

Sperlich, Stefan; Zelinka, Jiří

**Working Paper**

## Generalized additive models

SFB 373 Discussion Paper, No. 2000,50

**Provided in Cooperation with:**

Collaborative Research Center 373: Quantification and Simulation of Economic Processes,  
Humboldt University Berlin

*Suggested Citation:* Sperlich, Stefan; Zelinka, Jiří (2000) : Generalized additive models, SFB 373 Discussion Paper, No. 2000,50, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin,  
<https://nbn-resolving.de/urn:nbn:de:kobv:11-10047765>

This Version is available at:

<https://hdl.handle.net/10419/62217>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*

# Generalized Additive Models


*Stefan Sperlich and Jiří Zelinka*

Let's assume that we have independent random variables  $T_1$  and  $T_2$  and the response variable  $Y$  having the form

$$Y = f_1(T_1) + f_2(T_2) + \varepsilon.$$

The functions  $f_1$ ,  $f_2$  are unknown and the random error  $\varepsilon$  is independent with  $T_1$  and  $T_2$ . This situation can be simulated using XploRe very well:

```
n    = 100
t    = normal(n,2)           ; explanatory variable
f1   = - sin(2*t[,1])        ; estimated functions
f2   = t[,2]^2
eps  = normal(n,1) * sqrt(0.75) ; error
y    = f1 + f2 +eps          ; response variable
```

 gam01.xpl

The data can come from praxis, too. Our task is to estimate the unknown functions  $f_1$  and  $f_2$ .

This chapter deals with such problems and their solutions. It ought to demonstrate and to explain how to use XploRe for nonparametric regression and data analysis in **generalized additive models** (GAM). It describes all quantlets which belong to the **gam** quantlib which contains all routines of XploRe provided for estimation and testing in generalized additive models. It also has several links to the **gplm** quantlib for **generalized partial linear models** (GPLM) in XploRe thus many quantlets which are used in **gam** are fully described in Chapter ?? but not mentioned here.

# 1 Brief Theory

## 1.1 Models

An additive model (AM) with response variable  $Y$  and explanatory variable  $T \in \mathbb{R}^d$  has the form

$$E(Y|T = t) = \sum_{j=1}^d f_j(t_j) + c,$$

where  $c$  is a constant with  $E(Y) = c$  and the univariate functions, also called **additive components**,  $f_j$  obey  $E_{T_j}\{f_j(T_j)\} = 0$  for all  $j$ .

Possible extensions which can be handled in XploRe are **additive partially linear models** (APLM), **generalized additive models** (GAM), a mixture of both (GAPLM), **models with bivariate additive components** and **additive models with interaction**.

Generalized additive models are of the form

$$E(Y|T = t) = G\left\{\sum_{j=1}^d f_j(t_j) + c\right\}$$

with a known link function  $G$ ,  $c := G^{-1}\{E(Y)\}$  and the same conditions on  $f_j$  as above.

Special cases of these models are the well known probit and logit regression models.

Additive partially linear models allow for an additional linear part to the original additive model. When  $T \in \mathbb{R}^d$  is the explanatory variable with an influence of unknown functional form and  $X \in \mathbb{R}^p$  an explanatory variable with linear influence, we have

$$E(Y|T = t, X = x) = \sum_{j=1}^d f_j(t_j) + c + x^T \beta,$$

where the additive components  $f_j$  and the parameter  $\beta$  have to be estimated. These models are especially recommended to include discrete and dummy variables into the model.

The mixture of both, that is the **generalized additive partially linear model**, has the form

$$E(Y|T = t, X = x) = G \left\{ \sum_{j=1}^d f_j(t_j) + c + x^T \beta \right\} .$$

Sometimes we know of joint influence of some explanatory variables, e.g.  $T_k$  and  $T_l$  and thus their influence cannot be separated into two additive components. In those cases, the sum of them,  $f_k(\cdot) + f_l(\cdot)$ , has to be replaced by a bivariate additive component  $f_{k,l}(\cdot, \cdot)$ .

A further possible extension is to keep the additive separable structure as introduced above but to allow additionally for any kind of interaction term  $f_{kl}$ . Do not mix them up with the bivariate additive components we just spoke about! Here we focus on the isolated marginal influences with condition  $E_{T_k} \{f_{kl}(T_k, t_l)\} \equiv E_{T_l} \{f_{kl}(t_k, T_l)\} \equiv 0$ . The model we consider then is

$$E(Y|T = t) = \sum_{j=1}^d f_j(T_j) + \sum_{1 \leq k < l \leq d} f_{kl}(T_k, T_l) + c$$

with  $E_{T_j} \{f_j(T_j)\} = 0$  as in the models before.

## 1.2 Marginal Integration

As indicated by its name, the marginal integration estimator is estimating the marginal influence of a particular explanatory variable on a multidimensional regression. If the true model is an additive separable one, the functional to be estimated is exactly the additive component. Possible interaction in the model will be neglected and the estimator is still giving the marginal and thus interpretable influence of the considered variable.

The basic idea of the estimation procedure is to estimate a pre-estimator of the hyperdimensional regression surface and then to integrate out the dimensions not of interest keeping the direction of interest fixed. Assuming we are interested in the first additive component at point  $t_1$  this would lead to the formula

$$\hat{f}_1(t_1) = \sum_{l=1}^n \tilde{m}(t_1, T_{-1,l}) ,$$

where  $\tilde{m}$  is a pre-estimator for  $E(Y|T)$ .  $T_{-1,l}$  denotes the  $l$ -th observation of the explanatory vector  $T \in \mathbb{R}^d$  without the first influence variable and  $\{Y_i, T_i\}_{i=1}^n$  the observations.

The pre-estimators are kinds of multidimensional kernel smoother in XploRe. For the pre-estimation in generalized additive partially linear models we make use of a variant of profile maximum likelihood.

The calculation of such an estimate would need  $O(n^3)$  steps, to apply this procedure on large data sets will take plenty of time. For those data sets an alternative fast procedure can be applied which asymptotically (for a large number of observations) yields the same results but needs only  $O(n^2)$  computing steps. In that procedure for the pilot estimation the usual local polynomial estimator is replaced by the fully internalized smoother (see Jones, Davies, and Park 1994).

Since we know the asymptotics for the integration estimator, different test procedures for component analysis can be constructed. The main idea is always to estimate either the function or its derivative under hypothesis as well as under alternative and to look on the squared integral of their difference. This corresponds to the Euclidian distance between hypothesis and alternative. If this distance is too large we reject the hypothesis.

For further details, especially how to estimate in extended models, we refer to the list of the literature.

### 1.3 Backfitting

The backfitting estimator is projecting the multidimensional regression problem into the space of additive models. It is always looking for that additive model which yields the best regression fit. If the true model is an additive separable one, the estimated functionals are just the additive components of the true model.

If the true model is not additive, one reason for the use of backfitting is its effect of dimension reduction in high dimensional regression problems. Even if the model assumption is false, this method often leads to a reasonable regression fit. But in that case the additive components must not be interpreted.

Since this algorithm is directly related to the whole regression it is not possible to estimate a single component separately. The implemented iterative procedure works as follows. Given starting values on the  $(n \times 1)$  vectors  $\mathbf{f}_j^0$  for

$j = 1, 2, \dots, d$ , update these vectors in the  $r$ -th step by

$$\mathbf{f}_k^r = S_k \left( \mathbf{y} - \sum_{j \neq k} \mathbf{f}_j^{r-1} \right)$$

until some tolerance is reached. Here,  $S_k$  is the one dimensional smoothing operator calculating a regression from  $T_k$  on  $\left( \mathbf{y} - \sum_{j \neq k} \mathbf{f}_j^{r-1} \right)$ .

In generalized additive models we have to work with a quasi likelihood function and so the Fisher scoring algorithm is applied as an outer iterative procedure before applying the above mentioned algorithm.

For a more detailed description of the backfitting procedure we refer to Hastie and Tibshirani (1990) or Opsomer and Ruppert (1997).

## 1.4 Orthogonal Series

Another well known method to estimate regression functions nonparametrically is to approximate them by an orthogonal function basis. You choose a basis of functions which describes e.g. the  $L_2$  space, fix the degree of fineness (usually depending on the number of observations and the dimensions of the particular sample) and finally estimate the unknown parameters of the resulting function to get an optimal fit for your sample.

Such a basis can be constructed by wavelets. Here, the fineness of the estimation is mainly determined by the chosen level degree. The estimated coefficients of the wavelets are asymptotically normally distributed.

Assuming an additive model for the considered regression problem, the additive components are simply formed by the wavelets which correspond to the same particular explanatory variable.

A testing procedure to examine the additive components is based on an analysis of the estimated coefficients.

For an introduction into wavelets and further information we recommend the Chapter 14 in Härdle, Kline, and Müller (2000) and the quantlib `twave` and `wavelet` in XploRe.

More details and theory can be found in Kaiser (1994) and Härdle, Sperlich, and Spokoiny (1997).

## 2 Data Preparation

All estimation quantlets in the XploRe quantlib `gam` expect at least two input parameters:

**t**  
 $n \times p$  matrix, continuous explanatory variables for nonparametric part

**y**  
 $n \times q$  vector (more exactly a matrix of  $q$  vectors), observed responses

For partially linear models additionally:

**x**  
 $n \times d$  matrix, explanatory variables for linear part (at least the discrete variables).

There should be no vector of ones concatenated to the data. A constant is contained automatically in the nonparametric estimate for  $m(t)$ .

Neither the matrix `x` nor the vector `y` should contain missing values (`NaN`) or infinite values (`Inf`, `-Inf`). Those should be identified by `isNumber` and removed by `paf` (or replaced by something reasonable) before using GAM quantlets.

## 3 Noninteractive Quantlets for Estimation

```
m = intest(t, y, h, g, loc{, opt})  
    estimates an additive model (AM)  
  
{m, b, const} = intestpl(x, t, y, h, g, loc{, opt})  
    estimates an additive partially linear model (APLM)  
  
{m, b, const} = backfit(t, y, h, loc, kern{, opt})  
    estimates an additive and additive partially linear model
```

```

m = gintest(code, t, y, h, g, loc{, opt})
    estimates a generalized additive model (GAM)

{m, b, bv, const} = gintestpl(code, x, t, y, h, g{, opt})
    estimates a generalized additive partially linear model (GAPLM)

m = intest2d(t, y, h, g, loc{, opt})
    estimates a bivariate marginal influence

{fh, c} = interact(t, y, h, g, loc, incl{, tg})
    estimates an additive model with interaction terms

m = fastint(t, y, h1, h2, loc{, tg})
    estimates an additive model using marginal integration

```

Here is the list of all quantlets. Their use is described in the following subsections.

### 3.1 Estimating an Additive Model

```

m = intest(t, y, h, g, loc{, opt})
    estimates an additive model (AM)

```

```

library("gam")
randomize(1234)
t      = uniform(50,2)*2-1
g1     = 2*t[,1]
g2     = t[,2]^2
g2     = g2 - mean(g2)
y      = g1 + g2 + normal(50,1) * sqrt(0.25)
h      = #(1.2, 1.0)
g      = #(1.4, 1.2)
loc    = 1
gest   = intest(t,y,h,g,loc)
gest
bild   = createdisplay(1,2)


```



```

dat11 = t[,1]~g1
dat12 = t[,1]~gest[,1]
dat21 = t[,2]~g2
dat22 = t[,2]~gest[,2]
setmaskp(dat12,4,4,8)
setmaskp(dat22,4,4,8)
show(bild,1,1,dat11,dat12)
show(bild,1,2,dat21,dat22)

```

 gam02.xpl

The quantlet `intest` provides a way to estimate the univariate additive functions and derivatives of a separable additive model using Nadaraya-Watson, local linear or local quadratic estimation.

Input parameters:

**h**

$p' \times 1$  bandwidth vector for the directions of interest (see remarks). It can be  $p' = p$ ,  $p' = pg$  or  $p' = 1$  for the same bandwidth in all directions.

**g**

$p \times 1$  bandwidth vector for the directions **not** of interest

**loc** scalar specifying the estimation procedure:

- 0 —Nadaraya-Watson (local constant)
- 1 —local linear
- 2 —local quadratic

Optional parameters (see Section 5):

**opt.tg**

$ng \times pg$  matrix, a grid for continuous part If **tg** is given, the nonparametric function will be computed on this grid

**opt.shf**

scalar (show-how-far). If it exists and is equal to one, an output is produced and it indicates how the iteration is going on (additive function/point of estimation/number of iteration).

Output value:

`m`

$n \times p' \cdot (loc + 1) \times q$  matrix containing the marginal integration estimates in the first  $p'$  columns, followed by the 1-st and 2-nd derivative if local linear or quadratic estimation is used

**Remarks:** The grid may have less dimensions ( $p'$ ) than the explanatory data. The estimation will then be run on the first  $p'$  directions of interest. Consequently, it is possible to specify the bandwidth vector  $h$  for the directions of interest only.

### 3.2 Estimating an Additive Partially Linear Model


`{m, b, const} = intestpl(x, t, y, h, g, loc[, opt])`  
estimates an additive partially linear model (APLM)

```
library("gam")
randomize(1345)
loc= 2
x = matrix(50,2)
t = uniform(50,2)*2-1
xh = uniform(50,2)
x[,1]= 3*(xh>=0.8)+2*((0.8>xh)&&(xh>=0.3))+(0.3>xh)
x[,2]= (xh>(1/3))
g1    = 2*t[,1]
g2    = (2*t[,2])^2
g2    = g2 -mean(g2)
m     = g1 + g2 + x*(0.2|-1.0)
y     = m + normal(50,1)*0.25
h     = #(1.4, 1.4)
g     = #(1.4, 1.4)
{m,b,const} = intestpl(x,t,y,h,g,loc)
b
const
bild =createdisplay(1,2)
dat11= t[,1]~g1
```

```

dat12= t[,1]~m[,1]
setmaskp(dat12,4,4,8)
show(bild,1,1,dat11,dat12)
dat21= t[,2]~g2
dat22= t[,2]~m[,2]
setmaskp(dat22,4,4,8)
show(bild,1,2,dat21,dat22)

```

 gam03.xpl

The quantlet `intestpl` estimates the univariate additive functions and its derivatives in an additive partially linear model (APLM) using local linear or local quadratic estimation.

Input parameters:

**h**  
 $p \times 1$  vector or a scalar, the bandwidth for the directions of interest

**g**  
 $p \times 1$  vector or a scalar, the bandwidth for the directions **not** of interest

**loc**  
 scalar indicating the estimation procedure:  
 1 —local linear  
 2 —local quadratic

Optional parameters (see Section 5):

**opt.tg**  
 $ng \times pg$  matrix, a grid for continuous part. If `tg` is given, the nonparametric function will be computed on this grid.

**opt.shf**  
 scalar, if `shf=1` then it is shown how the process is going on (default: `shf=0`)

Output values:

**m**  
 $ng \times p \cdot (loc + 1) \times q$  matrix, the marginal integration estimates in the first  $p$  columns, followed by the 1-st and 2-nd derivative, if local linear or quadratic is used

**b**

$d \times 1$  vector, the coefficients of the linear part

**const**

scalar, the constant in the additive partial linear model

### 3.3 Estimating Additive and Additive Partially Linear Model


```
{m, b, const} = backfit(t, y, h, loc, kern{, opt})  
  estimates an additive and additive partially linear model
```

```
library("gam")  
randomize(1)  
n   = 100  
t   = normal(n,2)           ; explanatory variable  
x   = normal(n,2)           ; the linear part  
f1  = - sin(2*t[,1])        ; estimated functions  
f2  = t[,2]^2  
eps = normal(n,1) * sqrt(0.75)  
y   = x[,1] - x[,2]/4 + f1 + f2 +eps      ; response variable  
h   = 0.5  
opt = gamopt("x",x,"shf",1)   ; the linear part is used  
;                               and the iterations will be shown  
{m,b,const} = backfit(t,y,h,0,"qua",opt)  
;  
b                               ; coefficients for the linear part  
;                               ([1, -1/4] were used)  
const                           ; estimation of the constant  
;  
pic = createdisplay(1,2)      ; preparing the graphical output  
d1  = t[,1]~m[,1]  
d2  = t[,2]~m[,2]  
setmaskp(d1,4,4,4,4)  
setmaskp(d2,4,4,4,4)  
m1  = mean(f1)  
m2  = mean(f2)
```

```

yy = y - x*b - const
x1 = t[,1]~(yy - m[,2])
x2 = t[,2]~(yy - m[,1])
setmaskp(x1,1,11,4)
setmaskp(x2,1,11,4)
setmaskl(d1,(sort(d1~(1:rows(d1)))[,3]))',4,1,1)
setmaskl(d2,(sort(d2~(1:rows(d2)))[,3]))',4,1,1)
show(pic,1,1,d1,x1,t[,1]~(f1-m1))
show(pic,1,2,d2,x2,t[,2]~(f2-m2))

```

 gam04.xpl

The quantlet `backfit` estimates the univariate additive functions and its derivatives in an additive (AM) or additive partially linear model (APLM) using the backfitting algorithm. It accepts only one-dimensional response variables `y`.

Input parameters:

**h**  
 $p \times 1$  vector or a scalar, the bandwidth

**loc**  
 scalar indicates the estimation procedure:

- 0 —Nadaraya-Watson (local constant)
- 1 —local linear
- 2 —local quadratic

**kern**  
 string indicates the kernel function

- "qua" quartic kernel
- "epa" Epanechnikov kernel
- "gau" Gaussian kernel

Optional parameters:

**opt.x**  
 $n \times d$  matrix, the explanatory variables for the linear part (at least the discrete variables)

`opt.shf`

shf=1 to show how the iteration is going on (default: shf=0)

`opt.miter`

scalar, the maximal number of iterations (default: miter=50)

`opt.cnv`

scalar, the convergence criterion (default: cnv=0.000001)

The quantlet returns

**m**

$n \times p \cdot (loc + 1)$  matrix, the estimate of the additive functions in column 1 to  $p$ , the first derivatives in column  $(p+1)$  to  $2p$  and the second derivatives in column  $(2p+1)$  to  $3p$

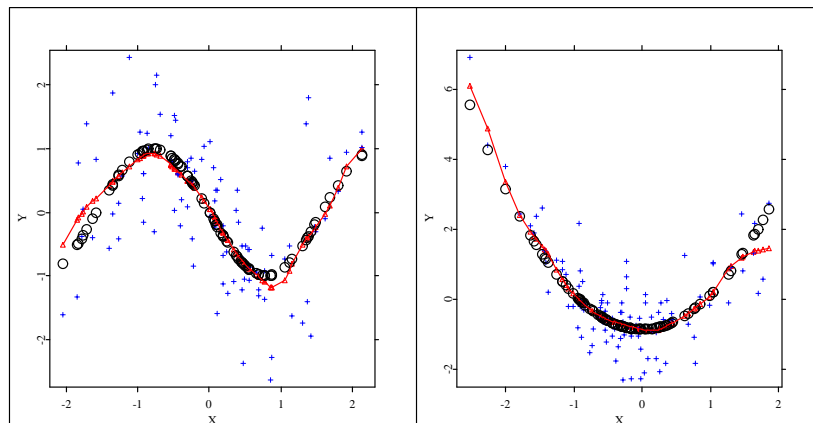
**b**

$d \times 1$  vector, the coefficients of the linear part

**const**

scalar, the estimate of the constant in the model

The example to this quantlet ([gam04.xpl](#)) produces the following graphical output:




It can be seen original data (crosses), exact values of the estimated functions (circles) and their estimations (small triangles connected by lines).

### 3.4 Estimating a Generalized Additive Model

```
m = gintest(code, t, y, h, g, loc{, opt})
      estimates a generalized additive model (GAM)
```

```
library("gam")
randomize(1235)
n      = 100
p      = 2
t      = uniform(n,p)*2-1
g1     = 2*t[,1]
g2     = t[,2]^2
g2     = g2 - mean(g2)
m      = g1 + g2
y      = cdfn(m) .> uniform(n)      ; probit model
h      = #(1.7, 1.5)
g      = #(1.7, 1.5)
tg     = grid(-0.8,0.1,19)
opt    = gamopt("tg",tg~tg,"shf",1)
loc    = 1
code   = "bipro"
m      = gintest(code,t,y,h,g,loc,opt)
d1     = tg[,1]~m[,1]
d2     = tg[,2]~m[,2]
setmaskp(d1,4,4,8)
setmaskp(d2,4,4,8)
bild   = createdisplay(1,2)
show(bild,1,1,d1,t[,1]~g1)
show(bild,1,2,d2,t[,2]~g2)
```

 gam05.xpl

The quantlet `gintest` estimates the univariate additive functions and its derivatives in a generalized additive model (GAM) using Nadaraya-Watson, local linear or local quadratic estimation.

Input parameters:

`code`

string, specifies the distribution of  $y$  and the link function. Currently implemented codes are:

"bilo" binomial with logistic link (logit)

"bipro" binomial with normal distribution link (probit)

"noid" normal with canonical (identity) link

`h`

$p' \times 1$  bandwidth vector for the directions of interest (see remarks). It can be  $p' = p$ ,  $p' = pg$  or  $p' = 1$  for the same bandwidth in all directions.

`g`

$p \times 1$  bandwidth vector for the directions **not** of interest

`loc`

scalar specifies the estimation procedure:

0 —Nadaraya-Watson (local constant)

1 —local linear

2 —local quadratic

Optional parameters:

`opt.tg`

$ng \times pg$  matrix, a grid for the continuous part (see remarks)

`opt.shf`

for shf=1 an indicator to show how the process is going on (default: shf=0)

The quantlet returns

`m`

$ng \times p' \cdot (loc + 1) \times q$ , the marginal integration estimates in the first  $p'$  columns, followed by the 1-st and 2-nd derivative, if local linear or quadratic estimation is used




**Remarks:** The grid may have less dimensions  $p'$  than the explanatory data. The estimation will then be run on the first  $p'$  directions of interest. Consequently, you need to specify the bandwidth vector  $h$  for the directions of interest only.

### 3.5 Estimating a Generalized Additive Partially Linear Model

```
{m, b, bv, const} = gintestpl(code, x, t, y, h, g{, opt})
  estimates a generalized additive partially linear model (GAPLM)
```

```
library("gam")
randomize(1235)
n      = 100
p      = 2
d      = 2
b      = 1|2
t      = uniform(n,p)*2-1
x      = 2.*uniform(n,d)-1
g1     = 2*t[,1]
g2     = t[,2]^2
g2     = g2 - mean(g2)
m      = g1 + g2
y      = cdfn(m+x*b) .> uniform(n)      ; probit model
h      = #(1.7, 1.5)
g      = #(1.7, 1.5)
tg     = grid(-0.8,0.1,18)
opt    = gamopt("tg",tg~tg)
opt    = gamopt("shf",1,opt)
code   = "bipro"
{m,b,bv,c} = gintestpl(code,x,t,y,h,g,opt)
gamout(t,y,m,b,c,gamopt("pl",1,"x",x,"bv",bv,opt))
```

 gam06.xpl

The quantlet `gintestpl` estimates the univariate additive functions in a generalized additive partially linear model (GAPLM) using Newton-Raphson or Fisher scoring algorithm.

Input parameters:

`code`

string specifying the distribution of  $y$  and the link function. It accepts only one-dimensional  $y$ . Currently implemented codes are:

**binomial**

"bilo" binomial with logistic link (logit)

"bipro" binomial with normal distribution link (probit)

"bicll" binomial with complementary log-log link

**normal**

"noid" normal with canonical=identity link

"nopow" normal with power (inverse) link

**gamma**

"gac1" gamma with canonical=reciprocal (inverse) link

"gapow" gamma poisson with power (inverse) link

**inverse gaussian**

"igc1" inverse gaussian with canonical=squared reciprocal (inverse) link

"igpow" inverse gaussian with power (inverse) link

**negative binomial**

"nbc1" negative binomial with canonical (inverse) link

"nbpow" negative binomial with power (inverse) link

`h`

$p \times 1$  bandwidth vector for the directions of interest

`g`

$p \times 1$  bandwidth vector for the directions **not** of interest

Optional parameters:

`opt.tg`

$np \times p$  matrix to estimate on a grid

`opt.shf`

if shf=1 then it is shown how the process is going on (default: shf=0)

`opt.b0`  
 $d \times 1$  vector to provide initial coefficients for the linear part (default: GLM pre-estimation)

`opt.nosort`  
 nosort=1 indicates that `t` is already sorted by its first column (default: nosort=0). Sorting is required by the algorithm, hence you should switch it off only when the data are already sorted.

`opt.miter`  
 maximal number of iterations (default: miter=10)

`opt.cnv`  
 scalar to determine the convergence criterion (default: cnv=0.0001)

`opt.fscor`  
 fscor=1 to switch to the Fisher-Scoring algorithm (default: Newton-Raphson). This parameter is ignored for canonical links.

`opt.wx`  
 scalar or  $n \times 1$  vector to make use of prior weights. For binomial models usually the binomial index vector (default: 1).

`opt.wt`  
 $n \times 1$  vector, weights for `t` (trimming factors) (default: all components set to 1)

`opt.wtc`  
 $n \times 1$  vector to apply weights for the convergence criterion, w.r.t.  $m(t)$  (default: wt is used)

`opt.off`  
 scalar or  $n \times 1$  vector, offset, can be used for constrained estimation (default: off=0)

`opt.pow`  
 scalar, power for power link (default: pow=0)

`opt.nbk`  
 scalar, extra parameter `k` for negative binomial distribution (default: nbk=1—geometric distribution)

The quantlet returns

**m**  
 $ng \times p$  matrix, the marginal integration estimates

**b**  
 $d \times 1$  vector, the coefficients of the linear part

**bv**  
 $d \times d$  covariance matrix for the estimated coefficients


**const**  
constant of the model

### 3.6 Estimating Bivariate Marginal Influence

```
m = intest2d(t, y, h, g, loc{, opt})  
  estimates a bivariate marginal influence
```

```
library("gam")  
randomize(12345)  
t      = grid( #(-0.9,-0.9), #(0.2,0.2), #(10,10))  
n      = rows(t)  
t      = t~(uniform(n)*2-1)  
g3     = sin(2*t[,3])  
g12    = t[,1] .* t[,2]^2  
y      = g3 + g12 + normal(n)*sqrt(0.5)  
h      = #(1.0, 1.0)  
g      = #(1.1, 1.1, 1.2)  
loc    = 1  
gest   = intest2d(t,y,h,g,loc)  
library("graphic")  
pic    = createdisplay(1,2)  
dat11  = grsurface(t[,1:2]~g12)  
dat12  = grsurface(t[,1:2]~gest[,1])  
gc     = grcube( dat11|dat12 )  
show(pic,1,1,dat11,gc.box,gc.x,gc.y,gc.z,gc.c)
```

```
show(pic,1,2,dat12,gc.box,gc.x,gc.y,gc.z,gc.c)
setheadline(pic, 1, 1, "Original function")
setheadline(pic, 1, 2, "Estimated function")
```

 gam07.xpl

The quantlet `intest2d` provides a way to estimate the bivariate marginal influence function of the explanatory variables  $t_1$  and  $t_2$ . You can choose the Nadaraya-Watson, the local linear or the local quadratic kernel smoother. Further, if local linear is chosen the program gives you the first derivative functions for both directions. If you choose the local quadratic smoother, you get the mixed derivative function. This quantlet can be used e.g. to explore the joint influence of two arbitrary explanatory variables in a multidimensional regression problem.

Input parameters:

**h**  
 scalar or  $2 \times 1$  vector, the bandwidth for the directions of interest

**g**  
 scalar or  $p \times 1$  vector, the bandwidth for the directions **not** of interest

**loc**  
 scalar specifying the estimation procedure:  
 0 —Nadaraya-Watson (local constant)  
 1 —local linear  
 2 —local quadratic

Optionally it is possible to use:

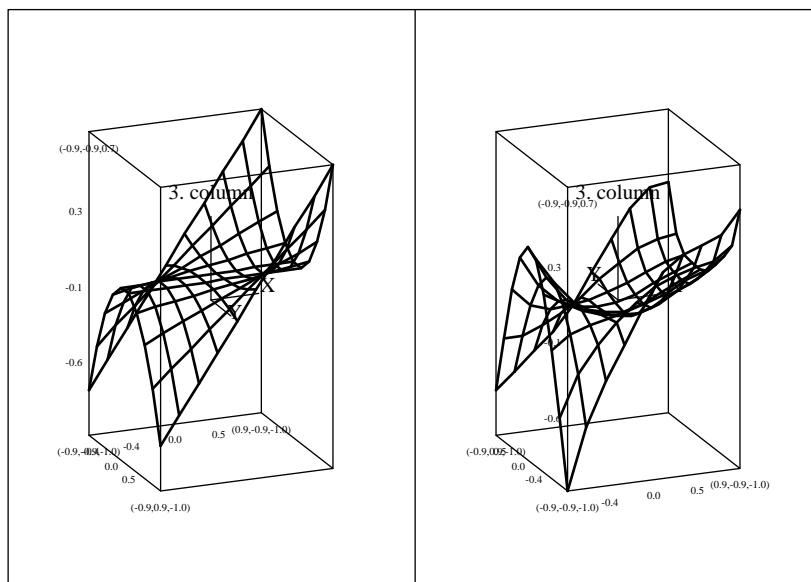
**opt.tg**  
 $ng \times 2$  matrix for estimating on a grid

**opt.shf**  
 shf=1 to show how the process is going on (default: shf=0)

The quantlet returns

**m**  
 $ng \times p' \times g$  matrix, the bivariate marginal integration estimate in the first column, the derivatives in the following columns

The example from this quantlet ([@gam07.xpl](#)) gives the following picture:



The original function is displayed on the left side, its estimate on the right side.

### 3.7 Estimating an Additive Model with Interaction Terms


```
{fh, c} = interact(t, y, h, g, loc, incl{, tg})
          estimates an additive model with interaction terms
```

```
library("gam")
randomize(12345)
t      = grid( #(-0.9,-0.9), #(0.2,0.2), #(10,10))
n      = rows(t)
t      = t~(uniform(n)*2-1)
g1     = 2*t[,1]
g2     = t[,2]^2 - mean(t[,2]^2)
```

```

g3      = sin(3*t[,3])
g12     = t[,1].*t[,2]
y       = g1+g2+g3+g12+normal(n)*sqrt(0.5)
h       = #(0.9, 0.9, 0.9)
g       = #(1.0, 1.0, 1.0)
incl    = 1~2
f       = interact(t,y,h,g,1,incl)
library("graphic")
pic     = createdisplay(2,2)
dat11   = sort(t[,2]~g2)
datf1   = sort(t[,2]~f.fh[,2])
dat12   = sort(t[,3]~g3)
datf2   = sort(t[,3]~f.fh[,3])
setmaskp(dat11,1,3,8)
setmaskp(dat12,1,3,8)
setmaskp(datf1,4,3,8)
setmaskp(datf2,4,3,8)
setmaskl(datf1,(1:rows(datf1))',4,1,1)
setmaskl(datf2,(1:rows(datf2))',4,1,1)
show(pic,1,1,dat11,datf1)
show(pic,1,2,dat12,datf2)
dat21   = grsurface(t[,1:2]~g12)
dat22   = grsurface(t[,1:2]~f.fh[,4])
gc      = grcube( dat21|dat22 )
show(pic,2,1,dat21,gc.box,gc.x,gc.y,gc.z,gc.c)
show(pic,2,2,dat22,gc.box,gc.x,gc.y,gc.z,gc.c)

```

 gam08.xpl

The quantlet `interact` estimates the univariate functions and the bivariate interaction terms wished by the user and the constant of the model, i.e., all functions  $f_j$  and  $f_{jk}$  of the model  $m = c + f_1 + \dots + f_d + f_{12} + \dots + f_{(d-1)d}$ , see also Subsection 1.2. Again the marginal integration estimator is used and you can choose between the Nadaraya-Watson, the local linear and the local quadratic smoother.

Input parameters:

**h**  
 scalar or  $p \times 1$  vector, the bandwidth for the directions of interest

**g**

scalar or  $p \times 1$  vector, the bandwidth for the directions **not** of interest

**loc**

scalar specifying the estimation procedure:

0 —Nadaraya-Watson (local constant)

1 —local linear

2 —local quadratic

**incl**

$pp \times 2$  matrix giving all pairs of indices  $j, k$  for which  $f_{jk}$  shall be included

Optional parameters:

**tg**

$ng \times p$  matrix to estimate on a grid (see remarks)


The quantlet returns

**fh**

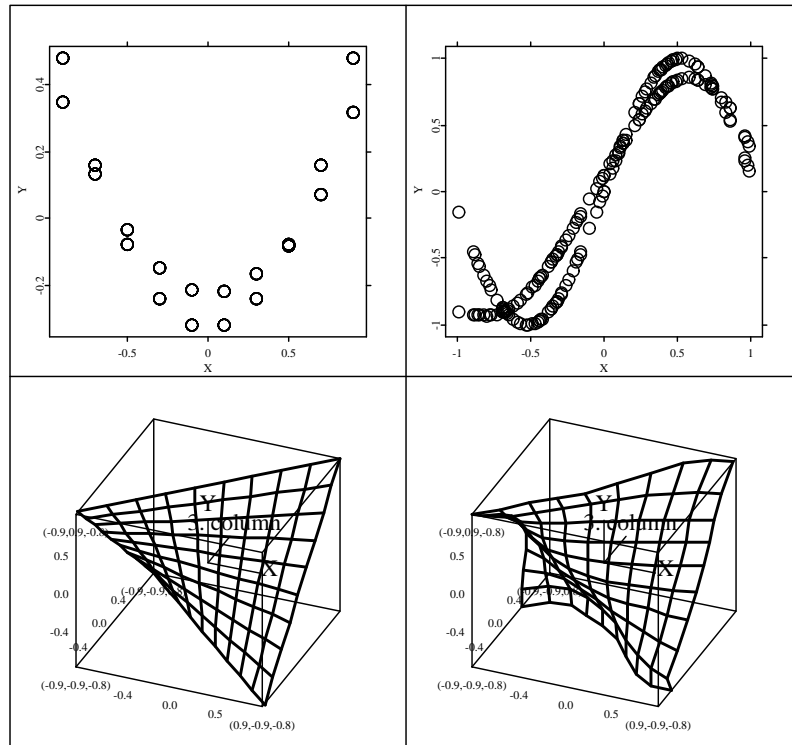
$ng \times (p + pp)$  matrix, the marginal integration estimates of the univariate functions and the chosen interaction terms

**c**

scalar, the constant of the model

The example  `gam08.xpl` gives the following picture:






You see displayed the second and third additive component in the upper plots, where the original functions are black and their estimates blue. In the lower plots are displayed the original interaction on the left and its estimate on the right.

**Remarks:** Note that `interact` accepts only one-dimensional `y`. If you choose a grid `tg`, the interaction functions can only be estimated up to a constant shift.

### 3.8 Estimating an Additive Model Using Marginal Integration

```
m = fastint(t, y, h1, h2, loc{, tg})  
      estimates an additive model using marginal integration
```

```
library("gam")  
randomize(1234)  
n = 100  
d = 2  
;  
      generate a correlated design:  
var = 1.0  
cov = 0.4 *(matrix(d,d)-unit(d)) + unit(d)*var  
{eval, evec} = eigsm(cov)  
t = normal(n,d)  
t = t*((evec.*sqrt(eval))'*evec')  
g1   = 2*t[,1]  
g2   = t[,2]^2 -mean(t[,2]^2)  
y    = g1 + g2 + normal(n,1) * sqrt(0.5)  
h1   = 0.5  
h2   = 0.7  
loc  = 0  
gest = fastint(t,y,h1,h2,loc)  
library("graphic")  
pic  = createdisplay(1,2)  
dat11 = t[,1]~g1  
dat12 = t[,1]~gest[,1]  
dat21 = t[,2]~g2  
dat22 = t[,2]~gest[,2]  
setmaskp(dat12,4,4,8)  
setmaskp(dat22,4,4,8)  
show(pic,1,2,dat11,dat12)  
show(pic,1,1,dat21,dat22)
```

 gam09.xpl

The quantlet `fastint` estimates the univariate additive components  $f_j$  if and only if the true model is of additive structure, i.e., the underlying model is  $m = c + f_1 + \dots + f_d$ . Here, the marginal integration estimator is applied

and followed by a one-step-backfit . For the backfit step you can choose between the Nadaraya-Watson, the local linear and the local quadratic smoother. Consequently you get estimates for the first or for the first and the second derivatives. For the integration step we use the fully internalized smoother, see Subsection 1.2.

This estimation procedure is very fast compared to the above mentioned integration procedures but we recommend to use it only if the number of observations is big compared to the number of covariates and if the true model is indeed additive. It accepts only higher-dimensional  $\mathbf{y}$  variables.

Input parameters:

**h1**  
 scalar or a  $p \times 1$  vector, the bandwidth for the pilot estimation,(marginal integration); it is recommended to undersmooth here

**h2**  
 scalar or a  $p \times 1$  vector, the bandwidth for the backfit step

**loc**  
 scalar specifying the estimation procedure:  
 0 —Nadaraya-Watson (local constant)  
 1 —local linear  
 2 —local quadratic

Optionally it is possible to use:

**tg**  
 $ng \times pg$  matrix for estimating on a grid


The quantlet returns

**m**  
 $ng \times (p + pp)$  matrix, the estimates of the univariate additive components and their derivatives on  $\mathbf{t}$  or  $\mathbf{tg}$ , respectively

## 4 Interactive Quantlet GAMFIT

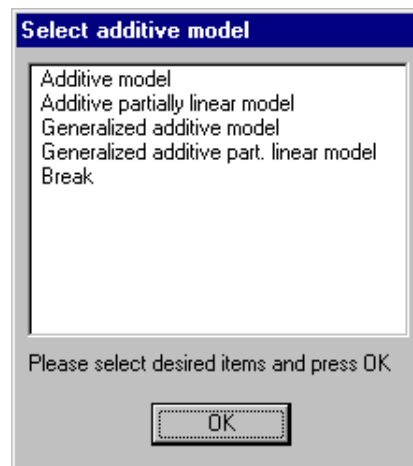
```
gamfit(t, y{, opt})  
    interactive tool for fitting of GAM models
```

```
library("gam")  
randomize(1234)  
t      = uniform(50,2)*2-1  
g1     = 2*t[,1]  
g2     = t[,2]^2  
g2     = g2 - mean(g2)  
y      = g1 + g2 + normal(50,1) * sqrt(0.25)  
gamfit(t,y)
```

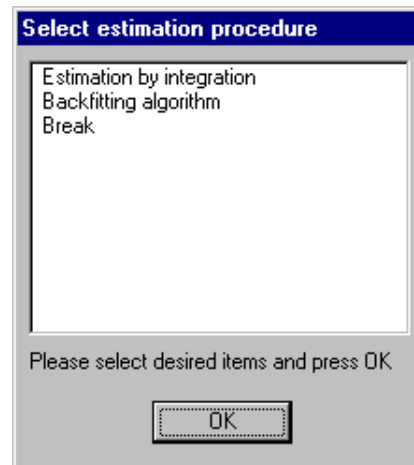
 gam10.xpl

`gamfit` provides a convenient interactive tool for the estimation of additive models.

The inputs `t` and `y` are obligatory parameters. All the other variables, selections and options needed for the estimation will be inquired interactively. The `Break` option ends the dialogue at any level without doing any calculations. `gamfit` starts with the model selection:

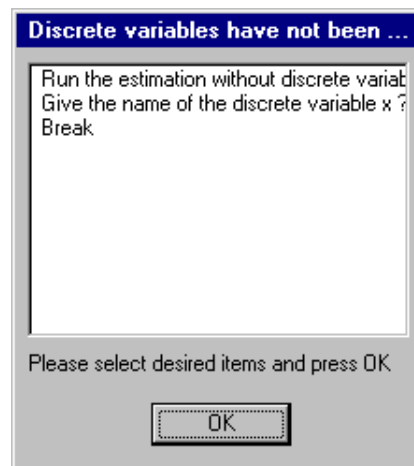


For the additive (AM) and the additive partially linear model (APLM) it is necessary to specify the estimation procedure. Since so far the backfitting procedure is not implemented for generalized additive models, the marginal integration estimator is automatically applied for GAM and GAPLM.

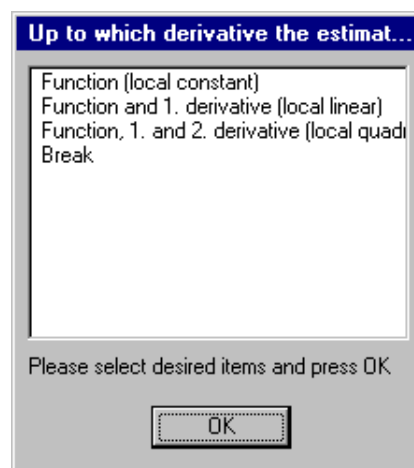


Depending on these selections `gamfit` checks the input and finds out, which further parameters are needed to run the estimation.

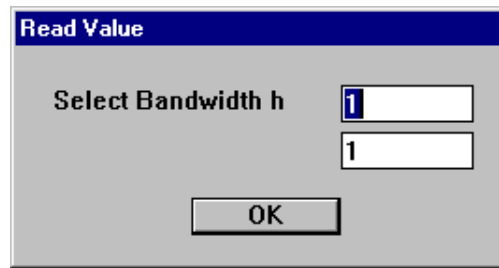
In case of a partially linear model you will be asked to quote the variable name for the linear part (named here the discrete variables). You may, alternatively, leave it out, which means to run the estimation on the linear part only and to switch back to an additive (AM) or a generalized additive model (GAM).



Most quantlets of the quantlib gam allow to use different estimation procedures: local constant, local linear or local quadratic. Hence, they are able to estimate the additive functions and its derivatives.

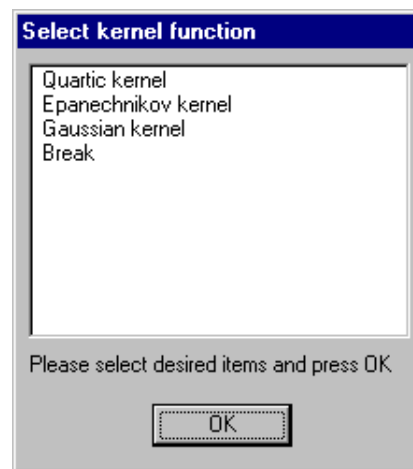


At least one vector of bandwidths is needed for all estimation procedures. Most of them additionally ask for a second bandwidth for the directions not of interest.



A dialog box titled "Read Value" with a blue header bar. It contains a label "Select Bandwidth h" followed by two input fields, both containing the number "1". Below the input fields is an "OK" button.

In case of the backfitting algorithm you need to specify the kernel function.



A dialog box titled "Select kernel function" with a blue header bar. It contains a list box with the following items: "Quartic kernel", "Epanechnikov kernel", "Gaussian kernel", and "Break". Below the list box is the text "Please select desired items and press OK". At the bottom is an "OK" button.

For generalized models you are asked to select the distribution of the dependent variable  $y$  and the link function.

Select exponential family for Y	Select a link function
Normal (Y : any real value) Binomial (Y : 0,1 or 0,...,m) Break	Logit (canonical) Probit Break
Please select desired items and press OK	Please select desired items and press OK
<input type="button" value="OK"/>	<input type="button" value="OK"/>

At last, before starting the estimation, it is possible to change the optional control parameters. It depends on the previous selections which of them appear (for detailed description of the options see Section 3). Some of them refer to the graphical output, others allow to exclude variables, to supply a grid, to produce an output with descriptive statistics and to display the steps of the estimation process.

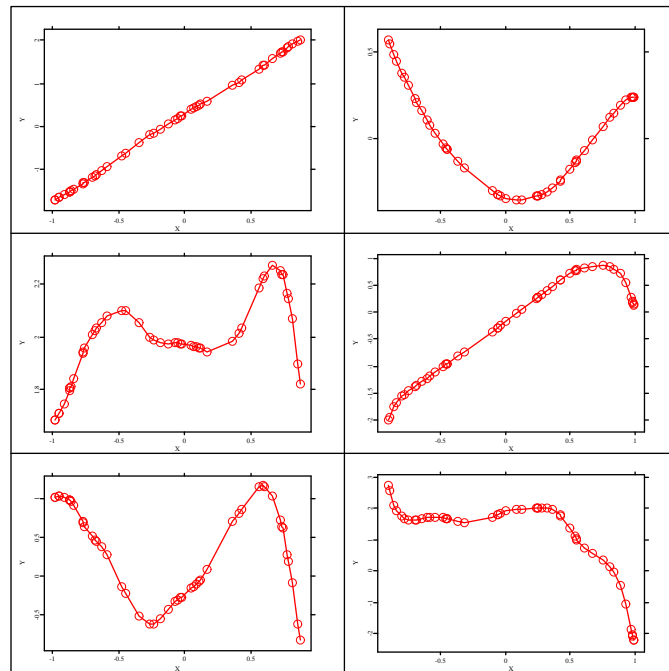
Select variables ? Name the variables ? Use a grid ? Show how the procedure is going on ? Don't show output graph ? Show descriptive statistics ? Name of output ? Title the output windows ? Break
Please select desired items and press OK
<input type="button" value="OK"/>

Subsequently a graphical output presents the estimation results for the nonlinear and should the occasion arise also for the linear part. If selected, descriptive



statistics are shown additionally.

The following pictures were acquired using the data from [gam10.xpl](#).



Descriptive Statistics					
n=50, 2 continuous variables					
	Minimum	Maximum	Mean	Median	Std.Error
y	-2.6209	2.2747	-0.28803	-0.23945	1.3253
t1	-0.98382	0.87715	-0.128	-0.12612	0.62649
t2	-0.89808	0.98741	0.059073	0.0816	0.62705

The resulting output is made globally available as a list object `gamfit`. It contains:

`gamfit.m`  
 $ng \times p$  matrix, the estimates for the nonlinear part

`gamfit.opt`  
internally used option list

and if given

`gamfit.b`  
 $d \times 1$  vector, the coefficients of the linear part

`gamfit.bv`  
 $d \times d$  covariance matrix for the estimated coefficients of (GAPLM)

`const`  
constant of the model

Although `gamfit` is an interactive tool, you are free to provide initially any additional option or parameter needed for the estimation of your model. The bandwidths `h` and `g`, the grid or the discrete variable `x` are typical examples. Look up the options corresponding to your model in Section 3. Section 5 gives instructions on optional parameters.

## 5 How to Append Optional Parameters

```
opt=gamopt(s0, v0{, s1, v1,...{, opt}})
  creates option list for gam quantlets
```

All quantlets in the XploRe quantlib `gam` expect optional parameters to be tailed onto the parameter list by means of a list object. The auxiliary quantlet `gamopt` presents a convenient tool to create the option list `opt`.

`s0,s1,...`  
string, names of the components to add. Allowed are:

"x"	discrete predictor variables
"loc"	indicator for local const, linear or quadratic estimation
"tg"	grid
"h"	bandwidth for the directions of interest
"g"	bandwidth for the directions <b>not</b> of interest
"code"	model code
"kern"	name of the kernel function
"wx"	weights
"off"	offset
"shf"	indicator to show iterations
"miter"	maximal number of iterations
"cnv"	convergence criterion
"fscor"	Fisher scoring will be used
"pow"	power for <code>gintestpl</code>
"nopic"	show no picture
"descript"	add descriptive statistics
"pl"	partially linear model
"name"	output variable name
"xvars"	discrete variable names
"tvars"	name of t-variable
"yvars"	name of y-variable
"title"	output picture title
"bv"	covariance matrix for b

`v0,v1,...`

the value of the corresponding component to add

`opt`

It is appropriate to use the old list of options as a parameter to save the previously defined ones.

For instance, calling

```
opt = gamopt("code","bipro","miter",10,"nopic",1,opt)
```

appends the optional parameters `code`, `miter` and `nopic` containing the values "bipro", 10 and 1 to the already existing option list `opt`.

Up to 10 optional parameters may be appended at one call. Rerun the quantlet to extend your option list.

Finally, check the list with the `names` command or by typing its name.

```
names(opt)
      [1,] code
      [2,] miter
      [3,] nopic

opt
      Content of object opt.code
      [1,] bipro
      Content of object opt.miter
      [1,] 10
      Content of object opt.nopic
      [1,] 1
```

The resulting option list may be used with different quantlets of the `gam` quantlib. Each quantlet picks out all the optional parameters needed, thus it is possible to use one option list for all `gam` quantlets.

Principally it is possible to define the list of optional parameters with the XploRe command `list`. However, this approach has some drawbacks. First, all elements of the option list would exist twice: as global objects as well as list components. Also, name conflicts could arise, since the components need to have specific names to be correctly identified by the `gam` quantlets. Finally, XploRe supports `list` with identical names for different list components. In this case, only the first of multiple elements with the same name can be identified.

Therefore, it is recommended to use the quantlet `gamopt` to set the options.

## 6 Noninteractive Quantlets for Testing

```
erg = wavetest(t, y, p{, dis, levels, data, adjust})
      component analysis in additive (partially linear) models

erg = intertest1(t, y, h, g{, opt, file})
      test for interaction
```


```
erg = intertest2(t, y, h, g{, opt, file})  
      test for interaction
```

With the following test procedures it is possible to do either component analysis in additive models, e.g. test for linearity or you can test for interaction and thus for additivity of a model.

## 6.1 Component Analysis in Additive (Partially Linear) Models

```
erg = wavetest(t, y, p{, dis, levels, data, adjust})  
      component analysis in additive (partially linear) models
```

```
library("gam")  
n    = 100  
randomize(1234)  
x    = normal(n,3)  
eps = normal(n,1) * sqrt(0.8)  
y    = sin(2*x[,1]) + x[,2]^2 + 2*x[,3] +eps  
p    = 1  
erg = wavetest(x,y,p)  
erg
```

 gam11.xpl

The quantlet `wavetest` provides a test procedure for component analysis in additive separable models. It is based on wavelets using the Haar basis. It consists of two separated tests, the local one is e.g. looking for jumps, the chi-square like one for the  $L_2$  distance. You can test whether the considered additive component is significantly different from a predetermined polynomial.

Input parameters:

**x**

$n \times p$  matrix, the observed explanatory variable, where the `dis` (see list of optional parameters) last columns are expected to be dummy variables

**y**  
 $n \times 1$  vector, the observed response variable

**p**  
 scalar, the degree of the polynomial of the hypothesis


Optionally it is possible to use:

**dis**  
 scalar, the number of dummy variables, if you have some included in **x**

**levels**  
 $(p - dis) \times 1$  vector, the wavelet levels for the components (default: maximal possible)

**data**  
 string, the name of the file in which the estimates shall be saved

**adjust**  
 $2 \times 1$  vector, multipliers to justify the first error probability, see help file

The quantlet returns a table displaying all information about the test results. The example of this quantlet ( `gam11.xpl`) produces the following output:

Contents of `erg`

```
[ 1,] " "
[ 2,] "-----"
[ 3,] "no output of function estimates"
[ 4,] "-----"
[ 5,] "HYPOTHESIS: 1.add.component is polynomial of degree 1"
[ 6,] "-----"
[ 7,] "Hypothesis has been rejected at level 0"
[ 8,] "-----"
[ 9,] "highest possible level is 3"
[10,] " "
[11,] "-----"
[12,] "-----"
[13,] "          local Test          chi-2 like Test "
[14,] "- - - - -"
```

```

[15,] "level  crit.value  test stat.      crit.value  test stat.  "
[16,] "-----"
[17,] "  0          2.74767      4.34697      3.24940      12.65449"
[18,] "-----"
[19,] "-----"
[20,] "  "

```

Here, the hypothesis of linearity has been rejected.

## 6.2 Testing for Interaction by `intertest1`

```


erg = intertest1(t, y, h, g{, opt, file})
      test for interaction

```

```

library("gam")
randomize(12345)
n      = 50
t      = uniform(n,3)*2-1
g1     = 2*t[,1]
g2     = t[,2]^2 - mean(t[,2]^2)
g3     = sin(3*t[,3])
g12    = t[,1].*t[,2]
y      = g1+g2+g3+g12+normal(n)*sqrt(0.5)
h      = #(0.9,0.9,0.7)
g      = #(1.0,1.0,0.9)
boot   = 99
hb     = 1.1
weight= matrix(n)-prod((abs(t[,1:2]).>0.85),2)
opt    = list(boot,hb,weight)
test   = intertest1(t,y,h,g,opt)
test

```

 [gam12.xpl](#)

The quantlet `intertest1` provides a test procedure to test the hypothesis that a predetermined interaction function is zero, i.e. the interaction is not existing. The assumed underlying model is  $m = c + f_1 + \dots + f_d + f_{12} + \dots + f_{(d-1)d}$ , see also 1.2 Marginal Integration. First, this procedure is estimating the interaction by the marginal integration estimator. Then the difference of this estimate to

the hypothesis is calculated. Since the procedure is based on bootstrap, the hypothesis model has to be determined by the user, in practice he has to decide which of the possible interaction terms have to be included.

Input parameters:

- t**  
 $n \times p$  matrix, the observed explanatory variable where the directions of interest have to be the first and the second column.
- y**  
 $n \times 1$  vector, the response variable.
- h**  
 $p \times 1$  bandwidth vector for the directions of interest.
- g**  
 $p \times 1$  bandwidth vector for the directions **not** of interest.

Optional parameters:

- opt**  
list:
  - opt.hyp**  
 $pp \times 2$  vector, all pairs of indices of which the interaction shall be included
  - opt.boot**  
number of bootstrap replications (default: boot=249)
  - opt.hb**  
scalar, the bandwidth multiplier for the bootstrap. When the test statistics are calculated we take **h\*hb** and **g\*hb** instead of **h** and **g** (default hb=1).
  - opt.weight**  
 $n \times 1$  vector, the weights for the test statistic (default: equal to 1 for all components)
- file**  
string, the name of the file to which the estimates will be saved if wished



The quantlet returns a table displaying all information about the test results. The example of this quantlet [gam12.xpl](#) (probably with a different seed for the function `randomize`) produces table with the following text:

Contents of test

```
[ 1,] " "
[ 2,] "- - - - -"
[ 3,] "no output of function estimates"
[ 4,] " "
[ 5,] "HYPOTHESIS: There is no interaction of x_1,x_2"
[ 6,] " "
[ 7,] "  looking at the interaction function estimate  "
[ 8,] "  Number of bootstrap replications:   99"
[ 9,] " "
[10,] "Hypothesis has not been rejected"
[11,] " "
[12,] "-----"
[13,] "-----"
[14,] " niveau   rejected      crit.value      test stat.  "
[15,] "- - - - -"
[16,] "      1          0          7.77893      0.96358"
[17,] "      5          0          6.00822      0.96358"
[18,] "     10          0          5.18759      0.96358"
[19,] "     15          0          4.37140      0.96358"
[20,] "     20          0          3.73898      0.96358"
[21,] "-----"
[22,] "-----"
[23,] " "
```

### 6.3 Testing for Interaction by `intertest2`


```
erg = intertest2(t, y, h, g{, opt, file})
      test for interaction
```

```
library("gam")
randomize(12345)
```

```

n      = 50
t      = uniform(n,3)*2-1
g1     = 2*t[,1]
g2     = t[,2]^2 - mean(t[,2]^2)
g3     = sin(3*t[,3])
g12    = t[,1].*t[,2]
y      = g1+g2+g3+g12+normal(n)*sqrt(0.5)
h      = #(1.1,1.0,0.9)
g      = #(1.2,1.2,1.1)
boot   = 99
hb     = 1.5
weight= matrix(n)-prod((abs(t[,1:2]).>0.85),2)
opt    = list(boot,hb,weight)
test   = intertest2(t,y,h,g,opt)
test

```

 gam13.xpl

The quantlet `intertest2` provides a test procedure to test the hypothesis that a predetermined interaction function is zero, i.e. the interaction is not existing. The assumed underlying model is  $m = c + f_1 + \dots + f_d + f_{12} + \dots + f_{(d-1)d}$ , see also Subsection 1.2. First, this procedure is estimating the mixed derivative of the interaction by the marginal integration estimator. Then the difference of this estimate to the hypothesis is calculated. Since the procedure is based on bootstrap, the hypothesis model has to be determined by the user, in practice he has to decide which of the possible interaction terms have to be included. For the bootstrap the user also can choose whether the model shall be estimated with a local linear or a local quadratic estimator.

Input parameters:

- t  
A  $n \times p$  matrix, the observed explanatory variable where the directions of interest have to be the first and the second column.
- y  
A  $n \times 1$  vector, the response variable.
- h  
A  $p \times 1$  bandwidth vector for the directions of interest.
- g  
A  $p \times 1$  bandwidth vector for the directions **not** of interest.

Optionally it is possible to use:

```
opt
  list:

  opt.hyp
     $pp \times 2$  vector, all pairs of indices of which the interaction shall be
    included

  opt.boot
    number of bootstrap replications (default: boot=249)

  opt.hb
    scalar, the bandwidth multiplier for the bootstrap. When the
    test statistics are calculated we take  $h*hb$  and  $g*hb$  instead of  $h$  and
     $g$  (default hb=1).

  opt.weight
     $n \times 1$  vector, the weights for the test statistic, (default: equal to 1
    for all components)

  opt.loc
    scalar, the degree of the local polynomial smoother (1=local linear,
    2=local quadratic)

file
  string, the name of the file to which the estimates will be saved if wished
```

The quantlet returns a table displaying all information about the test results. The example of this quantlet [@gam13.xpl](#) (probably with a different seed for the function randomize) produces the following output:

Contents of test

```
[ 1,] " "
[ 2,] "- - - - -"
[ 3,] "no output of function estimates"
[ 4,] " "
[ 5,] "HYPOTHESIS: There is no interaction of x_1,x_2"
[ 6,] " "
[ 7,] " testing for the mixed derivative  "
[ 8,] " Number of bootstrap replications:  99"
[ 9,] " "
```

```

[10,] "Hypothesis has not been rejected"
[11,] " "
[12,] "-----"
[13,] "-----"
[14,] " niveau   rejected      crit.value      test stat.  "
[15,] "-----"
[16,] "      1           0          48.26525        6.55640"
[17,] "      5           0          33.84235        6.55640"
[18,] "     10           0          32.34334        6.55640"
[19,] "     15           0          25.73994        6.55640"
[20,] "     20           0          20.97699        6.55640"
[21,] "-----"
[22,] "-----"
[23,] " "

```

## 7 Odds and Ends

### 7.1 Special Properties of GAM Quantlib Quantlets

This part of the chapter describes some features which may be interesting for special problems.

The `gam` quantlib automatically loads the quantlibs `xplore`, `glm` and `gplm`, if not yet active.


The quantlets `gintestpl` and `intestpl` perform the estimation on an internal grid if the number of observations exceeds 50 and 40, respectively. They interpolate the estimated additive functions for the explanatory variable `t`, or, if given, for the grid `tg`.

For graphical output `gamfit` makes use of the auxiliary quantlet `gamout`.

## 7.2 Estimation on Principal Component by PCAD

`{jhat, g, mhat} = pcad(x, xg, y, h, bn)`  
estimates the additive components, the significant directions and the regression on principal components

```
library("gam")
n = 100
v =uniform(n,4)
x =v[,2:4]
y =x[,1]^2+0.1*x[,2]+normal(n)
h =0.5
bn=0.02
gest=pcad(x,x,y,h,bn)
gest.jhat
gest.g
gest.mhat
```

 gam14.xpl

The quantlet `pcad` estimates the additive components, the significant directions and the regression function on principal components. It accepts only one-dimensional `y`. The standard call is:

Input parameters:

**x**  
 $n \times p$  design matrix

**xg**  
 $ng \times p$  matrix, the grid on which we will estimate

**y**  
 $n \times 1$  vector, the response variable

**h**  
 $p \times 1$  bandwidth vector

**bn**  
scalar, threshold for choosing significant directions

The quantlet returns

`jhat`

A  $q \times 1$  vector, the set of significant directions.

`g`

A  $ng \times q$  matrix, the function estimates of the significant directions.

`mhat`

A  $ng \times 1$  vector, the estimate of the regression using the significant directions.

## 8 Application for Real Data

We will demonstrate an example of processing of real data in this section. We can use two data sets of Wisconsin farm data, 1987, from originally 1000 data. Selected are middle sized animal farms, outliers were removed. The first data set `animal` contains 250 observations (rows) of **family labor**, **hired labor**, **miscellaneous inputs**, **animal inputs** and **intermediate run assets**. The response variable **livestock** is contained in the second data set `goods`. Detailed description of data, source, possible models of interest and some nonparametric analysis can be found in Sperlich (1998).

In this example we will deal with the first three inputs, i.e. family labor, hired labor, miscellaneous inputs and animal inputs. We will store them into the variable `t` and also we must read the response variable `y`:

```
data=read("animal.dat")
t1 = data[,1]
t2 = data[,2]
t3 = data[,3]
t4 = data[,4]
t=t1~t2~t3~t4
y=read("goods.dat")
```

Now we can calculate approximately bandwidth  $h$ :

```
h1=0.5*sqrt(cov(t1))
```

```

h2=0.5*sqrt(cov(t2))
h3=0.5*sqrt(cov(t3))
h4=0.5*sqrt(cov(t4))
h=h1|h2|h3|h4

```

Finally we set up the parameters for estimation and run the partial integration procedure `intest`. It will be shown running of the computation.

```

g=h
loc=0
opt=gamopt("shf",1)
m = intest(t,y,h,g,loc,opt)

```

For an objective view of the results we create the graphical output on Figure 1. It is produced by the following statements:

```

const=mean(y)*0.25
m1 = t[,1]^(m[,1]+const)
m2 = t[,2]^(m[,2]+const)
m3 = t[,3]^(m[,3]+const)
m4 = t[,4]^(m[,4]+const)
setmaskp(m1,4,4,4)
setmaskp(m2,4,4,4)
setmaskp(m3,4,4,4)
setmaskp(m4,4,4,4)
setmaskl(m1,(sort(m1^(1:rows(m1))))[,3])',4,1,1)
setmaskl(m2,(sort(m2^(1:rows(m2))))[,3])',4,1,1)
setmaskl(m3,(sort(m3^(1:rows(m3))))[,3])',4,1,1)
setmaskl(m4,(sort(m4^(1:rows(m4))))[,3])',4,1,1)
yy=y-mean(y)-sum(m,2)
d1=t[,1]^(yy+m[,1])
d2=t[,2]^(yy+m[,2])
d3=t[,3]^(yy+m[,3])
d4=t[,4]^(yy+m[,4])
setmaskp(d1,1,11,4)
setmaskp(d2,1,11,4)
setmaskp(d3,1,11,4)
setmaskp(d4,1,11,4)
pic = createdisplay(2,2)
show(pic,1,1,m1,d1)

```

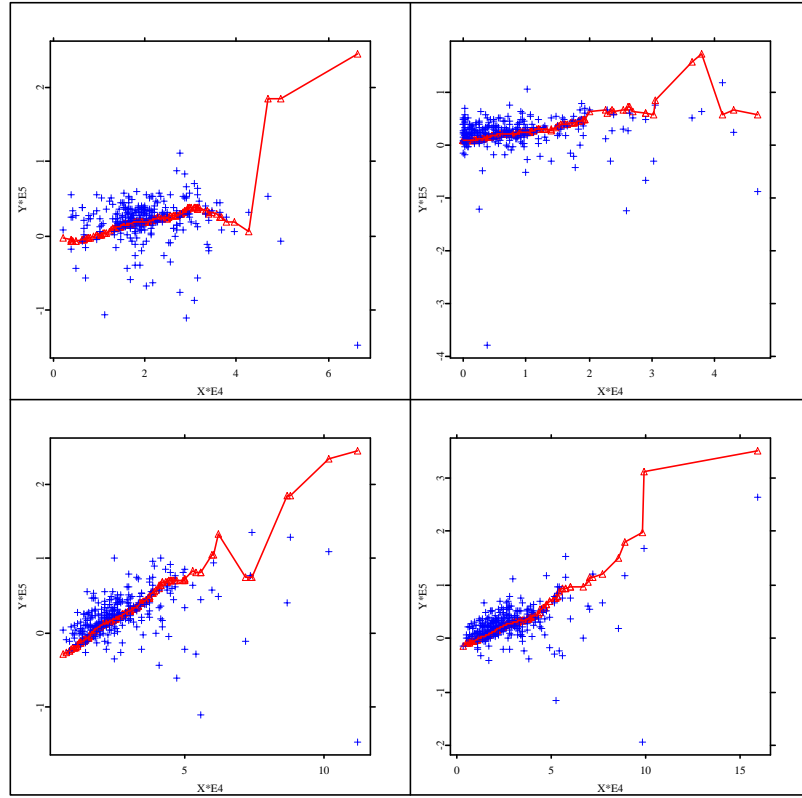



Figure 1: Generalized additive model for `animal`, partial integration.



```
show(pic,1,2,m2,d2)
show(pic,2,1,m3,d3)
show(pic,2,2,m4,d4)
```

 gam15.xpl

We see two properties of the data from the produced Figure 1:

1. the bandwidth  $h$  was chosen quite well; the data seems not to be over-smoothed or undersmoothed.
2. there are several outliers in the data; they can be seen in the right part of the pictures.

If we try to use quantlet `intest` with inner grid for computation (optional variable `opt.tg`) the quantlet ends with an error message. It is because of outliers where the data is too sporadic.

For better understanding the data we can use backfitting algorithm for estimation (quantlet `backfit`) and compare the results.

```
kern="qua"
{mb,b,const} = backfit(t,y,h,loc,kern,opt)
```


For graphical output we can use the similar approach as above with several differences.

```
m1 = t[,1]~mb[,1]
m2 = t[,2]~mb[,2]
m3 = t[,3]~mb[,3]
m4 = t[,4]~mb[,4]
setmaskp(m1,4,4,4)
setmaskp(m2,4,4,4)
setmaskp(m3,4,4,4)
setmaskp(m4,4,4,4)
setmaskl(m1,(sort(m1^(1:rows(m1))))[,3])',4,1,1)
setmaskl(m2,(sort(m2^(1:rows(m2))))[,3])',4,1,1)
setmaskl(m3,(sort(m3^(1:rows(m3))))[,3])',4,1,1)
setmaskl(m4,(sort(m4^(1:rows(m4))))[,3])',4,1,1)
yy=y-const-sum(mb,2)
d1=t[,1]~(yy+mb[,1])
```

```

d2=t[,2]~(yy+mb[,2])
d3=t[,3]~(yy+mb[,3])
d4=t[,4]~(yy+mb[,4])
setmaskp(d1,1,11,4)
setmaskp(d2,1,11,4)
setmaskp(d3,1,11,4)
setmaskp(d4,1,11,4)
pic2 = createdisplay(2,2)
show(pic2,1,1,m1,d1)
show(pic2,1,2,m2,d2)
show(pic2,2,1,m3,d3)
show(pic2,2,2,m4,d4)

```

 gam15.xpl

The graphs of this estimation on Figure 2 are like the graphs on Figure 1 achieved using `intest`; only different scale factor was used. It seems that the dependence of variable  $y$  on the miscellaneous inputs is almost linear. Unfortunately the quantlet `intestpl` for additive partially linear model ends with the error because of outliers. Likewise the testing of interactions (`intertest1` or `intertest1`) is aborting. For data manipulation using this quantlets the removing outliers from the data sets would be necessary.

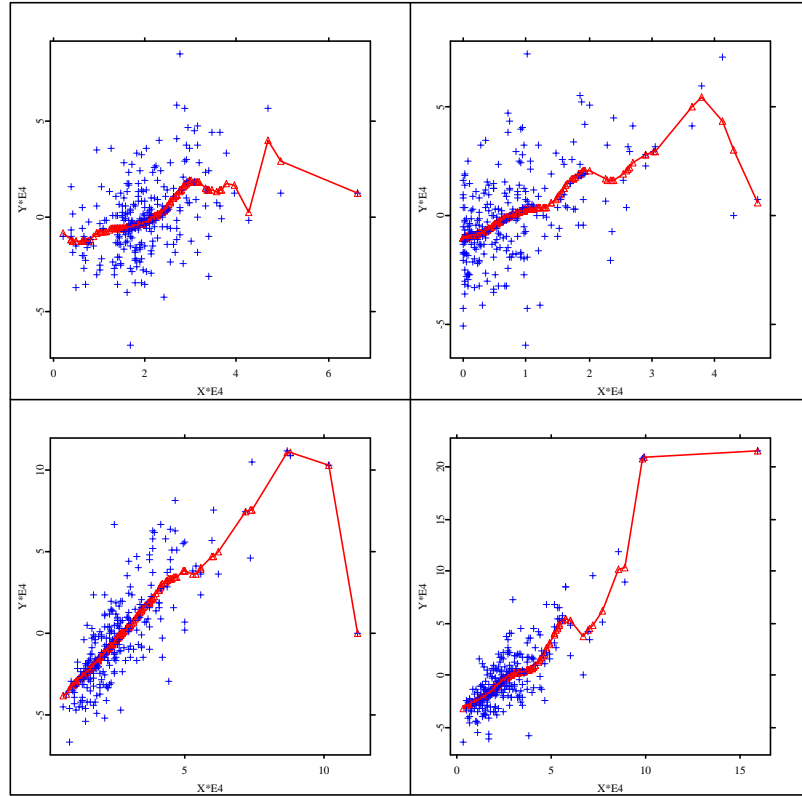


Figure 2: Generalized additive model for `animal`, backfitting.

## References

- Fan, J., Härdle, W. and Mammen, E. (1998). Direct estimation of low dimensional components in additive models, *Annals of Statistics* **26**: 943–971.
- Härdle, W., Huet, S., Mammen, E., and Sperlich S. (1998). Semiparametric additive indices for binary response models, *Discussion Paper, SFB 373*, Humboldt-Universität Berlin, Germany.
- Härdle, W., Klink, S., and Müller, M. (2000). *XploRe Learning Guide*, Springer.
- Härdle, W., Sperlich, S., and Spokoiny, V. (1997). Component analysis for additive models, *Discussion Paper, SFB 373*, Humboldt-Universität Berlin, Germany.
- Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*, Chapman and Hall, London.
- Jones, M. C., Davies, S. J. and Park, B. U. (1994). Versions of kernel-type regression estimators, *J. Amer. Statist. Assoc.* **89**: 825–832.
- Kaiser G. (1994). *A friendly guide to wavelets*, Birkenhäuser, Boston.
- Kim, W., Linton, O. B., and Hengartner, N. (1997). *A Nimble Method of Estimating Additive Nonparametric Regression*, Manuscript, Yale, US.
- Linton, O. B. and Härdle, W. (1996). Estimation of additive regression models with known links, *Biometrika* **83**: 529–540.
- Opsomer, J. D. and Ruppert, D. (1997). Fitting a bivariate additive model by local polynomial regression, *Annals of Statistics* **25**: 186–211.
- Severance-Lossin, E. and Sperlich, S. (1997). Estimation of Derivatives for Additive Separable Models, *Discussion Paper 30, SFB 373*, Humboldt-Universität Berlin, Germany.
- Sperlich S. (1998). Additive Modelling and Testing Model Specification, *Reihe Volkswirtschaft*, Shaker Verlag, Aachen, Germany.
- Sperlich S., Tjøstheim, D., and Yang, L. (1998). Nonparametric Estimation and Testing of Interaction in Additive Models, *Discussion Paper, SFB 373*, Humboldt-Universität Berlin, Germany.