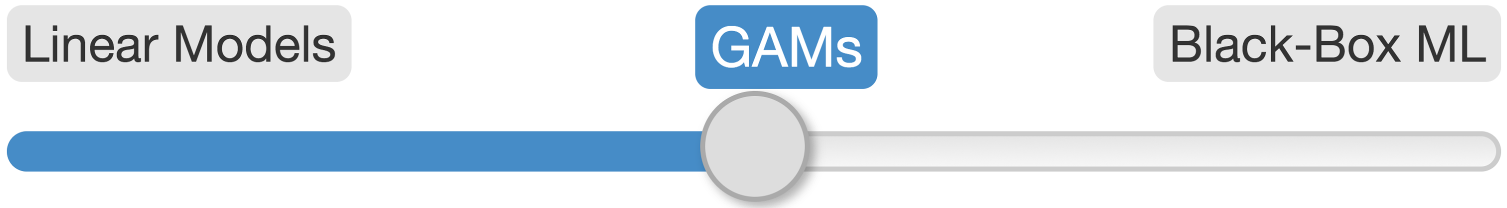
Nonlinear Modeling II

Generalized Additive Models (close to chapter 7, ISLR)

M Loecher

Contents

Trade-offs in Model Building



Pros and Cons of GAMs

+ GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we do not need to manually try out many different transformations on each variable individually.

+ The non-linear fits can potentially make more accurate predictions for the response Y .

+ Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMs provide a useful representation.

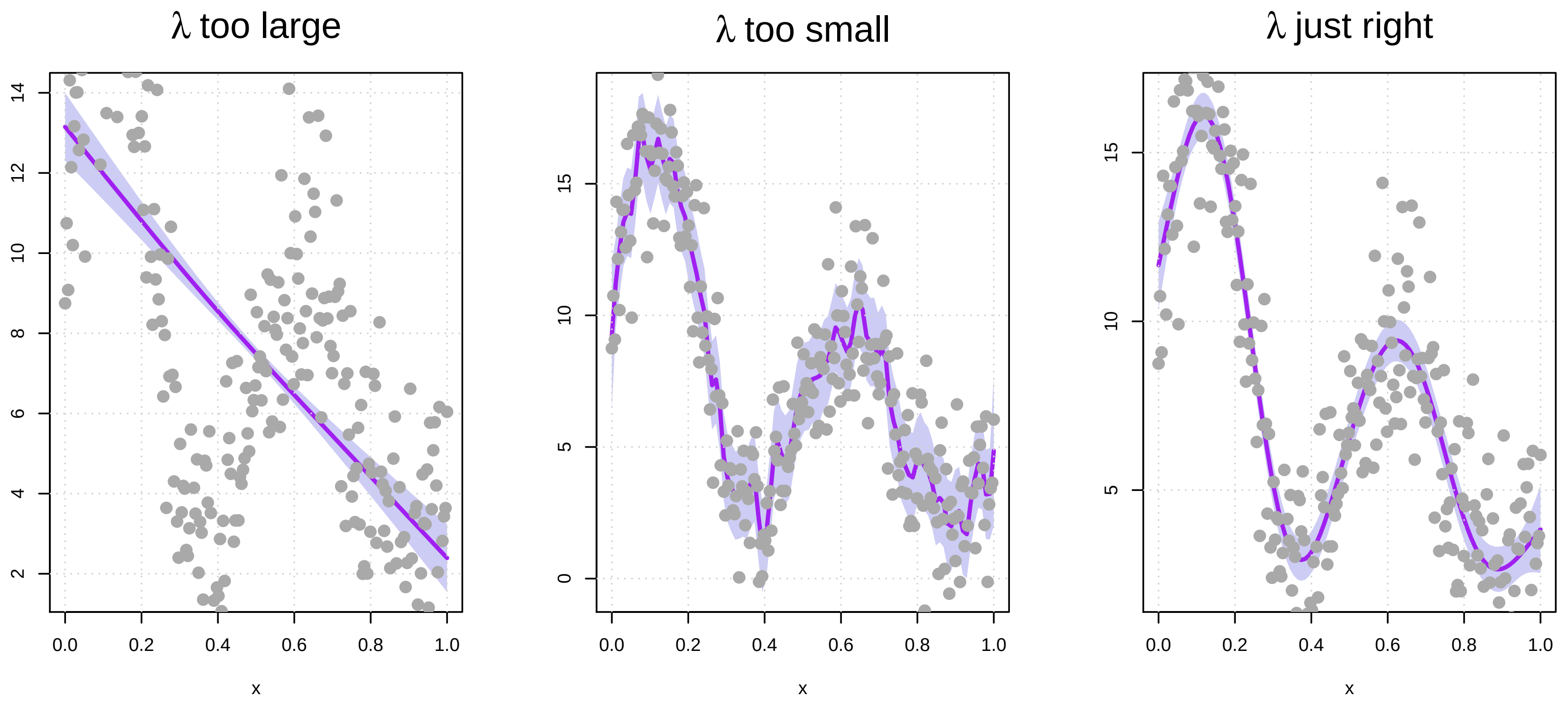
+ The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom.

- The main limitation of GAMs is that the model is restricted to be additive. With many variables, important interactions can be missed. However, as with linear regression, we can manually add interaction terms to the GAM model by including additional predictors of the form $X_j\ddot{\text{O}}X_k$. In addition we can add low-dimensional interaction functions of the form $f_{jk}(X_j, X_k)$ into the model; such terms can be fit using two-dimensional smoothers such as local regression, or two-dimensional splines (not covered here).

Syntax I

```
fit = gam(y~s(x,sp=0.1))
```

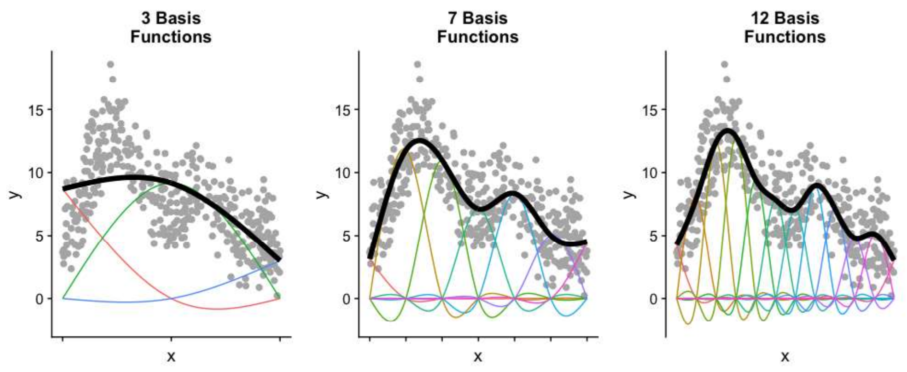
Choosing the Right Smoothing Parameter (Remember Goldilocks and the three bears?)



Syntax II

```
fit = gam(y~s(x,k=4))
```

Number of basis functions



GAM Exercises part I

Multiple Smooths

A true understanding of the label *additive* only comes in the context of multiple variables. A GAM is an extension of the multiple linear regression model which allows for non-linear re-

lationships between each feature and the response. We simply replace each linear component $\beta_j x_{ij}$ with a (smooth) nonlinear function $f_j(x_{ij})$ so that the model becomes:

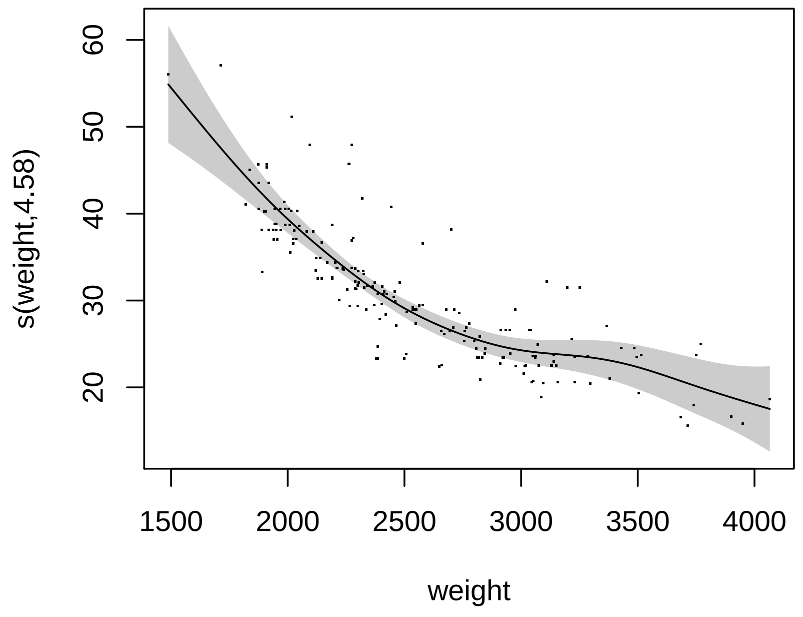
$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) \ldots + f_p(x_{ip}) + \epsilon_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

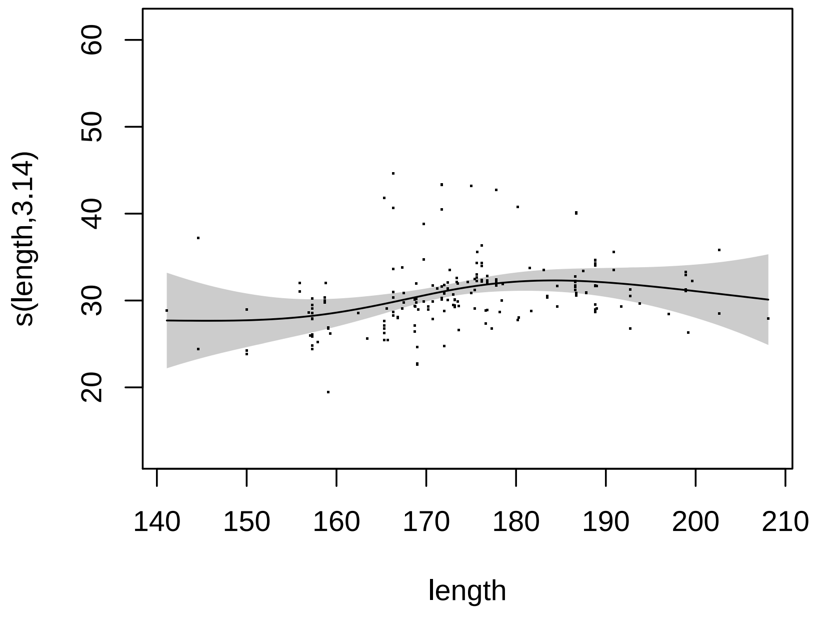
Our Working Dataset: mpg

make	fuel	aspir	doors	style	drive	eng.loc	wb	length	width	height	weight	eng.type	cylinders	eng.cc	fuel.sys	bore	stroke
alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68
alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68
alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47
audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.40
audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.40
audi	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	five	136	mpfi	3.19	3.40

Syntax III

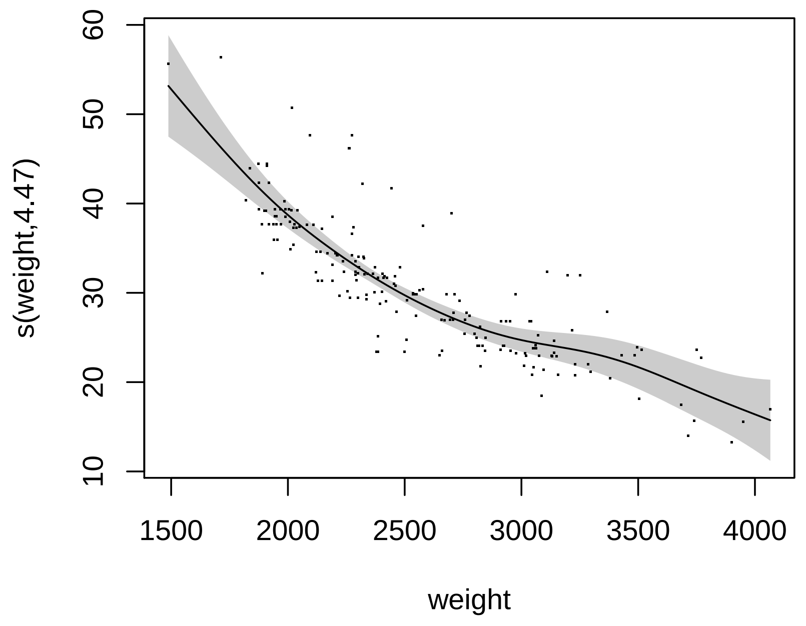
```r
mod2 <- gam(hw.mpg ~ s(weight) + s(length) , data = mpg, method = "REML")
```

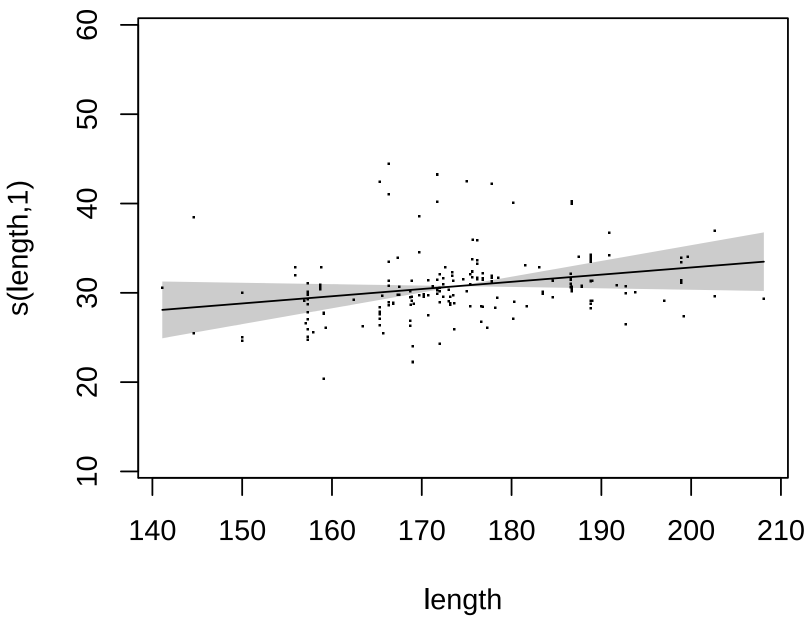




Linear Terms I

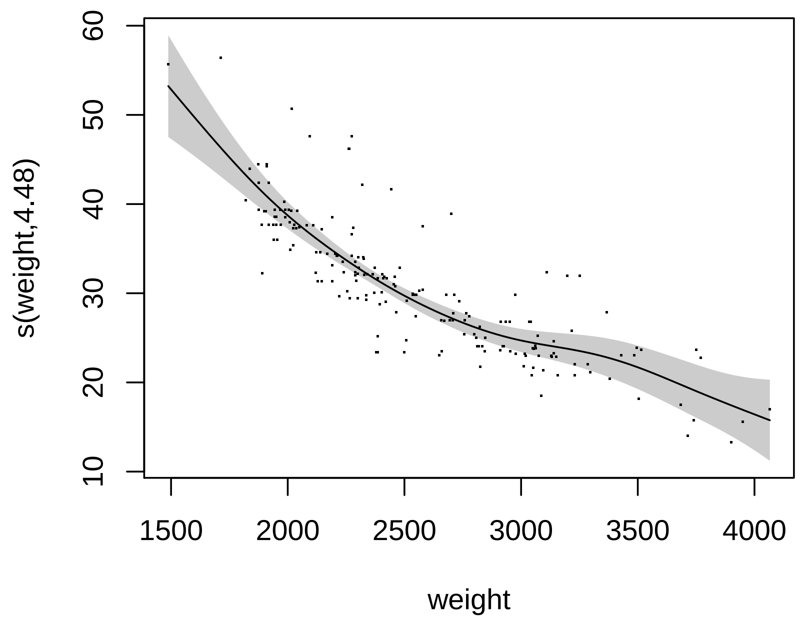
```r
mod3 <- gam(hw.mpg ~ s(weight) + length , data = mpg, method = "REML")
```

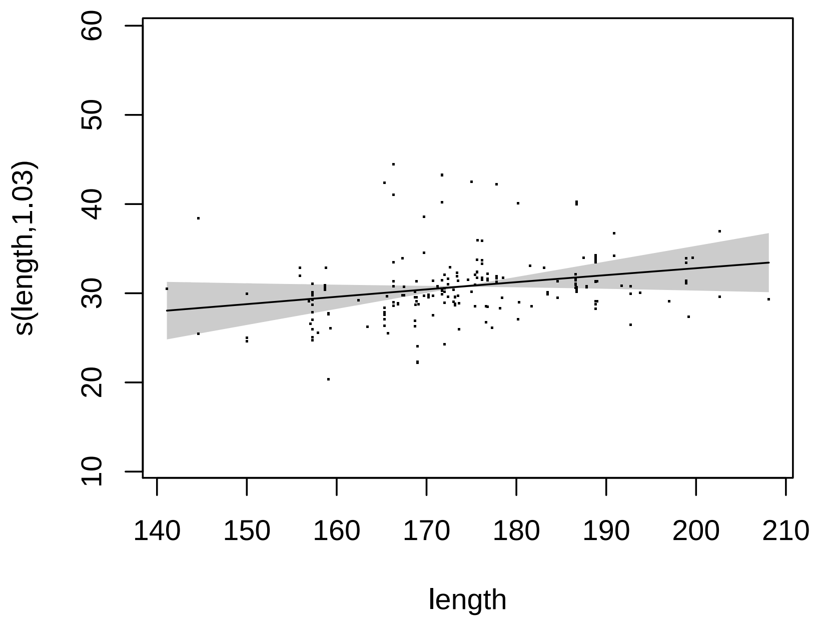




### Linear Terms II

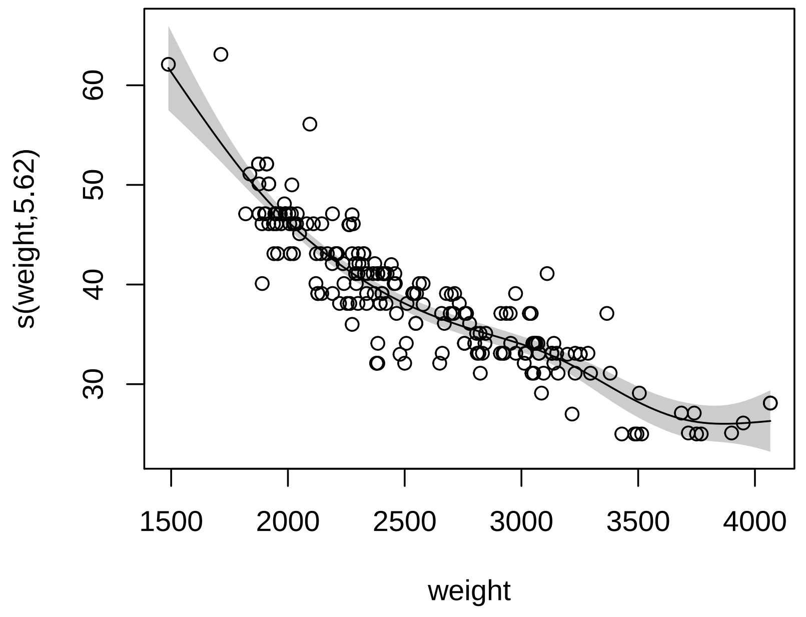
```
mod4 <- gam(hw.mpg ~ s(weight) + s(length,sp=100) , data = mpg, method = "REML")
```

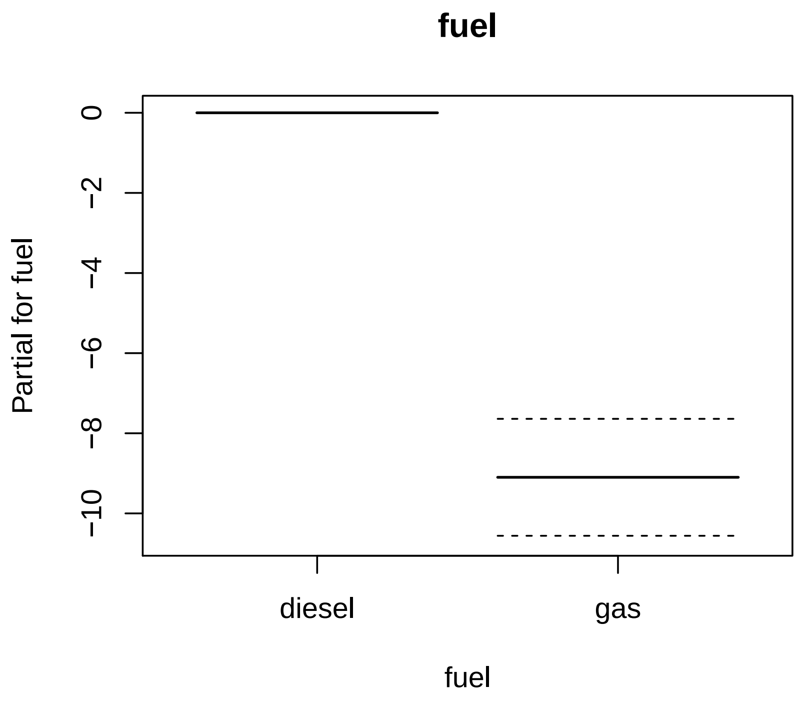




### Categorical Terms I

```
mod5 <- gam(hw.mpg ~ s(weight) + fuel , data = mpg, method = "REML")
```



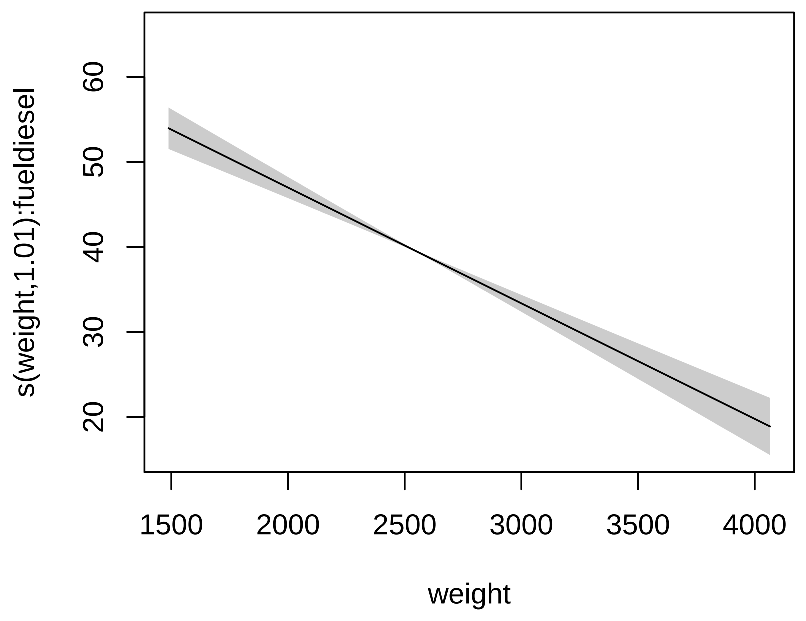


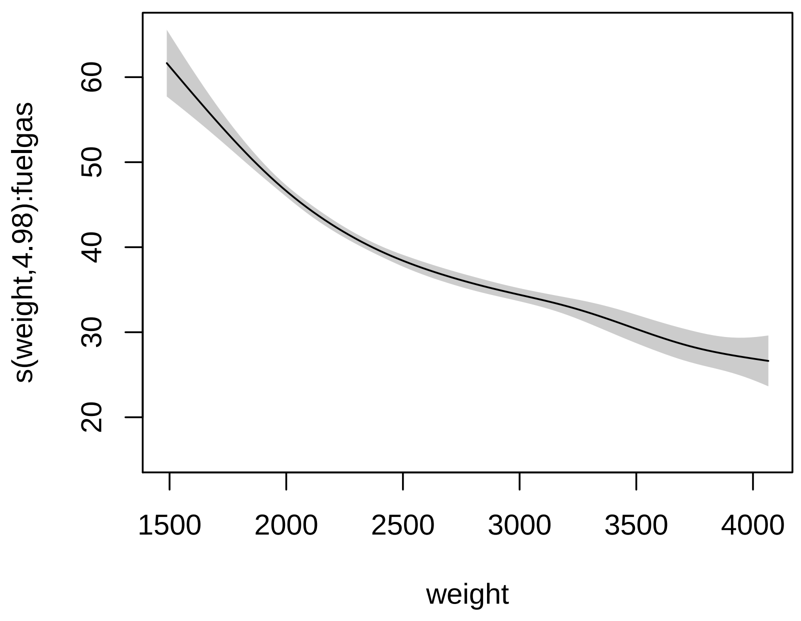
In this model the nonlinear effects of weight is the same for both fuel types. (You coulod view this as the analogy of the fixed slope, different intercepts in linear models!)

Categorical Terms II

We can also specify different smooths for different levels of the categorical variable! ("a factor smooth interaction")

```r
mod6 <- gam(hw.mpg ~ s(weight, by= fuel) + fuel, data = mpg, method = "REML")
```

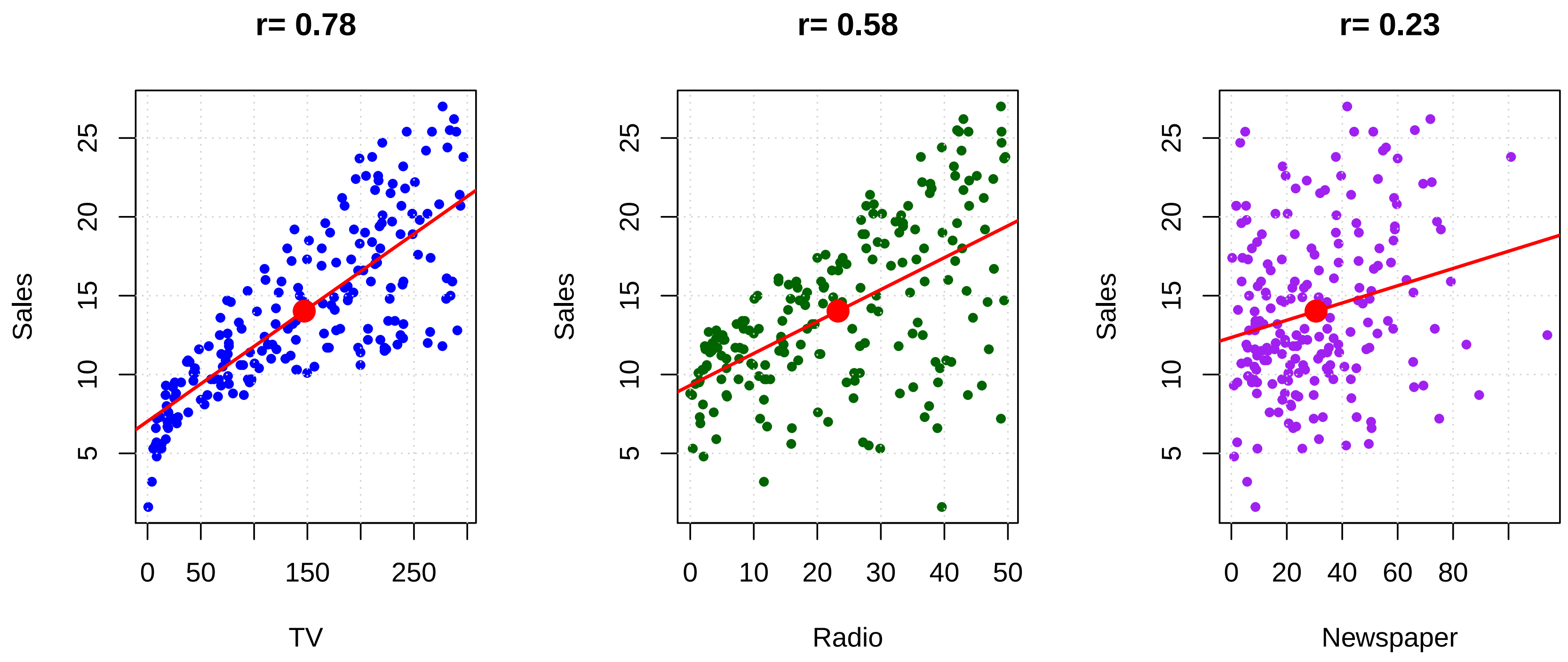




Interactions

Remember the Advertising Data from the ISLR book:

```r
Advertising <- read.csv("data/Advertising.csv")
#get rid of the first column
Advertising <- Advertising[,-1]
```



Classification

Let us make a binary feature in the Advertising data:

```
Advertising$HighSales = Advertising$Sales > median(Advertising$Sales)
Advertising$HighSales = factor(Advertising$HighSales)
```

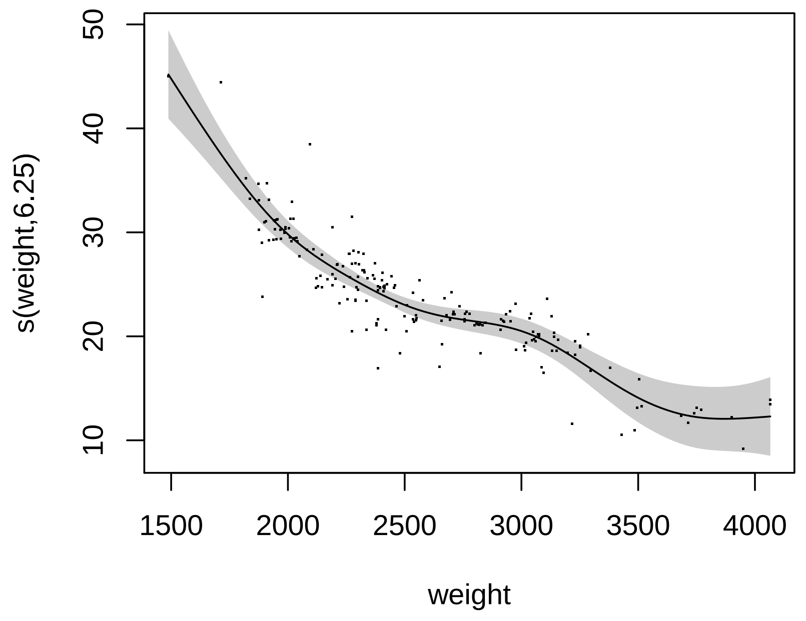
GAM Exercises part II

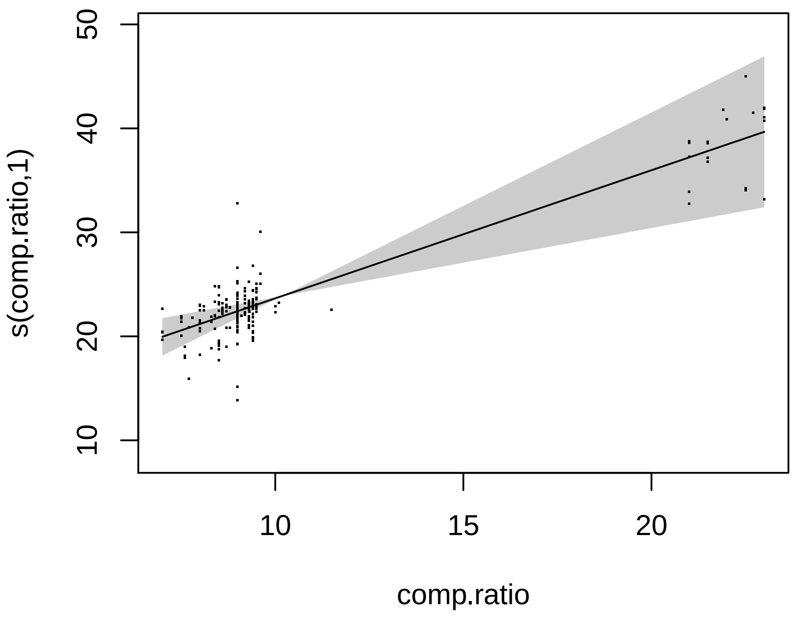
Interpreting GAM outputs

```
mod_city4 <- gam(hw.mpg ~ s(weight) +  s(price) + s(rpm) + s(comp.ratio) + s(wid
                 + fuel , data = mpg, method = "REML")
summary(mod_city4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## hw.mpg ~ s(weight) + s(price) + s(rpm) + s(comp.ratio) + s(width) +
##     fuel
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   23.873      3.531   6.760 1.89e-10 ***
## fuelgas        7.571      3.922   1.931   0.0551 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                  edf Ref.df      F  p-value
## s(weight)      6.254  7.439 20.909  < 2e-16 ***
## s(price)       2.681  3.421  1.678    0.155
## s(rpm)         7.499  8.285  8.534 2.07e-09 ***
## s(comp.ratio) 1.000  1.001 18.923 2.22e-05 ***
## s(width)       1.001  1.001  0.357    0.551
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.89   Deviance explained = 90.1%
## -REML = 464.81  Scale est. = 5.171     n = 199
```

Effective Degrees of Freedom

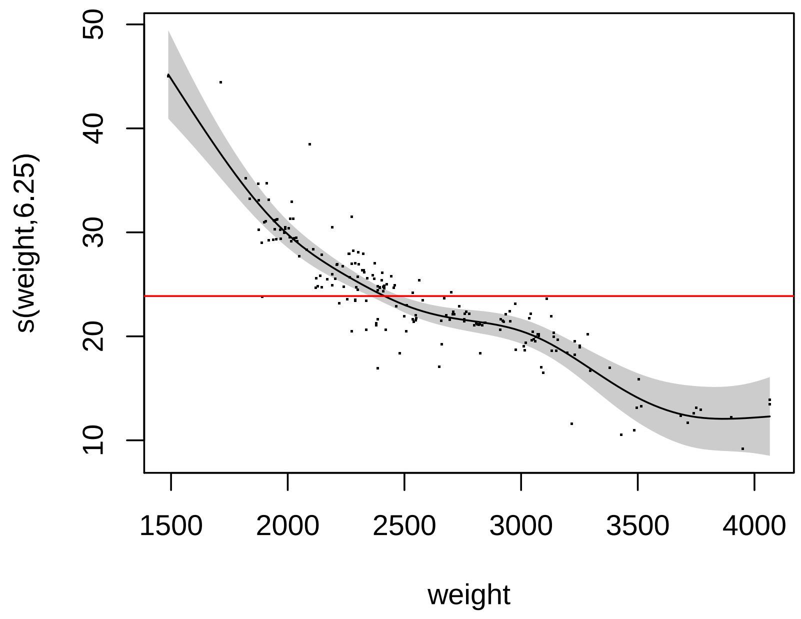


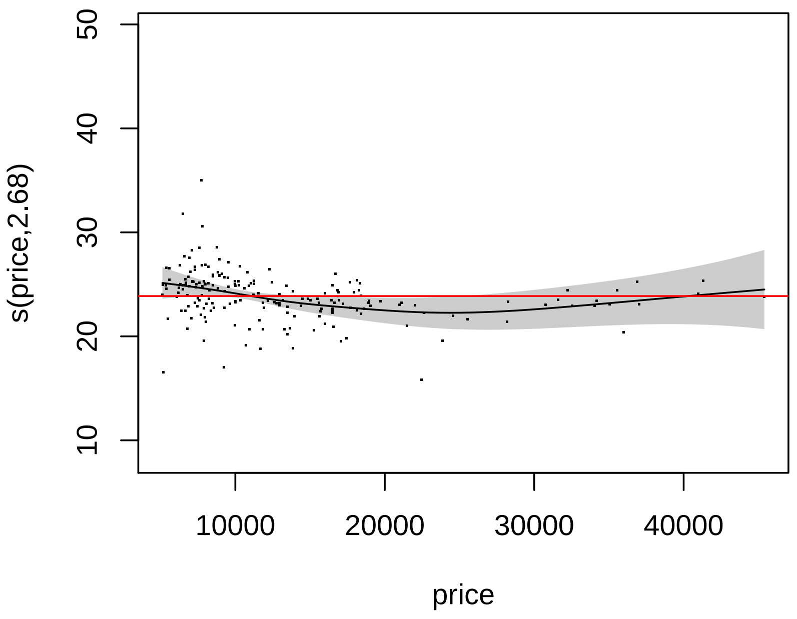


Significance of Smooth Terms

Overall significance of smooth. p-values are approximate.

One way to interpret significance: you cannot draw a horizontal line through the 95% confidence interval.

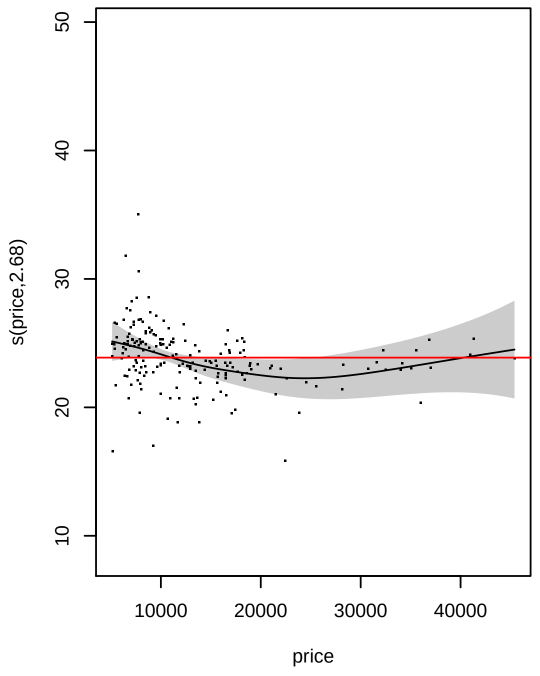


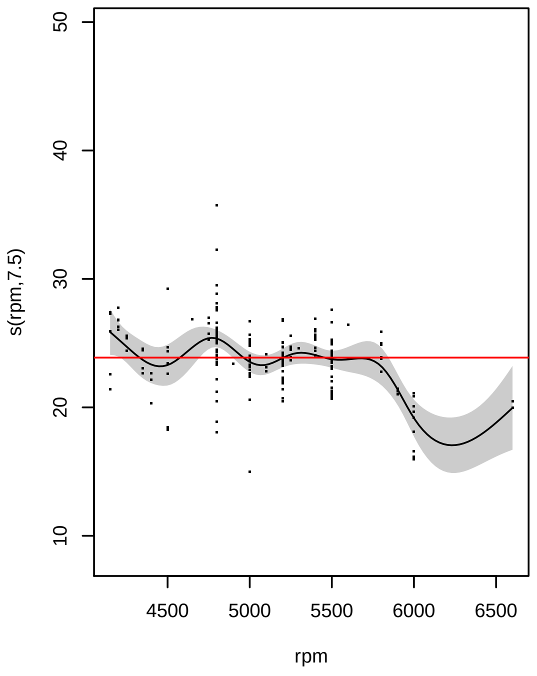


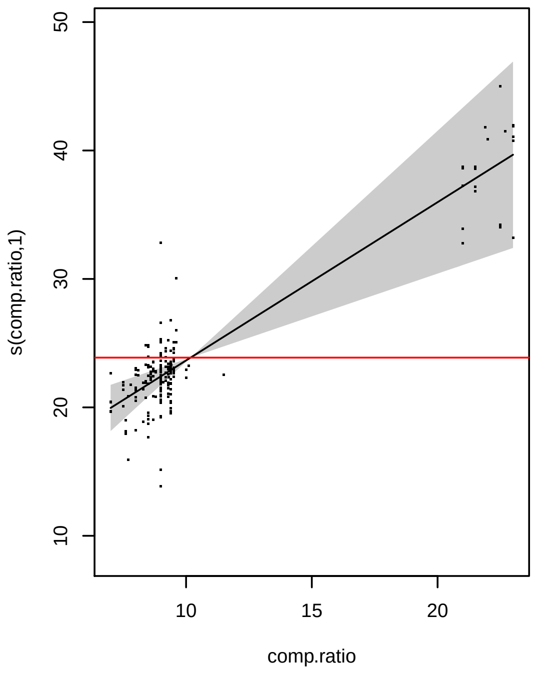
Significance and Effective Degress of Freedom

Two separate concepts, one does not imply the other !

	edf	Ref.df	F	p-value
s(weight)	6.254138	7.439475	20.909403	0.0000000
s(price)	2.681086	3.421421	1.678134	0.1549673
s(rpm)	7.499029	8.285177	8.533672	0.0000000
s(comp.ratio)	1.000337	1.000650	18.922944	0.0000222
s(width)	1.000696	1.001337	0.357271	0.5506887



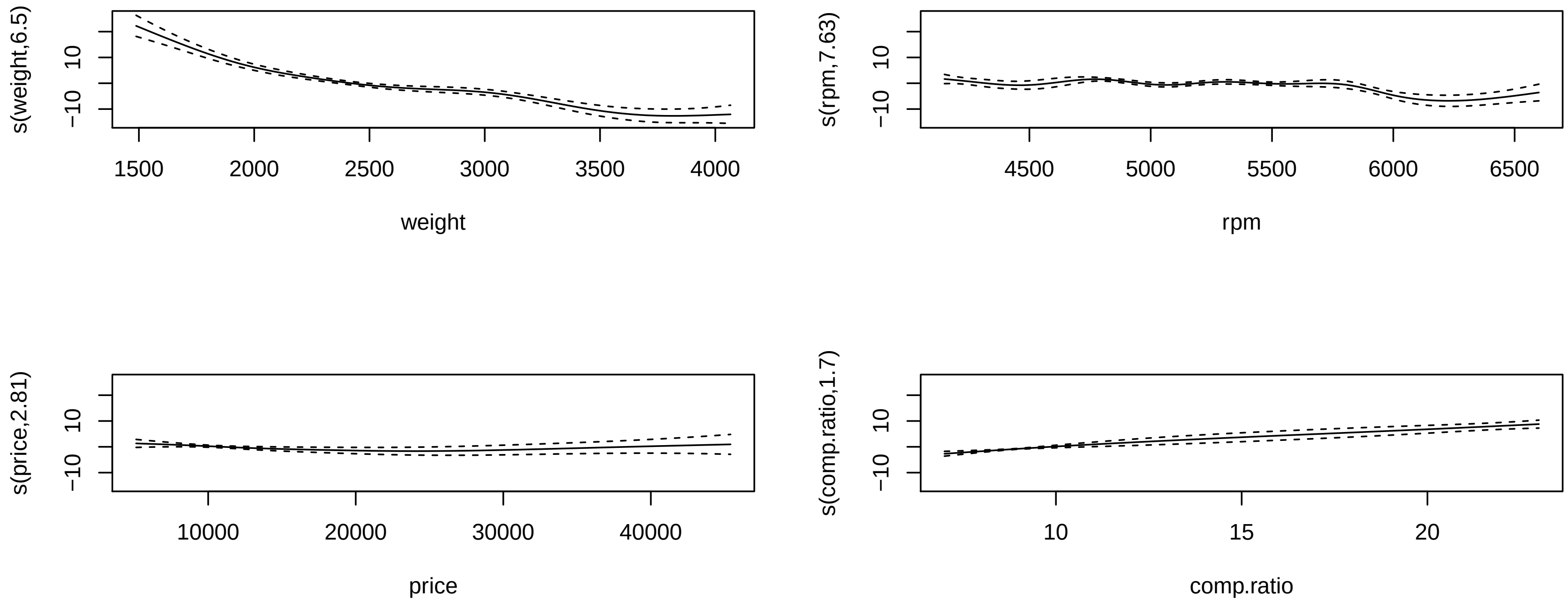




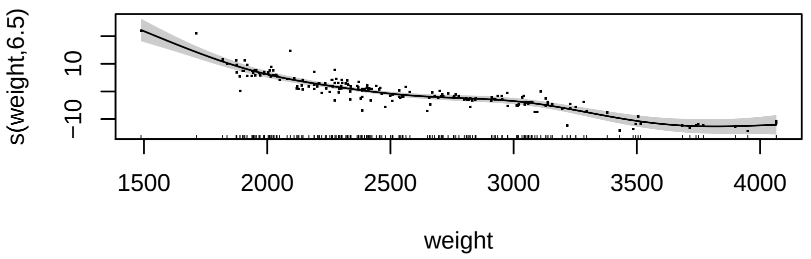
Visualizing GAMs

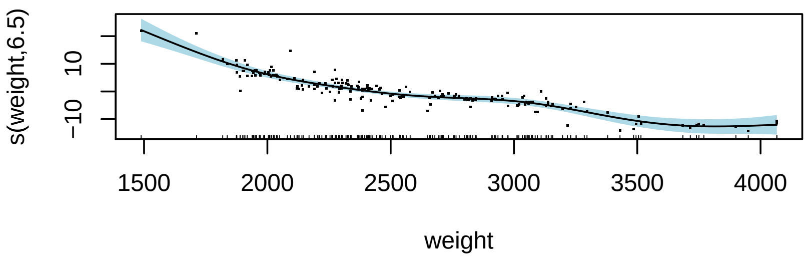
```
gam_mod <- gam(hw.mpg ~ s(weight) +  s(rpm) + s(price) + s(comp.ratio) , data =
```

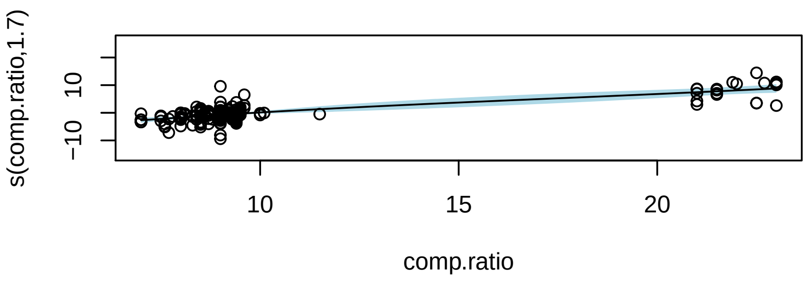
```
plot(gam_mod, pages=1, rug=FALSE)
```

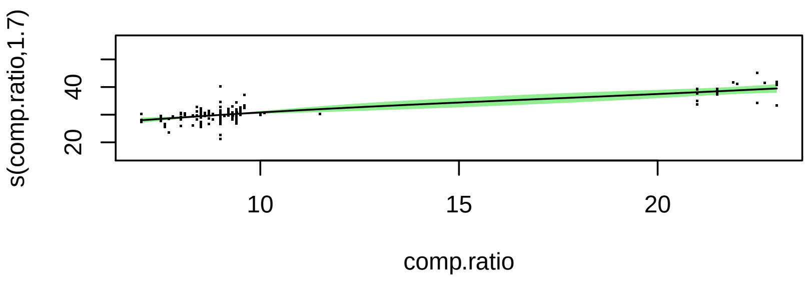


```
par(mfrow=c(2,2))
plot(gam_mod, select = 1, shade = TRUE, residuals = TRUE)
plot(gam_mod, select = 1, shade = TRUE, residuals = TRUE, shade.col ="lightblue"
plot(gam_mod, select = 4, shade = TRUE, residuals = TRUE, shade.col ="lightblue"
plot(gam_mod, select = 4, shade = TRUE, residuals = TRUE, shade.col ="lightgreen
```

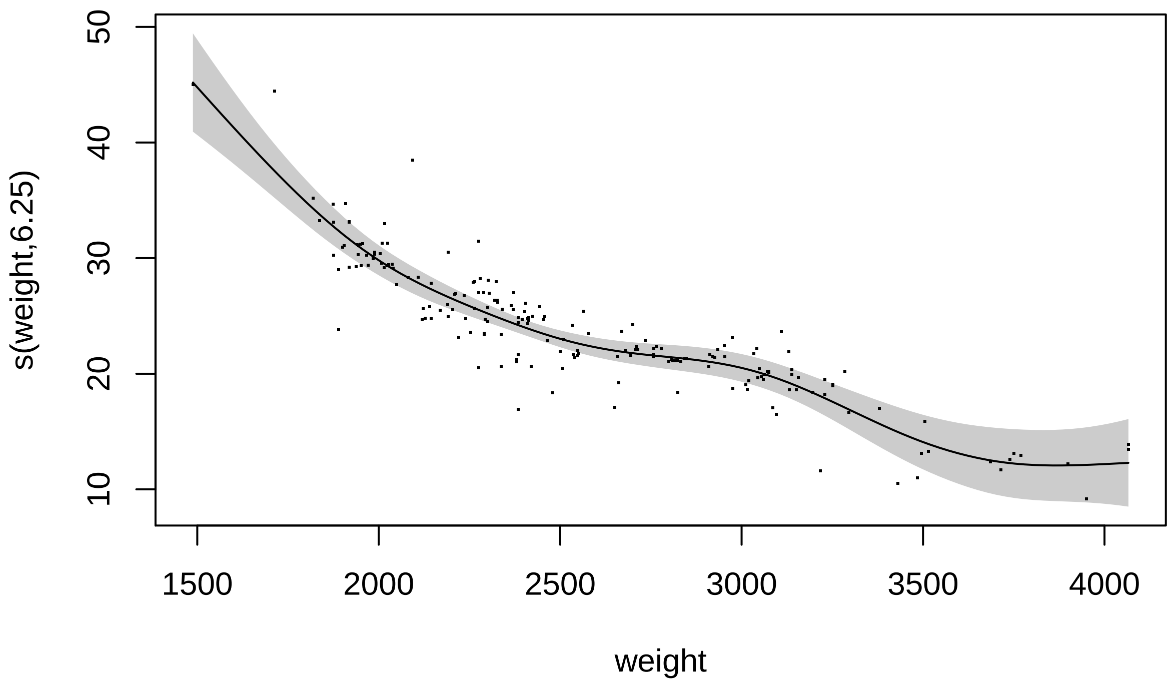








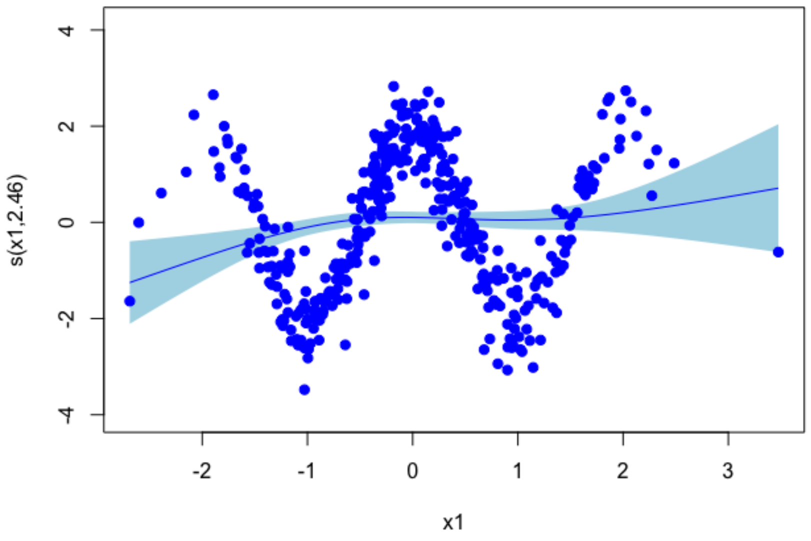
```
plot(mod_city4, select = 1, shade = TRUE, residuals = TRUE, cex=1.2,shift = coef
```



Model checking, Concurvity

Pitfall One: Inadequate Basis Number

```r
mod <- gam(y ~ s(x1, k = 4) + s(x2, k = 4),
           data = check_data, method = "REML")
```



Running gam.check

```
gam.check(mod)
```

```
Method: REML   Optimizer: outer newton
full convergence after 9 iterations.
Gradient range [-0.0001467222,0.00171085]
(score 784.6012 & scale 2.868607).
Hessian positive definite, eigenvalue range [0.00014,198.5]
Model rank =  7 / 7

Basis dimension (k) checking results. Low p-value
(k-index<1) may indicate that k is too low, especially
if edf is close to k'.

        k'  edf k-index p-value
s(x1) 3.00 1.00    0.35  <2e-16 ***
s(x2) 3.00 2.88    1.00    0.52
 ---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

Running gam.check (2)

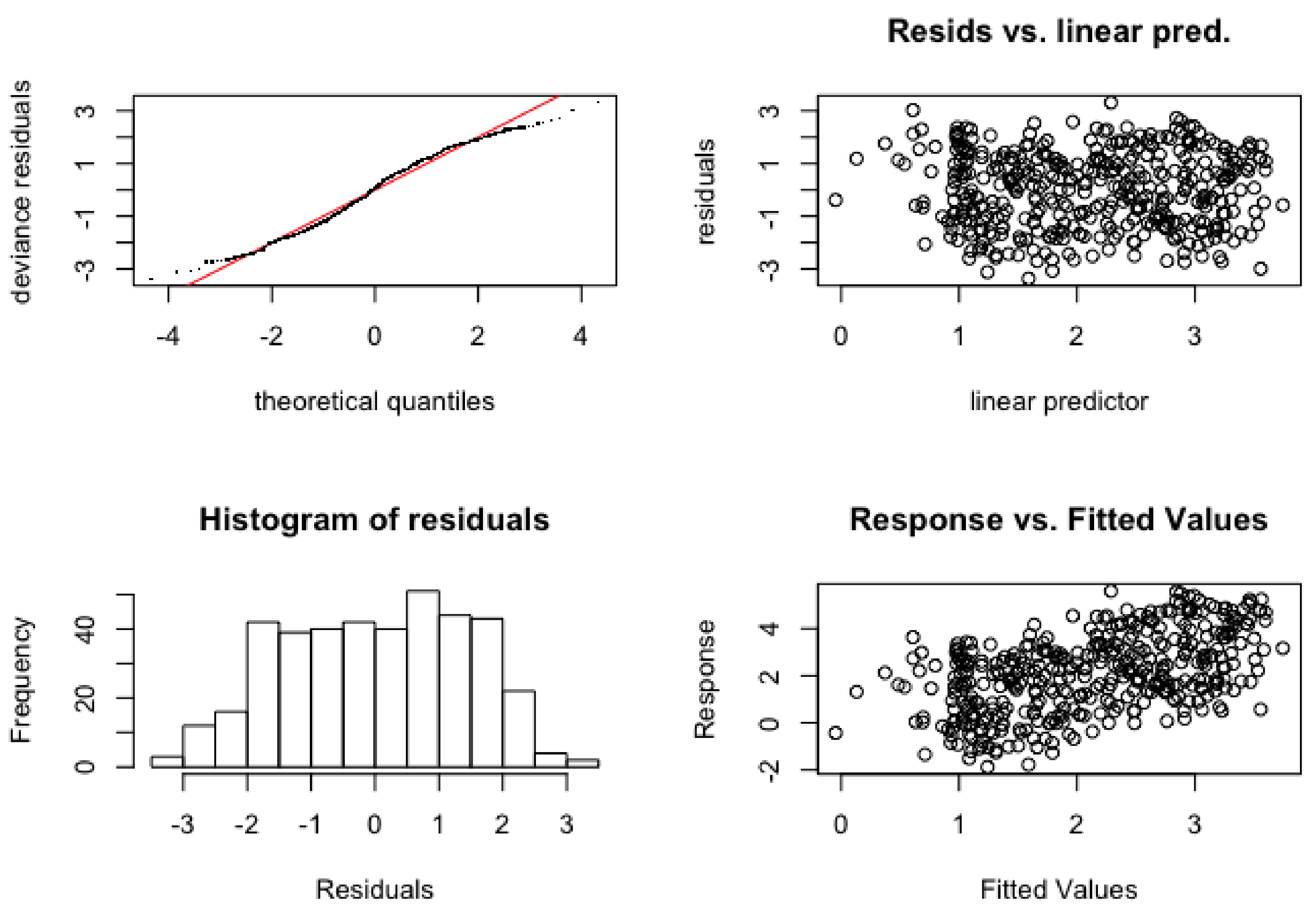
```
mod <- gam(y ~ s(x1, k = 12) + s(x2, k = 4),
                data = dat, method = "REML")
gam.check(mod)
```

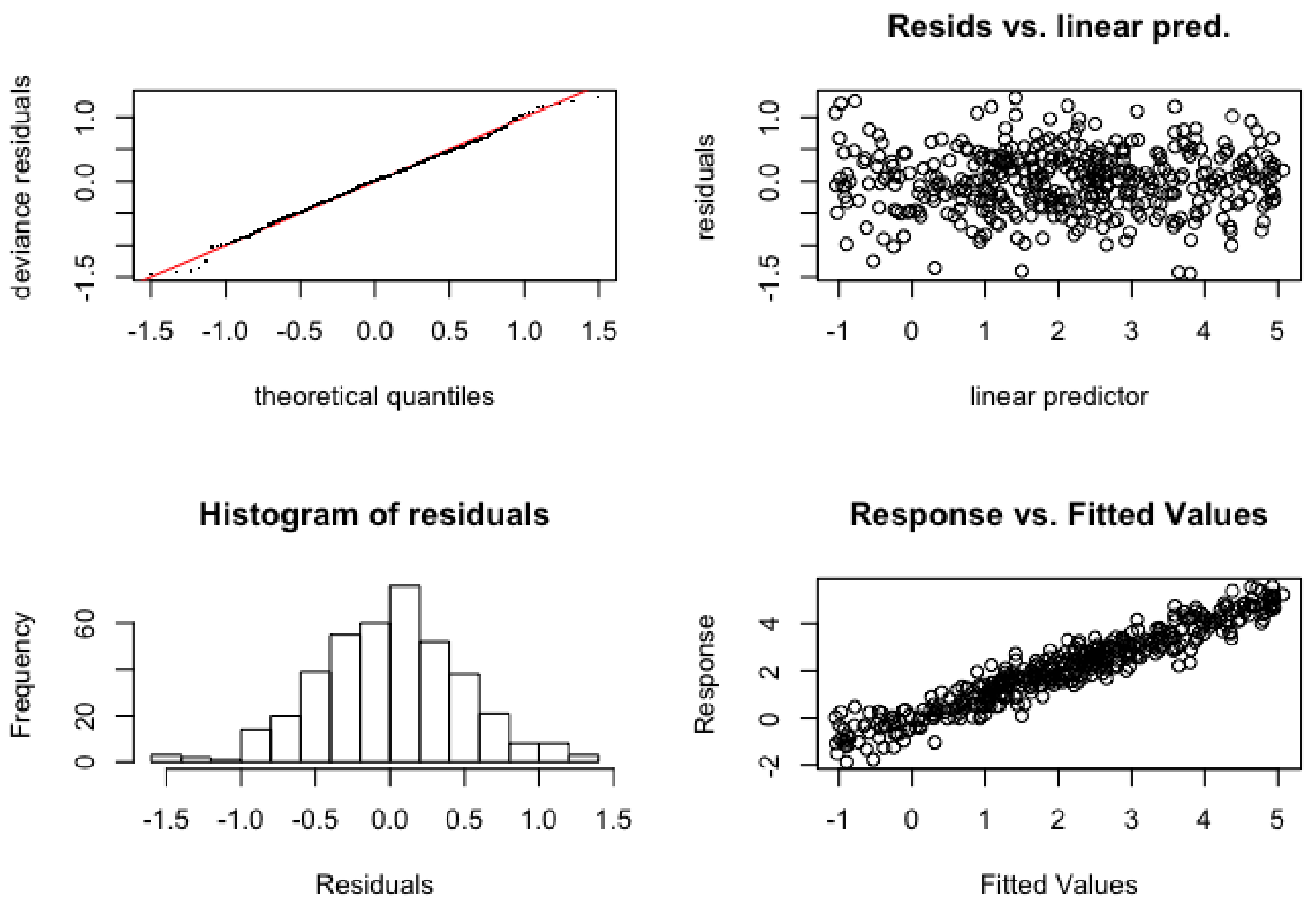
```
...

         k'   edf k-index p-value
s(x1) 11.00 10.85    1.05   0.830
s(x2)  3.00  2.98    0.89   0.015 *

...
```

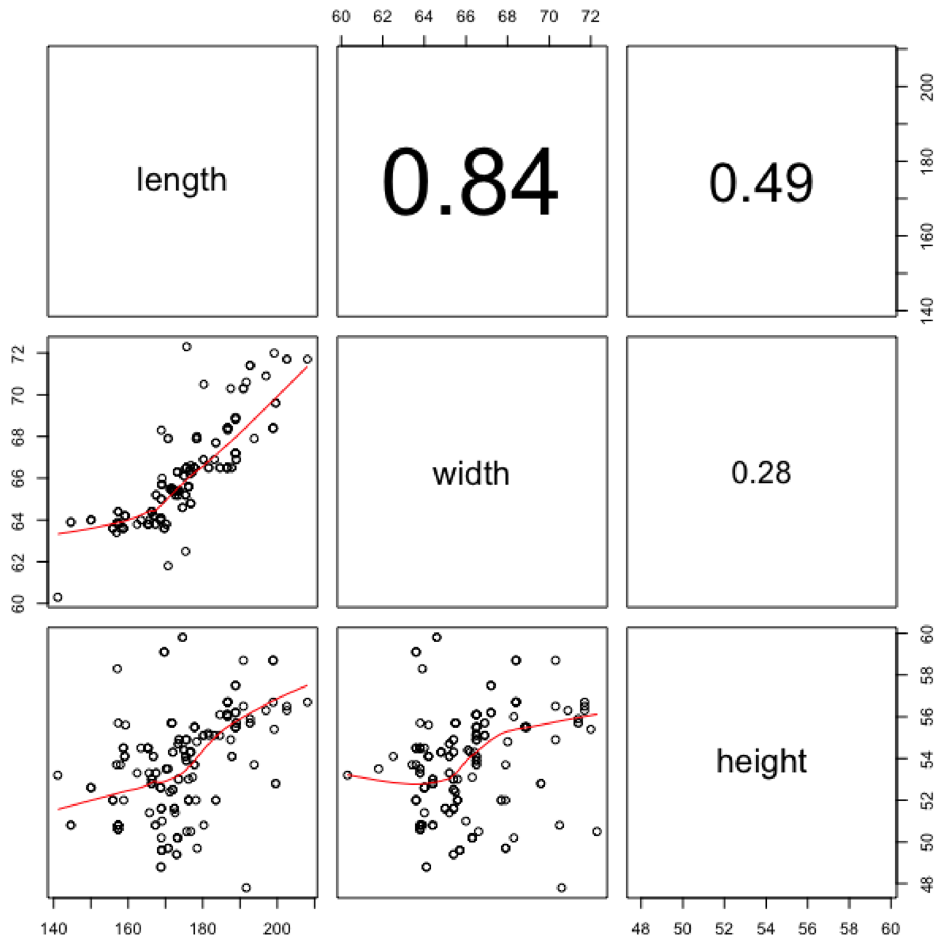
Running gam.check (3)

```
mod <- gam(y ~ s(x1, k = 12) + s(x2, k = 12),
            data = dat, method = "REML")
gam.check(mod)
```

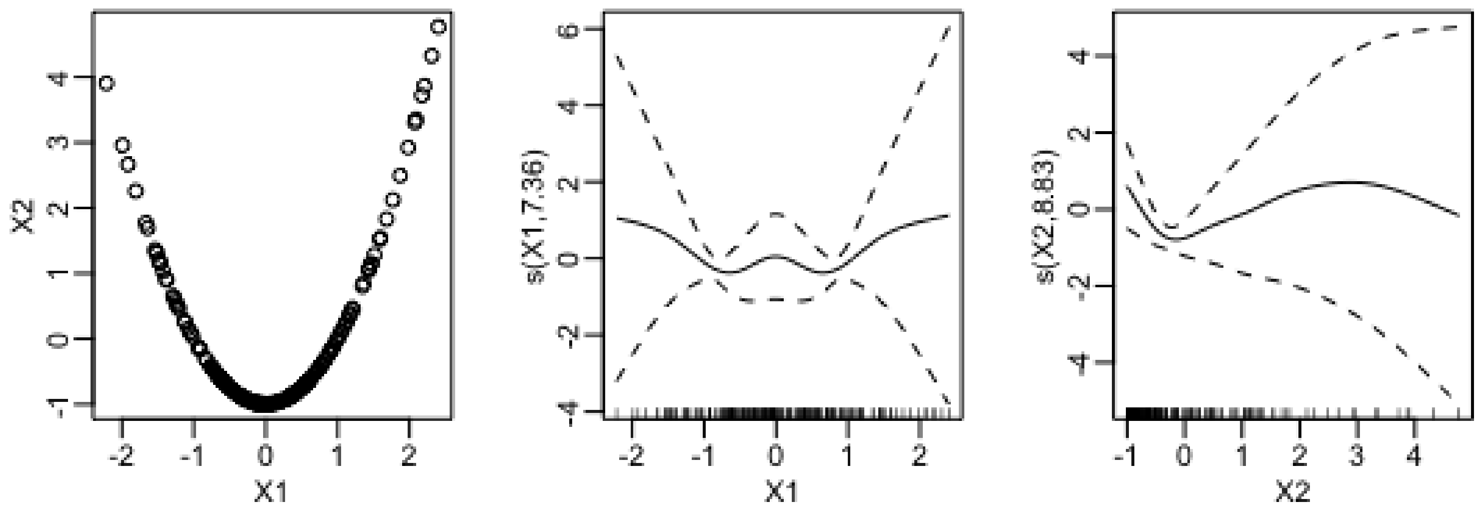
```
...
          k'   edf k-index p-value
s(x1) 11.00 10.86    1.08    0.94
s(x2) 11.00  7.78    0.94    0.12
...
```


Resids vs. linear pred.
deviance residuals
theoretical quantiles
residuals
linear predictor
Histogram of residuals
Frequency
Residuals
Response vs. Fitted Values
Response
Fitted Values

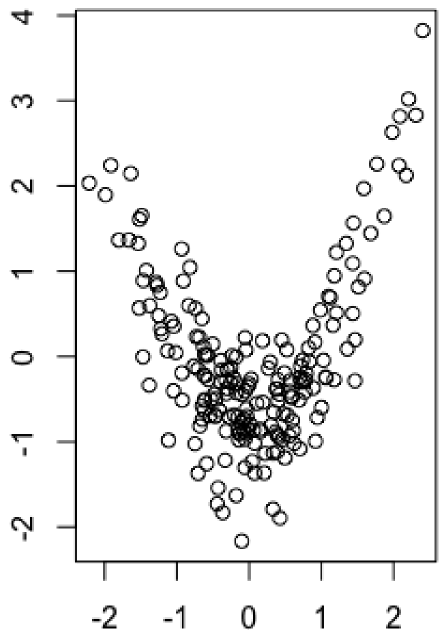

Resids vs. linear pred.
deviance residuals
theoretical quantiles
residuals
linear predictor
Histogram of residuals
Frequency
Residuals
Response vs. Fitted Values
Response
Fitted Values


length
0.84
0.49
width
0.28
height
60 62 64 66 68 70 72
140 160 180 200
48 50 52 54 56 58 60

Concurvity



The concurvity() function



```
concurvity(m1, full = TRUE)
```

```
         para s(X1) s(X2)
worst       0  0.84  0.84
observed    0  0.22  0.57
estimate    0  0.28  0.60
```

Pairwise concurvities

```
concurvity(model, full = FALSE)
```

```
$worst
      para s(X1) s(X2)
para     1  0.00  0.00
s(X1)    0  1.00  0.84
s(X2)    0  0.84  1.00

$observed                  |  $estimate
      para s(X1) s(X2)     |        para s(X1) s(X2)
para     1  0.00  0.00     |  para     1  0.00   0.0
s(X1)    0  1.00  0.57     |  s(X1)    0  1.00   0.6
s(X2)    0  0.22  1.00     |  s(X2)    0  0.28   1.0
```

NONLINEAR MODELING IN R WITH GAMS

Let's practice!

The gam library

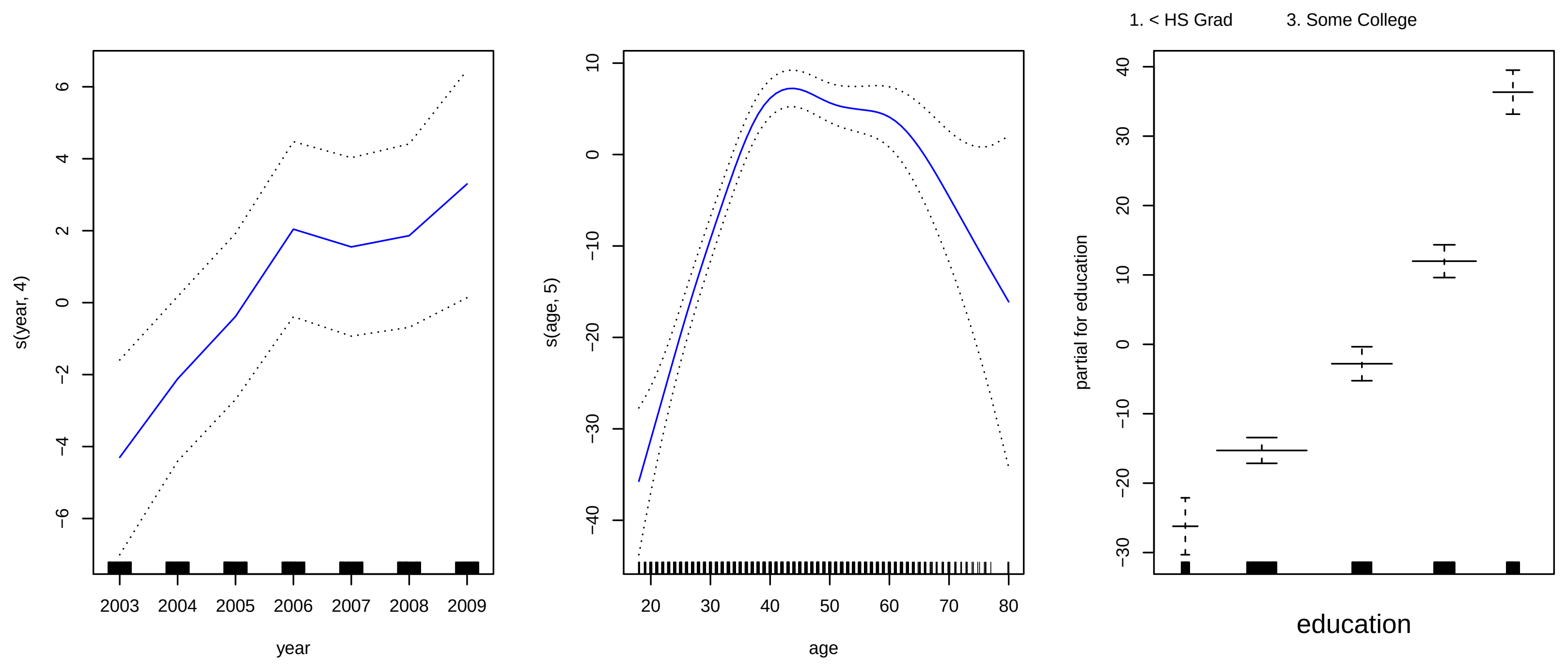
In order to fit more general sorts of GAMs, using smoothing splines or other components that cannot be expressed in terms of basis functions and then fit using least squares regression, we will need to use the gam library in R.

Smoothing/number of knots specified by user

A "simpler" version of mgcv. (no Bayesian smoothing) Note the slightly different syntax though (e.g., *df=4*, instead of *k=4*) !!

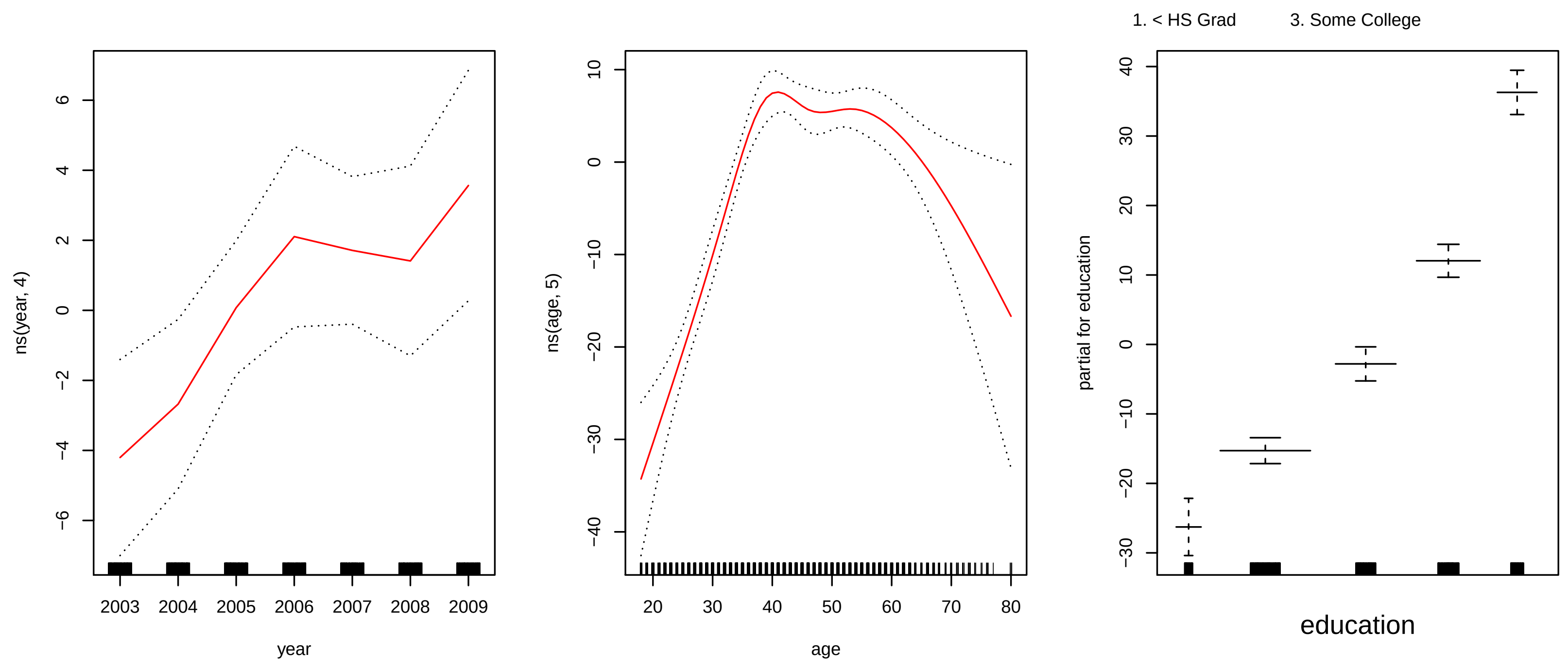
The `s()` function, which is part of the gam library, is used to indicate that we would like to use a smoothing spline. We specify that the function of year should have 4 degrees of freedom, and that the function of *age* will have 5 degrees of freedom. Since *education* is qualitative, we leave it as is, and it is converted into four dummy variables. We use the `gam()` function in order to fit a GAM using these components.

```
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data = Wage)

detach("package:mgcv", unload=TRUE)#avoid name space collision!
library(gam)
gam.m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
par(mfrow = c(1, 3));plot(gam.m3, se = TRUE, col = "blue")
```



Conveniently, even though gam1 is not of class gam but rather of class `lm`, we can still use `plot.gam()` on it.

```
par(mfrow = c(1, 3));plot.Gam(gam1, se = TRUE, col = "red")
```



In these plots, the function of year looks rather linear. We can perform a series of ANOVA tests in order to determine which of these three models is best: a GAM that excludes year (M1), a GAM that uses a linear function of year (M2), or a GAM that uses a spline function of year (M3).

```
gam.m1 <- gam(wage ~ s(age, 5) + education, data = Wage)
gam.m2 <- gam(wage~ year + s(age, 5) + education, data = Wage)
anova(gam.m1, gam.m2, gam.m3, test = "F")
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance       F    Pr(>F)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 14.4771 0.0001447 ***
## 3      2986    3689770  3   4071.1  1.0982 0.3485661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that there is compelling evidence that a GAM with a linear function of year is better than a GAM that does not include year at all (p-value=0.00014). However, there is no evidence that a non-linear function of year is needed (p-value=0.349). In other words, based on the results of this ANOVA, M2 is preferred. The summary() function produces a summary of the gam fit.

```r
summary(gam.m3)
```

```
##
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
## Deviance Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -119.43  -19.70   -3.33   14.17  213.48
##
## (Dispersion Parameter for gaussian family taken to be 1235.69)
##
##     Null Deviance: 5222086 on 2999 degrees of freedom
## Residual Deviance: 3689770 on 2986 degrees of freedom
## AIC: 29887.75
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## s(year, 4)     1   27162   27162  21.981 2.877e-06 ***
## s(age, 5)      1  195338  195338 158.081 < 2.2e-16 ***
## education      4 1069726  267432 216.423 < 2.2e-16 ***
## Residuals   2986 3689770    1236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F  Pr(F)
## (Intercept)
## s(year, 4)        3  1.086 0.3537
## s(age, 5)         4 32.380 <2e-16 ***
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

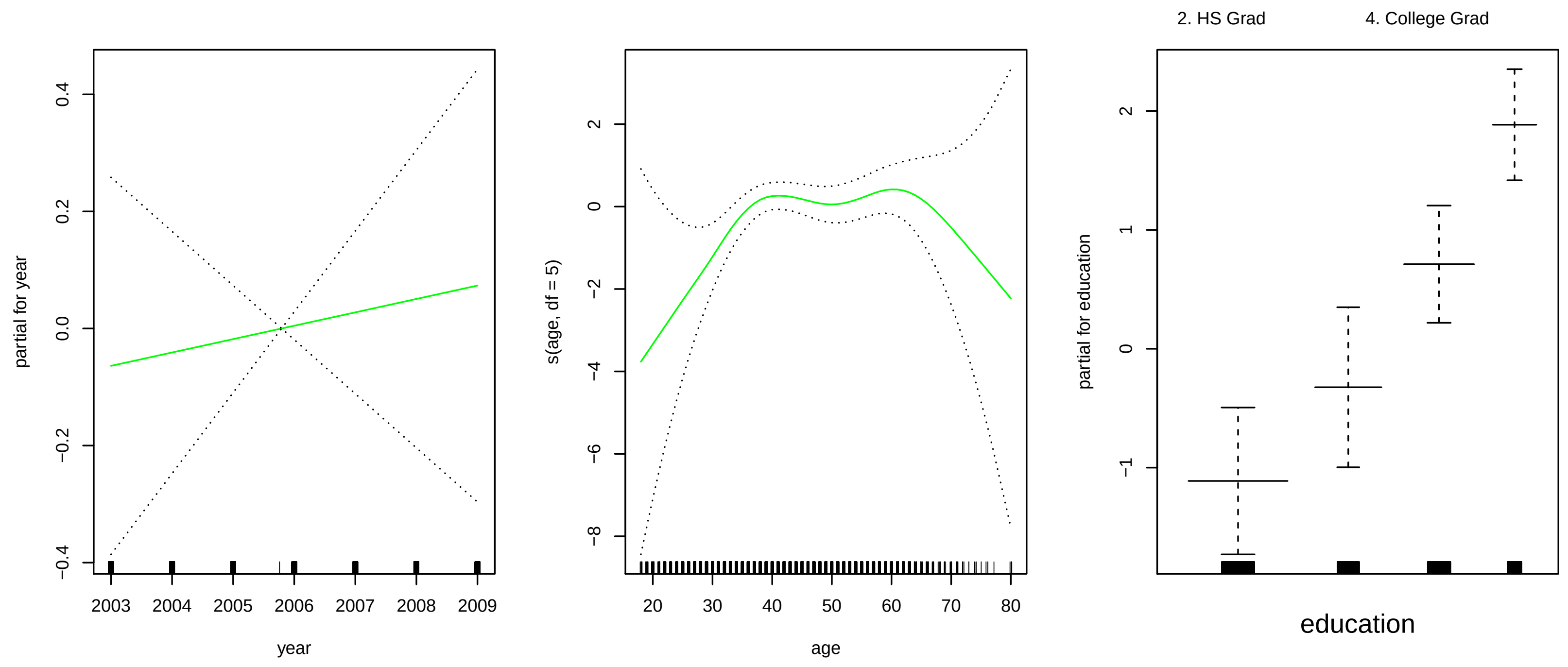
The p-values for *year* and *age* correspond to a null hypothesis of a linear relationship versus the alternative of a non-linear relationship. The large p-value for *year* reinforces our conclusion from the ANOVA test that a linear function is adequate for this term. However, there is very clear evidence that a non-linear term is required for *age*. We can make predictions from *gam* objects, just like from lm objects, using the predict() method for the class *gam*. Here we make predictions on the training set.

```r
preds <- predict(gam.m2, newdata = Wage)
```

Logistic regression

In order to fit a logistic regression GAM, we once again use the `I()` function in constructing the binary response variable, and set `family=binomial`.

```
gam.lr.s <- gam(I(wage > 250) ~ year + s(age, df = 5) + education,
                family = binomial, data = Wage,
                subset = (education != "1. < HS Grad"))
par(mfrow = c(1, 3)); plot(gam.lr.s, se = TRUE, col = "green")
```



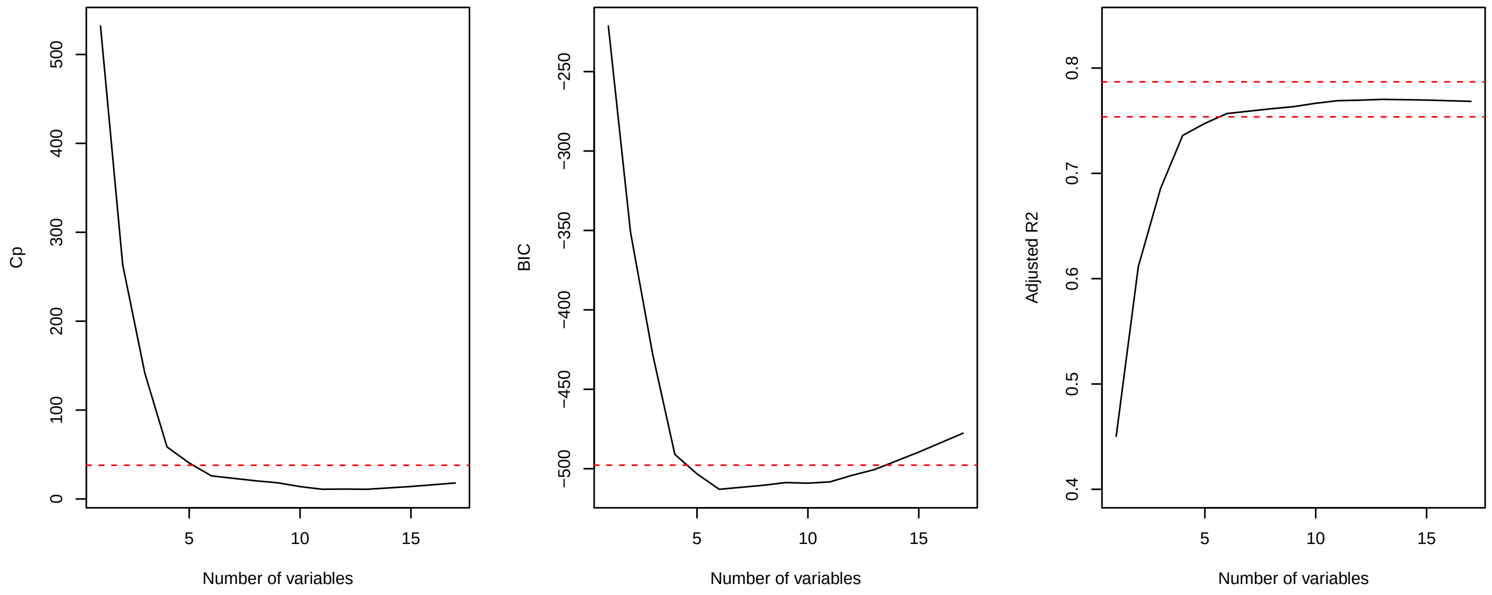
1. Exercises, Coding

This question relates to the "College" data set.

(a) Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

```r
library(leaps)
set.seed(1)
attach(College)
train <- sample(length(Outstate), length(Outstate) / 2)
test <- -train
College.train <- College[train, ]
College.test <- College[test, ]
fit <- regsubsets(Outstate ~ ., data = College.train, nvmax = 17, method =
fit.summary <- summary(fit)
par(mfrow = c(1, 3))
plot(fit.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
min.cp <- min(fit.summary$cp)
```

```r
std.cp <- sd(fit.summary$cp)
abline(h = min.cp + 0.2 * std.cp, col = "red", lty = 2)
abline(h = min.cp - 0.2 * std.cp, col = "red", lty = 2)
plot(fit.summary$bic, xlab = "Number of variables", ylab = "BIC", type='l')
min.bic <- min(fit.summary$bic)
std.bic <- sd(fit.summary$bic)
abline(h = min.bic + 0.2 * std.bic, col = "red", lty = 2)
abline(h = min.bic - 0.2 * std.bic, col = "red", lty = 2)
plot(fit.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
max.adjr2 <- max(fit.summary$adjr2)
std.adjr2 <- sd(fit.summary$adjr2)
abline(h = max.adjr2 + 0.2 * std.adjr2, col = "red", lty = 2)
abline(h = max.adjr2 - 0.2 * std.adjr2, col = "red", lty = 2)
```



Cp, BIC and adjr2 show that size 6 is the minimum size for the subset for which the scores are within 0.2 standard devitations of optimum.

```
fit <- regsubsets(Outstate ~ ., data = College, method = "forward")
coeffs <- coef(fit, id = 6)
names(coeffs)
```

```
## [1] "(Intercept)" "PrivateYes"  "Room.Board"  "PhD"         "perc.alumni"
## [6] "Expend"      "Grad.Rate"
```

(b) Fit a GAM on the training data, using out-of-state tuition as the response and

the features selected in the previous step as the predictors. Plot the results, and explain your findings.

(c) Evaluate the model obtained on the test set, and explain the results obtained.

(d) For which variables, if any, is there evidence of a non-linear relationship with the response ?

GAM Exercises part III

Autocorrelations

GAM or MLR have assumptions in a model that errors (residuals) are identically and independently distributed (i.i.d.). In the case of the time series regression, it is very strong assumption, which is here, logically, not fulfilled. Present time series values are highly correlated with past values, so errors of the model will be correlated too. This phenomenon is called an autocorrelation. This implies that estimated regression coefficients and residuals of a model might be negatively biased, which also implies that previously computed p-values of statistical tests or confidence intervals are wrong.

How can we handle this situation? By inclusion of autoregressive model (AR) for errors in our model. So we have the model with the term for errors like this:

$$y_i = \beta \cdot X + \epsilon_i,\ \epsilon_i = \phi\epsilon_{i-1} + \upsilon_i$$

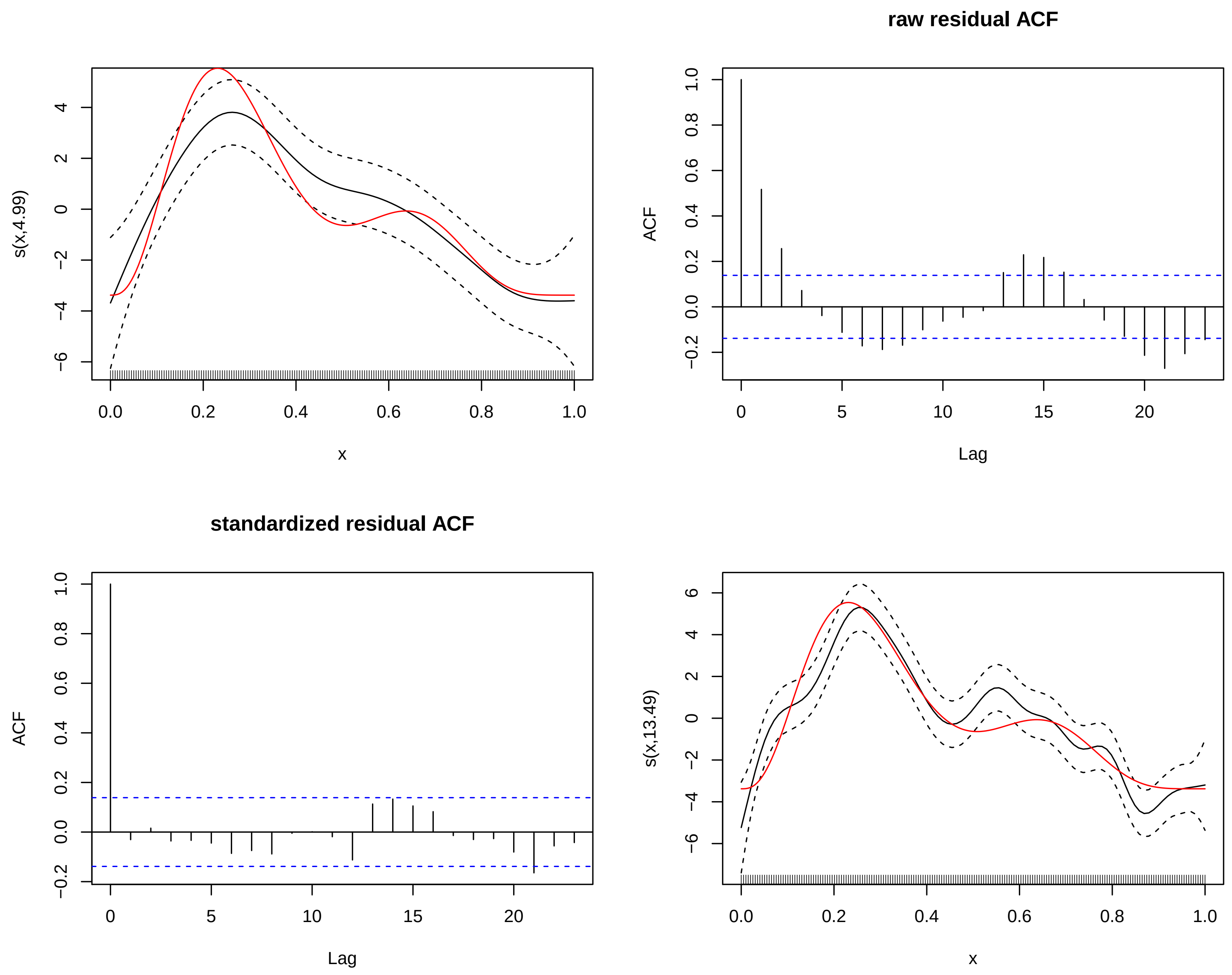
where the second equation is a classical AR(1) process and ϕ

is an unknown autoregressive coefficient to be estimated. Errors can be also nested within a week, which is in our case more appropriate, because of the double seasonal character of our time series. You can add also higher orders of AR process and also MA (moving average) model. You can read more about estimation and modeling of this kind of models in an excellent book by Box, Jenkins, and Reinsel: Time Series Analysis.

It's possible to add correlation term for errors with function gamm, which stands for GAM mixture models. It calls the `lme` function from package `nlme`.

```r
detach("package:gam", unload=TRUE)#avoid name space collision!
library(mgcv)
## now an example with autocorrelated errors....
n <- 200;sig <- 2
x <- 0:(n-1)/(n-1)
f <- 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
e <- rnorm(n,0,sig)
for (i in 2:n) e[i] <- 0.6*e[i-1] + e[i]
y <- f + e
op <- par(mfrow=c(2,2))
## Fit model with AR1 residuals
```

```
b <- gamm(y~s(x,k=20),correlation=corAR1())
plot(b$gam);lines(x,f-mean(f),col=2)
## Raw residuals still show correlation, of course...
acf(residuals(b$gam),main="raw residual ACF")
## But standardized are now fine...
acf(residuals(b$lme,type="normalized"),main="standardized residual ACF")
## compare with model without AR component...
b <- gam(y~s(x,k=20))
plot(b);lines(x,f-mean(f),col=2)
```


raw residual ACF
standardized residual ACF
s(x,4.99)
x
ACF
Lag
s(x,13.49)

Appendix I

Locations of knots

For penalized regression spline, the exact locations are not important, as long as:

- **k is adequately big;**
- **the spread of knots has good, reasonable coverage.**

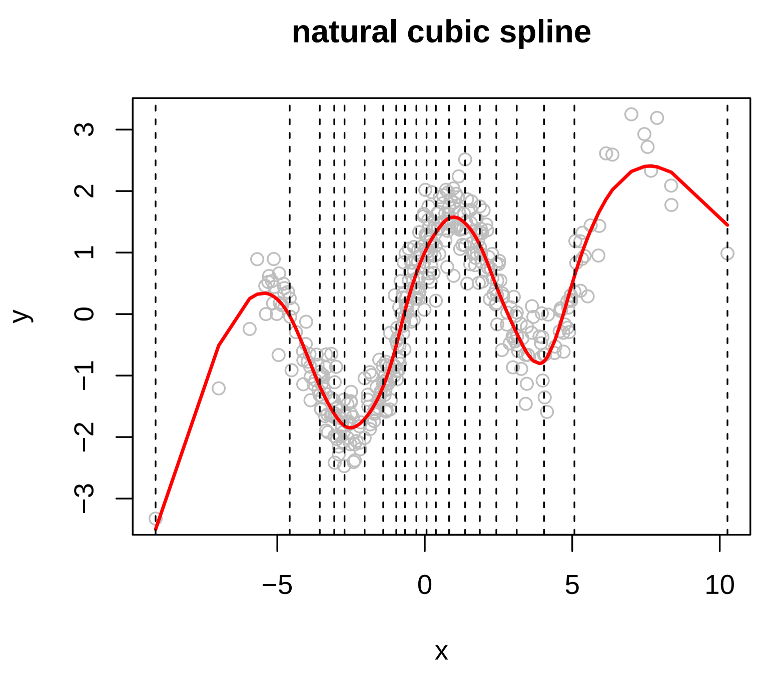
By default:

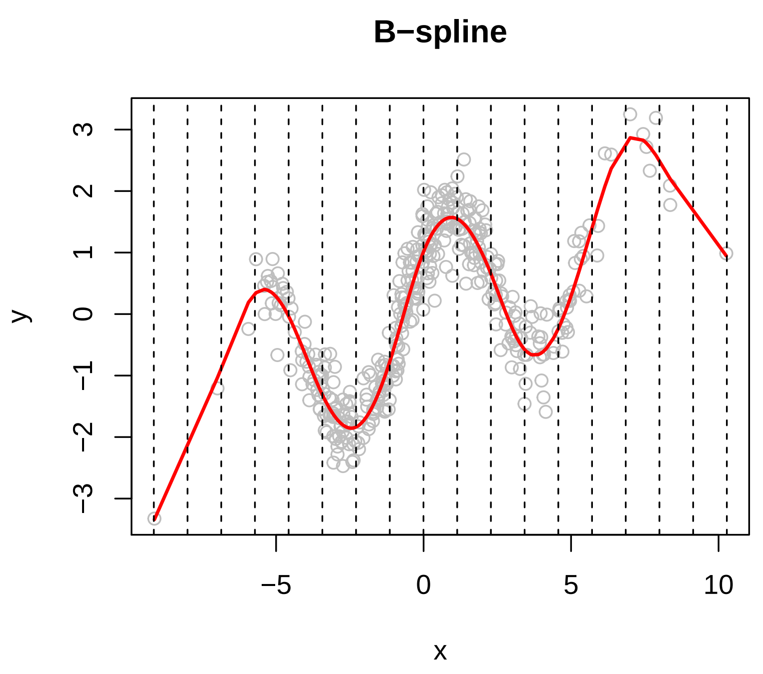
- **natural cubic regression spline bs = ‘cr’ places knots by quantile;**
- **B-splines family (bs = ‘bs’, bs = ‘ps’, bs = ‘ad’) place knots evenly.**

Compare the following:

```
library(mgcv)

## toy data
set.seed(0); x <- sort(rnorm(400, 0, pi))  ## note, my x are not uniformly sampl
set.seed(1); e <- rnorm(400, 0, 0.4)
y0 <- sin(x) + 0.2 * x + cos(abs(x))
y <- y0 + e

## fitting natural cubic spline
```

```r
cr_fit <- gam(y ~ s(x, bs = 'cr', k = 20))
cr_knots <- cr_fit$smooth[[1]]$xp  ## extract knots locations

## fitting B-spline
bs_fit <- gam(y ~ s(x, bs = 'bs', k = 20))
bs_knots <- bs_fit$smooth[[1]]$knots  ## extract knots locations

## summary plot
par(mfrow = c(1,2))
plot(x, y, col= "grey", main = "natural cubic spline");
lines(x, cr_fit$linear.predictors, col = 2, lwd = 2)
abline(v = cr_knots, lty = 2)
plot(x, y, col= "grey", main = "B-spline");
lines(x, bs_fit$linear.predictors, col = 2, lwd = 2)
abline(v = bs_knots, lty = 2)
```





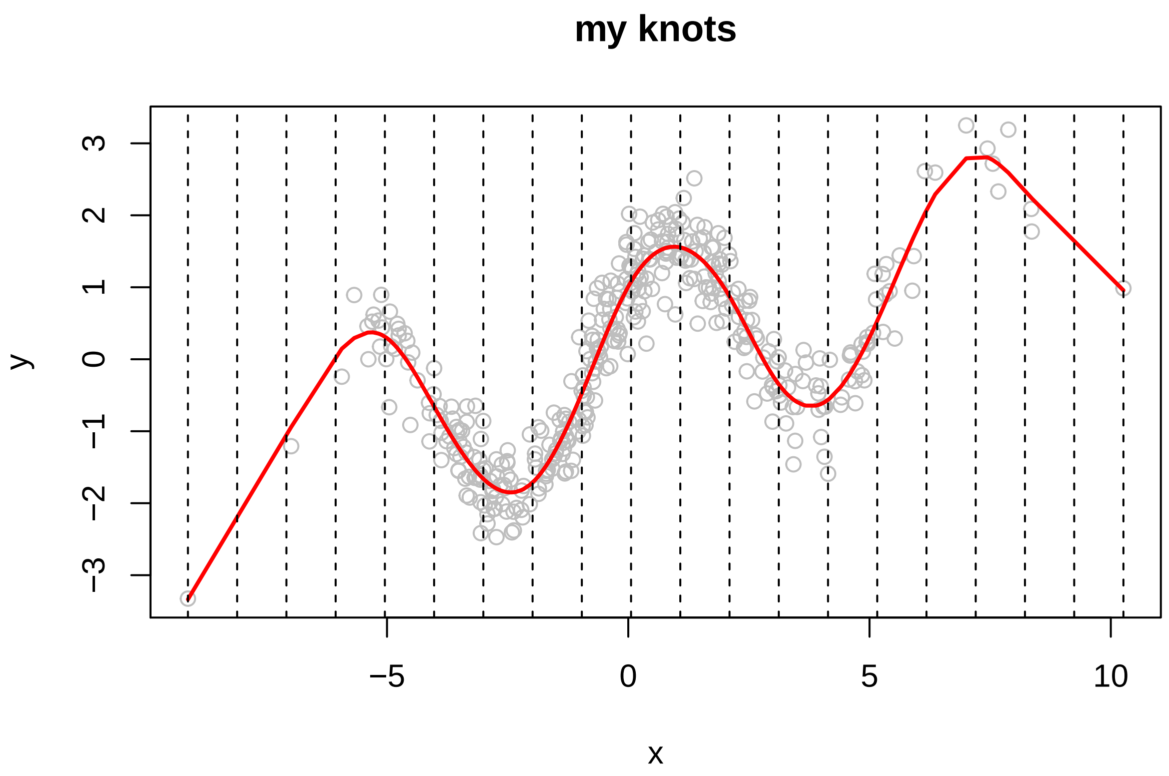
Setting your own knots locations:

You can also provide your customized knots locations via the knots argument of gam() (yes, knots are not fed to s(), but to gam()). For example, you can do evenly spaced knots for cr:

```
xlim <- range(x)  ## get range of x
myfit <- gam(y ~ s(x, bs = 'cr', k =20),
         knots = list(x = seq(xlim[1], xlim[2], length = 20)))
```

Now you can see that:

```
my_knots <- myfit$smooth[[1]]$xp
plot(x, y, col= "grey", main = "my knots");
lines(x, myfit$linear.predictors, col = 2, lwd = 2)
abline(v = my_knots, lty = 2)
```



Appendix II, REML

REML stands for *REstricted Maximum Likelihood.*

The discussion below is relatively technical and readers may want to consult Simon

Wood's excellent "Generalized Additive Models: An Introduction with R" while reading.

Smoothing parameter estimation in `mgcv`

`mgcv` has a number of options for selecting the smoothness of terms in the model. Smoothness selection method is specified via the `method=` argument and has the following options: `"GCV.Cp"`, `"GACV.Cp"`, `"REML"`, `"P-REML"`, `"ML"` and `"P-ML"`. These methods can be separated into two groups: "prediction error" (`"GCV.Cp"` and `"GACV.Cp"`) and "likelihood" (`"REML"`, `"P-REML"`, `"ML"` and `"P-ML"`). Here we'll just address `"GCV.Cp"` vs. `"REML"` (though similar arguments can be levelled in favour of `"ML"` vs `"GCV.Cp"`. `mgcv` uses `"GCV.Cp"` as a default, though this is mainly for backwards compatibility (Simon Wood, pers. comm.).

Theory

Both REML and GCV try to do the same thing: make the smooths in your model just wiggly enough and no wigglier. It has been shown that GCV will select optimal smoothing parameters (in the sense of low prediction error) when the sample size is infinite. In practice there is presumably some sample size at which this asymptotic result kicks in, but few of us have data of that size. At smaller (finite) sample sizes GCV can develop multiple minima making optimisation difficult and therefore tends to give

more variable estimates of the smoothing parameter. GCV also tends to undersmooth (i.e., smooths are more wiggly than they should be), as it penalizes overfit weakly (which makes sense for a prediction error based critera, where you want your model to fit your data well). This was shown in work by Phil Reiss and Todd Ogden (2009) as well as Simon Wood (2011). They also showed that REML (and ML, marginal likelihood) penalize overfitting more and therefore have more prounounced optima, leading to fewer optimisation issues and less variable estimates of the smoothing parameter.

This is best illustrated in this figure from Wood (2011) which shows the (log) smoothing parameter vs. REML and GCV. The ticks on the horizontal axis show where the optimum smoothing parameter lies for each example data set (profile of the REML/GCV criteria shown as lines). The REML ticks are close together and the lines all look similar. For GCV this is not the case.

Section 1.1 of Wood (2011) gives an overview of these issues, though the rest of the paper is somewhat technical.

Practice

In practice I've analysed multiple datasets using both REML and GCV and in some cases have seen very different results when it comes to model and term selection. If GCV is prone to undersmoothing at finite sample sizes, then we will end up fitting models that are more wiggly than we want. Overfitting is surely something we want to

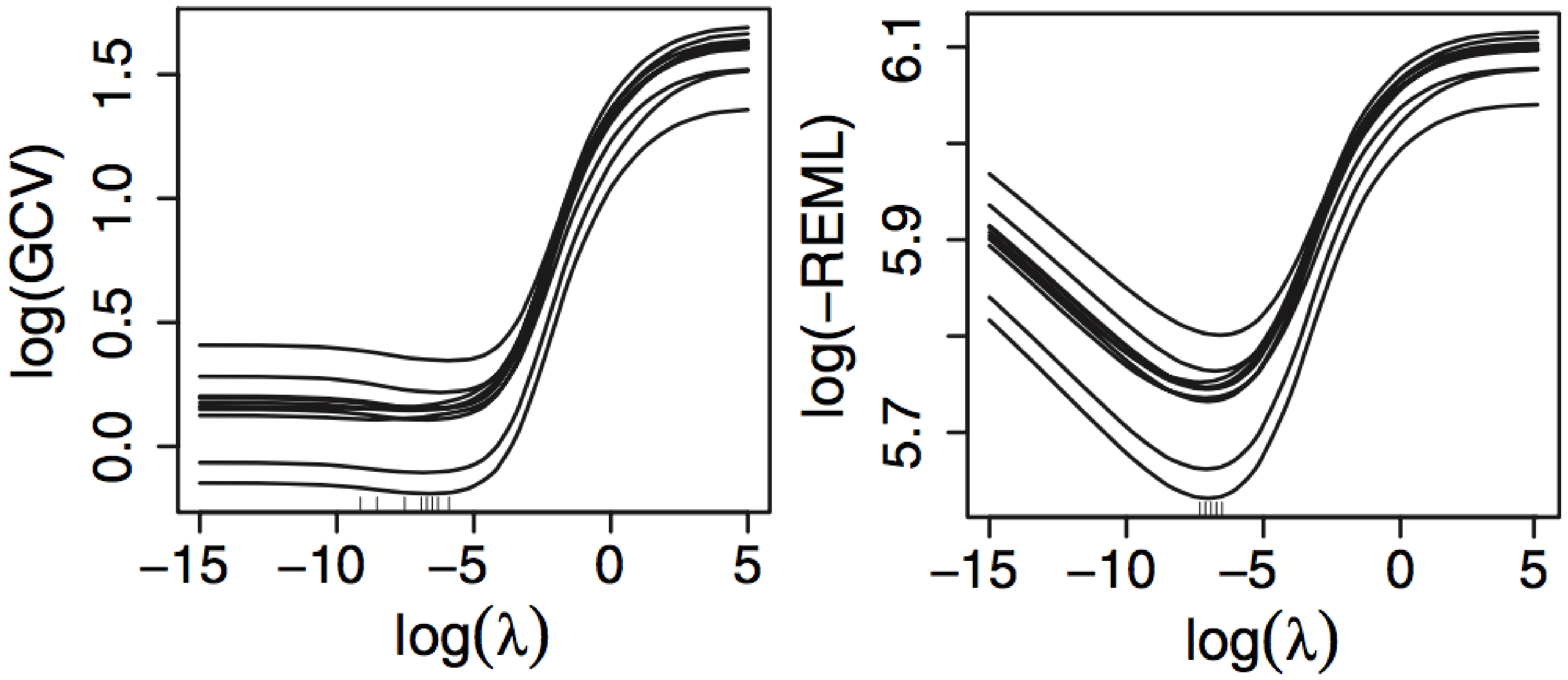


Figure 1: gcv and reml scores as functions of the smoothing parameter

avoid. It is not clear how many datasets (or what particular type of data) is vulnerable to this issue, but we (Distance Development Team) thought it best to switch to REML by default to avoid potential overfitting and highly variable smoothing parameter estimates.

Model comparison

Note that moving to REML, we need to be careful about model comparison. Models fitted with REML cannot be compared by their REML scores when their unpenalized components are different. In practice this means that models that use a shrinkage basis (e.g., `"ts"` or `"cs"`) for all terms in the model or that use `select=TRUE` are fine but other models are not.

If marginal likelihood (ML) is used, we can compare those scores.

One can compare AIC (extracted using the `AIC()` function in R) of models fitted with REML or ML (or GCV), though obviously this is not the only model selection step!

References

- Reiss, P.T. and Ogden, R.T. (2009) Smoothing parameter selection for a class of semiparametric linear models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71, 505–523. pdf

- **Wood, S.N. (2006) Generalized Additive Models. CRC Press.**
- **Wood, S.N. (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73, 3–36. pdf**