- data.frame(c.month=1:(n+36),
                  month=rep(1:12,length.out=n+36))
fv <- predict(b2,pd2,se=TRUE)
plot(pd$c.month,fv$fit,type="l")
lines(pd$c.month,fv$fit+2*fv$se,col=2)
lines(pd$c.month,fv$fit-2*fv$se,col=2)

## End(Not run)

## Not run:
## Q.6 - a bit slow - a few seconds
data(ipo)
n<-nrow(ipo)
## create lagged variables ...
ipo$ir1 <- c(NA,ipo$ir[1:(n-1)])
ipo$ir2 <- c(NA,NA,ipo$ir[1:(n-2)])
ipo$ir3 <- c(NA,NA,NA,ipo$ir[1:(n-3)])
ipo$ir4 <- c(NA,NA,NA,NA,ipo$ir[1:(n-4)])
ipo$dp1 <- c(NA,ipo$dp[1:(n-1)])
ipo$dp2 <- c(NA,NA,ipo$dp[1:(n-2)])
ipo$dp3 <- c(NA,NA,NA,ipo$dp[1:(n-3)])
ipo$dp4 <- c(NA,NA,NA,NA,ipo$dp[1:(n-4)])
## fit initial model and look at it ...
b<-gam(n.ipo~s(ir1)+s(ir2)+s(ir3)+s(ir4)+s(log(reg.t))+
   s(dp1)+s(dp2)+s(dp3)+s(dp4)+s(month,bs="cc")+s(t,k=20),
   data=ipo,knots=list(month=seq(1,13,length=10)),
   family=poisson,gamma=1.4)
par(mfrow=c(3,4))
plot(b,scale=0)
summary(b)
## re-fit model dropping ir4 ...
```

```
b1 <- gam(n.ipo~s(ir1)+s(ir2)+s(ir3)+s(log(reg.t))+s(dp1)+
          s(dp2)+s(dp3)+s(dp4)+s(month,bs="cc")+s(t,k=20),
          data=ipo,knots=list(month=seq(1,13,length=10)),
          family=poisson,gamma=1.4)
par(mfrow=c(3,4))
plot(b1,scale=0)
summary(b1)
## residual checking ...
gam.check(b1)
par(mfrow=c(1,1))
acf(residuals(b1))

## End(Not run)

## Q.7
data(wine)
wm<-gam(price~s(h.rain)+s(s.temp)+s(h.temp)+s(year),
    data=wine,family=Gamma(link=identity),gamma=1.4)
plot(wm,pages=1,residuals=TRUE,pch=1,scale=0)
acf(residuals(wm))
gam.check(wm)
predict(wm,wine,se=TRUE)

## Q.8
## a)
par(mfrow=c(1,1))
data(blowfly)
bf <- blowfly
plot(bf$day,bf$pop,type="l")

## b)
## prepare differenced and lagged data ...
n <- nrow(bf)
bf$dn <- c(NA,bf$pop[2:n]-bf$pop[1:(n-1)])
lag <- 6
bf$n.lag <- c(rep(NA,lag),bf$pop[1:(n-lag)])
bf1 <- bf[(lag+1):n,] # strip out NAs, for convenience
## fit model, note no intercept ...
b<-gam(dn~n.lag+pop+s(log(n.lag),by=n.lag)+
       s(log(pop),by=-pop)-1,data=bf1)
plot(b,pages=1,scale=-1,se=FALSE) ## effects
plot(abs(fitted(b)),residuals(b))
acf(residuals(b))

## c)
fv <- bf$pop
e <- rnorm(n)*0 ## increase multiplier for noisy version
min.pop <- min(bf$pop);max.pop <- max(bf$pop)
for (i in (lag+1):n) { ## iteration loop
  dn <- predict(b,data.frame(n.lag=fv[i-lag],pop=fv[i-1]))
  fv[i] <- fv[i-1]+dn + e[i];
  fv[i]<-min(max.pop,max(min.pop,fv[i]))
}
```

```
plot(bf$day,fv,type="l")

## Not run:
## Q.9 - takes several minutes
## a)
data(chl)
pairs(chl,pch=".")

## b)
fam <- quasi(link=log,var=mu^2)
cm <- gam(chl ~ s(I(chl.sw^.4),bs="cr",k=20)+
      s(I(bath^.25),bs="cr",k=60)+s(jul.day,bs="cr",k=20),
      data=chl,family=fam,gamma=1.4)
gam.check(cm)
summary(cm)

## c)
## create fit and validation sets ...
set.seed(2)
n<-nrow(chl);nf <- floor(n*.9)
ind <- sample(1:n,nf,replace=FALSE)
chlf <- chl[ind,];chlv <- chl[-ind,]
## fit to the fit set
cmf<-gam(chl ~ s(I(chl.sw^.4),bs="cr",k=20)+
     s(I(bath^.25),bs="cr",k=60)+s(jul.day,bs="cr",k=20),
     data=chlf,family=fam,gamma=1.4)
## evaluate prop. dev. explained for validation set
y <- chlv$chl;w <- y*0+1
mu <- predict(cmf,chlv,type="response")
pred.dev <- sum(fam$dev.resids(y,mu,w))
null.dev <- sum(fam$dev.resids(y,mean(y),w))
1-pred.dev/null.dev # prop dev. explained

## End(Not run)

## Not run:
## Q.10 - a few seconds run time
## a)
g1<-gamm(weight ~ Variety + s(Time)+
    s(Time,by=ordered(Variety)),data=Soybean,
    family=Gamma(link=log), random=list(Plot=~Time))
plot(g1$lme) ## standard mean variance plot
par(mfrow=c(1,3))
plot(g1$gam,residuals=TRUE,all.terms=TRUE,scale=0) ## gam plot

## b)
summary(g1$gam) ## evidence for variety dependence
## could also do following ....
g2 <- gamm(weight~s(Time),family=Gamma(link=log),
      data=Soybean,random=list(Plot=~Time))
g3 <- gamm(weight~Variety+s(Time),family=Gamma(link=log),
      data=Soybean,random=list(Plot=~Time))
## following only a rough guide, but also supports g1 ...
```

```
AIC(g1$lme,g2$lme,g3$lme)

## Q.11
data(med); head(med) ## look at data
data(coast)

## initial plots...
plot(med$lo,med$la,cex=0.2+med$count^.5/10,col="grey",
     pch=19,xlab="lo",ylab="la",main="mackerel")
ind <- med$count==0
points(med$lo[ind],med$la[ind],cex=0.1,pch=19)
lines(coast)
## ... survey seems to cover spawning area this time!

require(mgcv)
m1 <- gam(count~s(lo,la,k=100)+s(T.surf)+s(T.20)+s(I(b.depth^.5))+s(Sal20)+
          s(ship,bs="re")+offset(log(vol)),data=med,select=TRUE,family=tw)
gam.check(m1) ## mean variance relationship not quite right?

m2 <- gam(count~s(lo,la,k=100)+s(T.surf)+s(T.20)+s(I(b.depth^.5))+s(Sal20)+
          s(ship,bs="re")+offset(log(vol)),data=med,select=TRUE,family=nb)
gam.check(m2)

par(mfrow=c(1,2)) ## re-check residuals vs fitted
plot(fitted(m1)^.5,residuals(m1));plot(fitted(m2)^.5,residuals(m2))

AIC(m1,m2) ## neg bin much better
plot(m2,pages=1) ## effects


## End(Not run)
```

`chicago` *Chicago air pollution and death rate data*

Description

Daily air pollution and death rate data for Chicago.

Usage

```
data(chicago)
```

Format

A data frame with 7 columns and 5114 rows. Each row refers to one day. The columns are:

death total deaths (per day).

pm10median median particles in 2.5-10 per cubic m

pm25median median particles < 2.5 mg per cubic m (more dangerous).

o3median Ozone in parts per billion

so2median Median Sulpher dioxide measurement

time time in days

tmpd temperature in fahrenheit

Details

See the `NMMAPSdata` package for fuller details. Note that there are missing values in some fields.

Source

Roger D. Peng, Leah J. Welty and Aiden McDermott. R package NMMAPSdata.

References

Peng, R.D. and Welty, L.J. (2004) The NMMAPSdata package. R News 4(2).

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R

`chl` *Chlorophyll data*

Description

Data relating to the callibration of remote sensed satellite data. The SeaWifs satellite provides estimates of chlorophyll concentration at the ocean surface from measurements of ocean surface colour. It is of interest to attempt to use these data to predict direct bottle measurements of chl. conc.

Usage

```
data(chl)
```

Format

A data frame with 6 columns and 13840 rows. The columns are:

lon longitude

lat latitude

jul.day Julian day (i.e. day of year starting at Jan 1st.)

bath Ocean depth in metres.

chl direct chlorophyll concentration measured at given location from a bottle sample.

chl.sw chl. conc. as measured by Seawifs Satellite

Source

https://oceancolor.gsfc.nasa.gov/SeaWiFS/

and the World Ocean Database.

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
data(chl)
with(chl,plot(chl,chl.sw))
```

`co2s` *Atmospheric CO2 at South Pole*

Description

Monthly CO2 concentration in parts per million at the South Pole.

Usage

`data(co2s)`

Format

A data frame with 3 columns and 507 rows. The columns are:

co2	atmospheric CO2 concentration in parts per million
c.month	cumulative number of months since Jan 1957
month	month of year

Source

http://cdiac.esd.ornl.gov/trends/co2/

References

Keeling C.P. and T.P Whorf (2000) Atmospheric CO2 records from sites in the SIO air sampling network. In Trends: A Compedium of Data on Global Change. Carbon Dioxide Analyis Center, Oak Ridge National Laboratory, U.S. Department of Energy, Oak Ridge Tenn., USA

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
data(co2s)
with(co2s,plot(c.month,co2,type="l",ylab=
expression(paste(CO[2]," in ppm.")),xlab="Month since Jan. 1957"))
```

`coast` *European coastline from -11 to 0 East and from 43 to 59 North*

Description

The data are longitudes (degrees E) and latitudes (degrees N) defining points that can be joined up to get the European coastline in the rectangle (-11E,43N)-(0E,59N). Discontinuous sections of coast are separated by NA's.

Usage

```
data(coast)
```

Format

A data frame with 2 columns.

lon Longitude in degrees East for points used to define the coast.

lat Latitude in degrees North for points used to define the coast.

Details

`lon`, `lat` together define the co-ordinates of points that can be joined up in order to plot the coastline. The original data come from the NOAA www site given below, but have been substantially thinned, to a much lower resultion than the source.

Author(s)

Simon Wood.

References

Originally from... `http://rimmer.ngdc.noaa.gov/coast/`

Examples

```
data(coast)
# plot the entire coast .....
plot(coast$lon,coast$lat,type="l")
# or draw it clipped to  whatever the current plot is....
lines(coast$lon,coast$lat,col="blue")
```

`engine` *Engine wear versus size data*

Description

Data on engine wear against engine size for 19 Volvo car engines.

Usage

```
data(engine)
```

Format

A data frame with 2 columns and 19 rows. Each row refers to one engine model. The columns are:

wear an index of engine wear rate.

size cylinder capacity in litres.

Details

See the source for further details.

Source

Originally from... `http://www3.bc.sympatico.ca/Volvo_Books/engine3.html`

`gas` *Octane rating data*

Description

The octane rating of fuel determines its ‘knocking’ resistance. So the higher the octane rating the higher the compression ratio that an engine can run at. Traditionally octane measurement involves comparing the knocking resistance of fuel samples to standard mixtures in special variable compression ratio engines. This is an expensive process relative to obtaining the near infra-red spectrum of a sample. It would be good to be able to predict octane rating from the spectrum.

Usage

```
data(gas)
```

Format

A three item list

octane Octane rating of gasoline (petrol) sample.

NIR A matrix each row of which contains the near infra-red reflectance spectrum of the corresponding gasoline sample.

nm Matrix of same dimension as `NIR` containing wavelengths at which measurements were taken.

Details

A scalar-on-function regression (also known as 'signal regression') works quite well for these data.

Source

Originally from the `pls` package

https://cran.r-project.org/package=pls

Examples

```
require(gamair);require(mgcv)
data(gas)
## plot some spectra...
with(gas,plot(nm[1,],NIR[1,],type="l",ylab="log(1/R)",
     xlab="wavelength (nm)",col=1))
text(1000,1.2,"octane");text(1000,1.2-.1,gas$octane[1],col=1)
for (i in 2:8) { lines(gas$nm[i,],gas$NIR[i,],col=i)
  text(1000,1.2-.1*i,gas$octane[i],col=i)
}

## Fit scalar on function regression...

b <- gam(octane~s(nm,by=NIR,k=50),data=gas)

gam.check(b)

par(mfrow=c(1,2))
plot(b,scheme=1)
plot(fitted(b),gas$octane,xlab="fitted octane",
     ylab="observed octane");abline(0,1)
```

`harrier` *Hen Harriers Eating Grouse*

Description

Data on the rate at which Hen Harriers consume Grouse as a function of Grouse density.

Usage

```
data(harrier)
```

Format

A data frame with 2 columns and 37 rows. The columns are:

Grouse.Density Density of Grouse per square kilometre.

Consumption.Rate Number of Grouse consumed per Hen Harrier per day.

Details

Data have been read from Figure 1 of Asseburg et al. (2005)

Source

Asseburg, C., S. Smout, J. Matthiopoulos, C. Fernandez, S. Redpath, S. Thirgood and J. Harwood (2005) The functional response of a generalist predator. Web preprint

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
data(harrier)
with(harrier,plot(Grouse.Density,Consumption.Rate))
```

`hubble` *Hubble Space Telescope Data*

Description

Data on distances and velocities of 24 galaxies containing Cepheid stars, from the Hubble space telescope key project to measure the Hubble constant.

Usage

```
data(hubble)
```

Format

A data frame with 3 columns and 24 rows. The columns are:

Galaxy A (factor) label identifying the galaxy.

y The galaxy's relative velocity in kilometres per second.

x The galaxy's distance in Mega parsecs. 1 parsec is 3.09e13 km.

Details

Cepheids are variable stars which have a known relationship between brightness and period. Hence the distance to galaxies containing these stars can be estimated from the observed brightness of the Cepheid, relative to its absolute brightness as predicted by its period. The velocity of the galaxy can be estimated from its mean red-shift.

The data can be used to get a reasonably good idea of the age of the universe. A data free alternative estimate of 6000 years is given in the reference (not the source!).

Source

Tables 4 and 5 of Freedman et al. 2001. The Astrophysical Journal 553:47-72

References

Freedman et al. (2001) Final results from the Hubble space telescope key project to measure the Hubble constant. The Astrophysical Journal (553), 47-72.

http://www.icr.org/pubs/imp/imp-352.htm

NUCLEAR DECAY: EVIDENCE FOR A YOUNG WORLD - IMPACT No. 352 October 2002 by D. Russell Humphreys, Ph.D.

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

`ipo` *Initial Public Offering Data*

Description

Data on the relationship between the number of initial public offerings (of shares in a company) and other potentially important variables. It is probably necessary to lag some of the explanatory variables.

Usage

```
data(ipo)
```

Format

A data frame with 6 columns and 156 rows. The columns are:

n.ipo number of initial pubilc offerings each month.

ir the average initial return (volume weighted): this is the percentage difference between the offer proce of shares and the price after the first day of trading.

dp the average percentage difference between middle of the price range proposed at first filing of the IPO, and the eventual offer price.

reg.t the average time between filing and offer.

t time, in months.

month month of the year (1 = January).

Source

http://schwert.ssb.rochester.edu

References

Lowry, M. and G.W. Schwert (2002) IPO market cycles: Bubbles or sequential learning? The Journal of Finance 67(3), 1171-1198

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
data(ipo)
pairs(ipo)
```

`Larynx` *Cancer of the larynx in Germany*

Description

The data give counts of deaths from cancer of the Larynx by region of Germany from 1986 to 1990, along with the expected count according to the populaiton of the region and the total deaths for the whle of Germany. A list of polygons defining the boundaries of the districts is also provided.

Usage

```
data(larynx)
data(german.polys)
```

Format

The `Larynx` data frame has the following columns

region A factor with 544 levels identifying the health reporting region.

E Expected number of deaths according to population of region and pan-German total.

Y Number of deaths from cancer of the Larynx in the region.

x A measure of level of smoking in the region.

`german.polys` is a list with one item per health reporting region in `Larynx`. The name of each item identifies the region using the same labels as `Larynx$region`. Each item is a two column matrix defining a polygon approximating the outline of the region it relates to. Each row of the matrix defines a polygon vertex. `NA` rows separate geographically disjoint areas which are part of the same region.

Details

Note that the polygons are set up to exactly share vertices with their neighbours, which facilitates the auto-identification of neighbourhood structures.

Source

Data are from the INLA website:

http://www.r-inla.org/

Examples

```
require(gamair);require(mgcv)
data(larynx);data(german.polys)

## plot raw deaths over expected deaths by region...
polys.plot(german.polys,Larynx$Y/Larynx$E)

## Fit additive model with Gauss MRF for space and smooth of
## smoking level. k somewhat low to reduce computational time
b <- gam(Y~s(region,k=60,bs="mrf",xt=list(polys=german.polys)) +
offset(log(E))+s(x,k=10),family=poisson,data=Larynx,method="REML")

summary(b)
plot(b,scheme=c(0,1),pages=1)
```

---

mack *Egg data from 1992 mackerel survey*

Description

The data relate to the distribution of mackerel eggs and were collected as part of the 1992 mackerel survey aimed at assessing the mackerel spawning stock biomass using the daily egg production method.

Usage

```
data(mack)
```

Format

A data frame with 16 columns. Each row corresponds to one sample of eggs.

egg.count The number of stage I eggs in this sample.

egg.dens The number of stage I eggs per square metre of sea surface, produced per day. This is calculated from `egg.count` and other information about sampling net size, and egg stage duration.

b.depth The sea bed depth at the sampling location.

c.dist The distance from the sample location to the 200m contour measured in degrees as if degrees latitude equalled degrees longitude, which actually they don't.

lon The longitude of the sample station in degrees east.

lat The latitude of the sample station in degrees north.

time The time of day (in hours) at which the sample was taken.

salinity The salinity (saltiness) of the water at the sampling location.

flow Reading from the flow meter attached to the sampling net - used for calibration.

s.depth The depth that the sampling net started sampling from (the net is dropped to this depth and then hauled up to the surface, filtering eggs etc out of the water as it goes).

temp.surf The temperature at the sea surface at the sampling location.

temp.20m The temperature 20m down at the sampling location.

net.area The area of the sampling net in square metres.

country A code identifying the country responsible for the boat that took this sample.

vessel A code identifying the boat that took this sample.

vessel.haul A code uniquely identifying this sample, given that the vessel is known.

Details

At each of a number of stations located as defined in `lon` and `lat`, mackerel eggs were sampled by hauling a fine net up from deep below the sea surface to the sea surface. The egg count data are obtained from the resulting samples, and these have been converted to (stage I) eggs produced per metre squared per day - the egg density data. Other possibly useful predictor variable information has been recorded, along with identification information, and some information that is probably useless!

Source

The data are effectively a combination of datasets `mackerel` and `smacker` from the `sm` library. They were originally analyzed using GAMs by:

Borchers, D.L., S.T. Buckland, I.G. Priede and S. Ahmadi (1997) "Improving the precision of the daily egg production method using generalized additive models". Can. J. Fish. Aquat. Sci. 54:2727-2742.

Examples

```
data(mack)
# plot the egg densities against location
plot(mack$lon,mack$lat,cex=0.2+mack$egg.dens/150,col="red")
```

`mackp` *Prediction grid data for 1992 mackerel egg model*

Description

This data frame provides a regular grid of values of some predictor variables useful for modelling mackerel egg abundances. Its main purpose is to enable mackerel egg densities to be predicted over a regular spatial grid within the area covered by the 1992 mackerel egg survey (see `mack`), using a fitted generalised additive model.

Usage

```
data(mackp)
```

Format

A data frame with 5 columns. Each row corresponds to one spatial location within the survey area. The columns are as follows:

lon Longitude of the gridpoint in degrees east

lat Latitude of the gridpoint in degrees north.

b.depth The sea bed depth at the gridpoint.

c.dist The distance from the gridpoint to the 200m sea bed depth contour.

salinity Salinity interpolated onto the grid (from `mack` measurements).

temp.surf Surface temperature interpolated onto grid (from `mack` data).

temp.20m Temperature at 20m interpolated from `mack` data.

area.index An indexing vector that enables straightforward copying of the other variables into a matrix suitable for plotting against longitude and lattitude using `image()`. See the example below.

Details

The grid is defined on a series of 1/4 degree lon-lat squares.

References

Borchers, D.L., S.T. Buckland, I.G. Priede and S. Ahmadi (1997) "Improving the precision of the daily egg production method using generalized additive models". Can. J. Fish. Aquat. Sci. 54:2727-2742.

Examples

```
## example of how to use `area.index' to paste gridded info.
## into a square grid (of NA's) for plotting
data(mackp)
lon<-seq(-15,-1,1/4);lat<-seq(44,58,1/4)
zz<-array(NA,57*57)
zz[mackp$area.index]<-mackp$b.depth
image(lon,lat,matrix(zz,57,57))
```

`meh` *Data from 2010 horse mackerel and mackerel egg survey*

Description

The data relate to the distribution of horse mackerel (`meh`, Trachurus trachurus) eggs and mackerel (`med`, Scomber scombrus) eggs and were collected as part of the 2010 mackerel survey aimed at assessing the mackerel spawning stock biomass using the daily egg production method.

Usage

```
data(med)
data(meh)
```

Format

A data frame with the following columns. Each row corresponds to one sample of eggs.

count The number of stage I eggs in this sample.

la sample station latitude

lo sample station longitude

vol volume of water sampled

T.surf surface temperature in centigrade

T.x temperature at x metres depth.

T1.x Second temperature measurements.

Sal20 Salinity at 20m depth

b.depth seabed depth in metres for `med` only.

lon The longitude of the sample station in degrees east.

lat The latitude of the sample station in degrees north.

time The time of day (in hours) at which the sample was taken.

salinity The salinity (saltiness) of the water at the sampling location.

period sampling period

country Country responsible for sample

ship Vessel ID

DT sample data and time

ID Sample ID

gear type of sampling gear used

The remaining fields are undocumented.

Details

The original data files do not always exactly match the file documentation, so these data should not be treated as definitive.

Source

ICES Eggs and Larvae Dataset 2012, ICES, Copenhagen

http://www.ices.dk/

http://eggsandlarvae.ices.dk/Download.aspx

Examples

```
require(gamair)
par(mfrow=c(1,2))
data(meh);data(med);data(coast)
# plot the egg counts against location
plot(meh$lo,meh$la,cex=0.2+meh$count^.5/10,col="grey",
     pch=19,xlab="lo",ylab="la",main="horse mackerel")
ind <- meh$count==0
points(meh$lo[ind],meh$la[ind],cex=0.1,pch=19)
lines(coast)

# same for med
plot(med$lo,med$la,cex=0.2+med$count^.5/10,col="grey",
     pch=19,xlab="lo",ylab="la",main="mackerel")
ind <- med$count==0
points(med$lo[ind],med$la[ind],cex=0.1,pch=19)
lines(coast)
```

`mpg` *Data on automobile efficiency on town streets and highway.*

Description

Fuel efficiency in miles per gallon for a variety of cars in the USA.

Usage

```
data(mpg)
```

Format

A data frame listing fuel efficiency and other car characteristics including

symbol Insurers measure of relative riskiness of car from -3 (safe) to 3 (risky)

loss average insurance loss payment per insured vehicle per year.

hw.mpg Fuel consumption on highway as miles per US gallon.

city.mpg Fuel consumption in town as miles per US gallon.

make Name of car maker.

fuel 2 level factor. `gas` or `diesel`.

aspir 2 level factor. `std` or `turbo`.

doors 2 level factor. `two` or `four`.

style Factor indicating style of car.

drive 3 level factor indicating front, rear or all wheel drive: `fwd`, `rwd` or `4wd`.

eng.loc Engine location

wb wheel base in inches

length in inches

width in inches

height in inches

weight in pounds

eng.type Factor giving engine type

cylinders Factor for number of cylinders

eng.cc cubic capicity of engine in cubic inches.

fuel.sys fuel system

bore in inches

stroke in inches

comp.ratio compression ratio

hp horse power

rpm maximum RPM

price in US dollars

Details

Data were collected by Jeffrey C. Schlimmer from 1) 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook. 2) Personal Auto Manuals, Insurance Services Office, 160 Water Street, New York, NY 10038 3) Insurance Collision Report, Insurance Institute for Highway Safety, Watergate 600, Washington, DC 20037

Source

`https://archive.ics.uci.edu/ml/datasets/Automobile`

References

Wood, S.N. (2006) Generalized Additive Models: An Introduction with R

Examples

```
require(gamair);require(mgcv)
data(mpg)
b <- gam(list(city.mpg~fuel+style+drive+s(weight)+s(hp)+s(make,bs="re"),
         hw.mpg~fuel+style+drive++s(weight)+s(hp)+s(make,bs="re")),
         family=mvn(d=2),data=mpg)
plot(b,pages=1,scheme=1)
```

`prostate` *Prostate cancer screening data*

Description

Protein mass spectographs for patients with normal, benign enlargement and cancer of the prostate gland.

Usage

```
data(prostate)
```

Format

A three item list

type 1 for normal, 2 for benign enlargement and 3 for cancerous.

intensity A matrix with rows corresponding to measurements in `type`. Each row is a normalized spectral intensity measurement for the protein mass given in `MZ`

MZ Matrix corresponding to `intensity` giving the protein masses in Daltons.Actually all rows are identical.

Details

See the source article for fuller details. The intensity data here have been smoothed so that each measurement is an average of 40 adjacent measurements from the raw spectrum. The intensity data have also been rounded to 3 significant figures. This pre-processing was done to reduce the dataset size to something reasonable for distribution.

Source

Originally from the msProstate package version 1.0.2.

References

Adam, B-L. Y. Qu, J.W. Davis et al. (2002) Serum Protein Fingerprinting Coupled with a Pattern-matching Algorithm Distinguishes Prostate Cancer from Benign Prostate Hyperplasia and Healthy Men. Cancer Research 62:3609-3614

Examples

```
require(gamair);require(mgcv)
data(prostate)
## plot some spectra...
par(mfrow=c(2,3),mar=c(5,5,3,1))
ind <- c(1,163,319)
lab <- list("Healthy","Enlarged","Cancer")
for (i in 1:3) {
```

```
  plot(prostate$MZ[ind[i],],prostate$intensity[ind[i],],type="l",ylim=c(0,60),
  xlab="Daltons",ylab="Intensity",main=lab[[i]],cex.axis=1.4,cex.lab=1.6)
  lines(prostate$MZ[ind[i],],prostate$intensity[ind[i]+2,]+5,col=2)
  lines(prostate$MZ[ind[i],],prostate$intensity[ind[i]+4,]+10,col=4)
}
## treat as ordered cat control, bph, cancer
b <- gam(type ~ s(MZ,by=intensity,k=100),family=ocat(R=3),
         data=prostate,method="ML")
## results...
pb <- predict(b,type="response")
plot(b,rug=FALSE,scheme=1,xlab="Daltons",ylab="f(D)",
cex.lab=1.6,cex.axis=1.4,main="a")
plot(factor(prostate$type),pb[,3],cex.lab=1.6,cex.axis=1.4,main="b")
qq.gam(b,rep=100,lev=.95,cex.lab=1.6,cex.axis=1.4,main="c")
```

`sitka` *Sitka spruce growth data.*

Description

Tree growth data under enhanced ozone and control conditions.

Usage

```
data(sitka)
```

Format

A data frame with 1027 rows and 5 columns columns:

id.num identity of the tree: 1...79.

order time order ranking within each tree.

days since 1st January, 1988.

log.size log of tree ‘size’.

ozone 1 - enhanced ozone treatment; 0 - control.

Details

The data were analysed in Crainiceanu CM, Ruppert D, Wand MP (2005) using WinBUGS, and in Wood (2016) using auto-generated JAGS code.

Source

The `SemiPar` package, from:

Diggle, P.J, Heagery, P., Liang, K.-Y. and Zeger, S.L. (2002) Analysis of Longitudinal Data (2nd ed.) OUP.

References

Wood SN (2016) "Just Another Gibbs Additive Modeller: Interfacing JAGS and mgcv" Journal of Statistical Software 75

Crainiceanu C.M., Ruppert D. and Wand M.P. (2005). "Bayesian Analysis for Penalized Spline Regression Using WinBUGS." Journal of Statistical Software, 14(14).

Examples

```
require(gamair); require(lattice)
data(sitka)
xyplot(log.size~days|as.factor(ozone),data=sitka,type="l",groups=id.num)
```

`sole` *Sole Eggs in the Bristol Channel*

Description

Data on Sole Egg densities in the Bristol Channel (West Coast of England, UK.) The data are from 5 research cruises undertaken for the purpose of measuring Sole egg densities. Samples were taken at each of a number of sampling stations, by hauling a net vertically through the water column. Sole eggs were counted and assigned to one of four developmental stages.

Usage

```
data(sole)
```

Format

A data frame with 7 columns and 1575 rows. The columns are:

la	latitude of sampling station
lo	longitude of sampling station
t	time of sampling station: actually time of midpoint of the cruise on which this sample was taken. Measured in Julian days (days since January 1st).
eggs	egg density per square metre of sea surface.
stage	to which of 4 stages the sample relates.
a.0	lower age limit for the stage (i.e. age of youngest possible egg in this sample).
a.1	upper age limit of this stage (i.e. age of oldest possible egg in sample).

Source

Dixon (2003)

References

Dixon, C.E. (2003) Multi-dimensional modelling of physiologically and temporally structured populations. PhD thesis. University of St Andrews

Horwood, J. (1993) The Bristol Channel Sole (solea solea (L.)): A fisheries case study. Advances in Marine Biology 29, 215-367

Horwood, J. and M. Greer Walker (1990) Determinacy of fecundity in Sole (solea solea) from the Bristol Channel. Journal of the Marine Biology Association of the United Kingdom. 70, 803-813.

Wood (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
require(gamair)
data(sole);data(coast)
par(mfrow=c(2,3))
sample.t <- unique(sole$t)
stage <- 1
for (i in 1:5)
{ egg<-sole[sole$stage==stage&sole$t==sample.t[i],]
  plot(egg$lo,egg$la,xlab=”lo”,ylab=”la”,main=paste(”day”,sample.t[i]),cex=egg$eggs/4,
  xlim=range(sole$lo),ylim=range(sole$la),cex.axis=1.5,cex.lab=1.5,cex.main=1.5)
  points(egg$lo,egg$la,pch=”.”,col=2)
  lines(coast)
}
## boundary definition list and knots suitable for soap film smoothing
bnd <- list(list(lo=c(-6.74,-5.72,-5.7 ,-5.52,-5.37,-5.21,-5.09,-5.02,
        -4.92,-4.76,-4.64,-4.56,-4.53,-4.3,-4.16,-3.8 ,-3.8,-5.04,-6.76,
-6.74),
        la=c(50.01,50.02,50.13,50.21,50.24,50.32,50.41,50.54,50.59,50.64,
50.74,50.86,51.01,51  ,51.2,51.22,51.61,51.7,51.7,50.01)))

knt <- list(lo=c(-4.643,-5.172,-5.638,-6.159,-6.665,-6.158,-5.656,-5.149,
-4.652,-4.154,-3.901,-4.146,-4.381,-4.9,-5.149,-5.37,-5.866,-6.36,-6.635,
-6.12,-5.626,-5.117,-4.622,-4.695,-4.875,-5.102,-5.609,-5.652,-5.141,
-5.354,-5.843,-6.35,-6.628,-6.127,-5.63,-5.154,-5.356,-5.652,-5.853,
-6.123),
 la=c(51.626,51.61,51.639,51.638,51.376,51.377,51.373,51.374,51.374,
 51.376,51.379,51.226,51.129,51.194,51.083,51.147,51.129,51.151,50.901,
 50.891,50.959,50.958,50.942,50.728,50.676,50.818,50.825,50.684,50.693,
 50.568,50.564,50.626,50.397,50.451,50.443,50.457,50.325,50.193,50.322,
 50.177))

 points(knt$lo,knt$la,pch=19,col=2,cex=.6)
 lines(bnd[[1]]$lo,bnd[[1]]$la,col=2)
```

`sperm.comp1` *Sperm competition data I*

Description

Data relating sperm count to time since last inter-pair copulation and proportion of that time spent together for 15 couples living in Manchester UK.

Usage

```
data(sperm.comp1)
```

Format

A data frame with 4 columns and 15 rows. The columns are:

subject An identifier for the subject/couple.

time.ipc Time since last inter-pair copulation, in hours.

prop.partner Proportion of `time.ipc` that the couple had spent together.

count Sperm count in millions.

Details

The sperm counts reported are total counts in ejaculate from a single copulation, for each of 15 couples. Also recorded are the time since the couple's previous copulation, and the proportion of that time that the couple had spent together. The data are from volunteers from Manchester University and were gathered to test theories about human sperm competition. See the source article for further details.

Source

Baker, RR and Bellis M.A. (1993) 'Human sperm competition: ejaculate adjustment by males and the function of masturbation'. Animal behaviour 46:861-885

`sperm.comp2` *Sperm competition data II*

Description

Data relating average number of sperm ejaculated per copulation to physical characterisics of partners involved, for 24 heterosexual couples from Manchester, UK.

Usage

```
data(sperm.comp2)
```

Format

A data frame with 10 columns and 24 rows. The columns are:

pair an identifier for the couple. These labels correspond to those given in `sperm.comp1`.

n the number of copulations over which the average sperm count has been calculated.

count the average sperm count in millions, per copulation.

f.age age of the female, in years.

f.height height of the female, in cm.

f.weight weight of the female, in kg.

m.age age of the male, in years.

m.height height of the male, in cm.

m.weight weight of the male, in kg.

m.vol volume of one male teste in cubic cm.

Details

In the source article, these data are used to argue that males invest more reproductive effort in heavier females, on the basis of regression modelling. It is worth checking for outliers.

Source

Baker, RR and Bellis M.A. (1993) 'Human sperm competition: ejaculate adjustment by males and the function of masturbation'. Animal behaviour 46:861-885

`stomata` *Stomatal area and CO2*

Description

Fake data on average stomatal area for 6 trees grown under one of two CO2 concentrations

Usage

```
data(stomata)
```

Format

A data frame with 3 columns and 24 rows. The columns are:

area mean stomatal area.

CO2 label for which CO2 treatment the measurement relates to.

tree label for individual tree.

Details

The context for these simulated data is given in section 6.1 of the source book.

Source

The reference.

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

`swer` *Swiss 12 hour extreme rainfall*

Description

Records the most extreme 12 hourly total rainfall each year for 65 Swiss weather stations. The data period is 1981-2015, although not all stations start in 1981.

Usage

```
data(swer)
```

Format

The `swer` data frame has the following columns

year The year of observation.

exra The highest rainfall observed in any 12 hour period in that year, in mm.

nao Annual North Atlantic Oscillation index, based on the difference of normalized sea level pressure (SLP) between Lisbon, Portugal and Stykkisholmur/Reykjavik, Iceland. Positive values are generally associated with wetter and milder weather over Western Europe.

location The measuring station location name.

code Three letter code identifying the station.

elevation metres above sea level.

climate.region One of 12 distinct climate regions.

N Degrees north.

E Degrees east.

Details

The actual extreme rainfall measurements are digitized from plots in the MeteoSwiss reports for each station. The error associated with digitization can be estimated from the error in the digitized year values, since the true values are then known exactly. This translates into a mean square error in rainfall of about 0.1% of the station maximum, and a maximum error of about 0.3% of station maximum.

Source

Mostly from the MeteoSwiss website:

http://www.meteoswiss.admin.ch/home/climate/past/climate-extremes/extreme-value-analyses/standard-period.html?

NAO data from:

Hurrell, James & National Center for Atmospheric Research Staff (Eds). Last modified 16 Aug 2016. "The Climate Data Guide: Hurrell North Atlantic Oscillation (NAO) Index (station-based)."

https://climatedataguide.ucar.edu/climate-data/hurrell-north-atlantic-oscillation-nao-index-statio

Examples

```
require(gamair);require(mgcv)
data(swer)
## GEV model, over-simplified for speed...
system.time(b <- gam(list(exra~s(elevation)+ climate.region,
         ~s(elevation),~1),family=gevlss,data=swer))
plot(b,pages=1,scale=0,scheme=1)
```

`wesdr` *Diabetic retinopathy in Wisconsin*

Description

The data, originally from the `gss` package, record whether or not diabetic patients developed retinopathy along with three possible predictors.

Usage

```
data(wesdr)
```

Format

The `wesdr` data frame has the following columns

ret binary variable: 1 = retinopathy, 0 = not.

bmi Body mass index (weight in kg divided by square of height in metres)

gly Glycosylated hemoglobin - the percentage of hemoglobin bound to glucuse in the blood. This reflects long term average blood glucose levels: less than 6% is typical of non-diabetics, but is only rarely acheived by diabetic patients.

dur Duration of disease in years.

Details

Retinopathy is a common problem in diabetic patients and the interst is in predicting the risk using the measured predictors.

Source

Data are from Chong Gu's `gss` package.

Examples

```
require(gamair);require(mgcv)
data(wesdr)
## Smooth ANOVA model...
k <- 5
b <- gam(ret~s(dur,k=k)+s(gly,k=k)+s(bmi,k=k)+ti(dur,gly,k=k)+
         ti(dur,bmi,k=k)+ti(gly,bmi,k=k),select=TRUE,
         data=wesdr,family=binomial(),method="ML")
ow <- options(warn=-1) ## avoid complaint about zlim
plot(b,pages=1,scheme=1,zlim=c(-3,3))
options(ow)
```

`wine` *Bordeaux Wines*

Description

Data on prices and growing characteristics of 25 Bordeaux wines from 1952 to 1998.

Usage

```
data(wine)
```

Format

A data frame with 7 columns and 47 rows. The columns are:

year year of production

price average price of the wines as a percentage of the 1961 price.

h.rain mm of rain in the harvest month.

s.temp Average temperature (C) over the summer preceding harvest.

w.rain mm of rain in the winter preceding harvest.

h.temp average temperature (C) at harvest.

parker a rating of the wine quality (see source for details).

Source

`http://schwert.ssb.rochester.edu/a425/a425.htm`

References

Wood, S.N. (2006, 2017) Generalized Additive Models: An Introduction with R. CRC

Examples

```
data(wine)
pairs(wine[,-7])
```

Index