Survival Models: Week 10

Methods of Graduation/Smoothing

- Graduation is the process of smoothing a set of crude mortality rates (or force of mortality).
- The process of graduation allows a "pooling" of information and produces smooth rates that are generally more desirable (e.g. in premium pricing).
- There are many methods available for smoothing.
- Excellent resource is "The Elements of Statistical Learning" by Hastie, Tibshirani and Friedman.

Mathematical Formula

In the actuarial context the following mathematical formulae are sometimes used to perform a graduation:

Recall

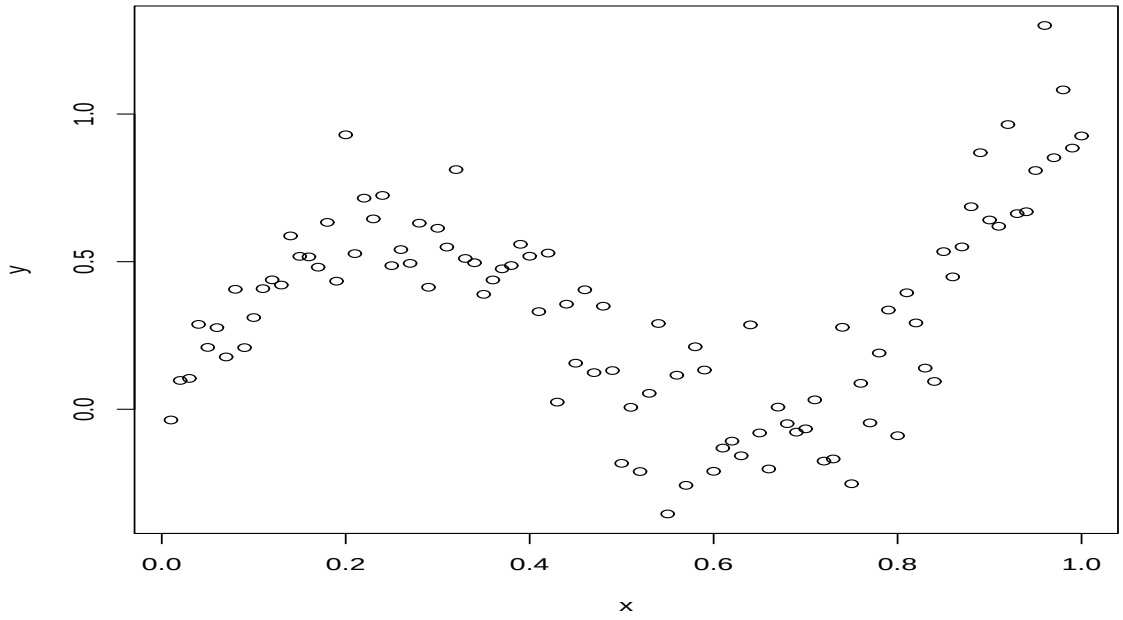
- Gompertz: $\mu_x = BC^x$
- Makeham: $\mu_x = A + BC^x$

These models can be fit using methods including maximum likelihood estimation and weighted least squares. These formulae have only two and three parameters respectively. It is hard to imagine such formulae being able to fit the mortality rates of the entire life span very well.

In this course our focus will be on more general methods of performing a graduation. These methods are more commonly refereed to as "smoothing" methods in statistics.

Motivating Example

We observe data, shown below, on a response y and a covariate x.



Motivating Example

- Our goal is to model the underlying relationship between x and y, denoted $f(x)$.
- We may only want to assume that this relationship is "smooth" and not specify a particular functional form.
- We will discuss a range of methods for achieving this goal.

Kernel Smoothing

- Estimates $f(x^*)$ using information contained in observations that are close to the point of interest x^*.
- The weight assigned to each observation is determined by the form of the *kernel*. The kernel typically has the following two properties:

  1. Observations further away from x^* receive less weight.
  2. The kernel is symmetric.

Kernel Smoothing

The kernel smoothed value at point x^* is given by:

$$\hat{f}(x^*) = \hat{y}(x^*) = \sum_{j=1}^{n} w_j y_j,$$

where y_j is the jth response value and w_j is the weight assigned to the jth response value. The w_j are given by:

$$w_j = \frac{K(\frac{|x^* - x_j|}{b})}{\sum_{m=1}^{n} K(\frac{|x^* - x_m|}{b})},$$

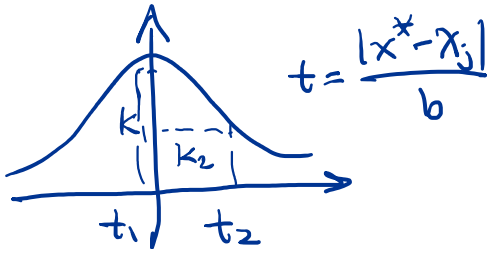
where $K(.)$ is the kernel and b is the bandwith.

Kernel Smoothing

Some common kernels include:

- The normal kernel: $K(t) = \frac{\exp(-t^2/2)}{\sqrt{2\pi}}$.
- The triangle kernel: $K(t) = 2 - 4|t|$, $|t| \leq 0.5$ (and 0 otherwise).
- The box kernel: $K(t) = 1$, $|t| \leq 0.5$ (and 0 otherwise).

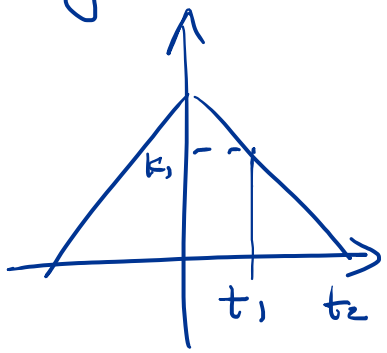
normal kernel



$t_1 < t_2$

$k_1 > k_2$

triangle kernel



$k_2 = 0$?

box every weight is the same = 1

as long as it is in the box

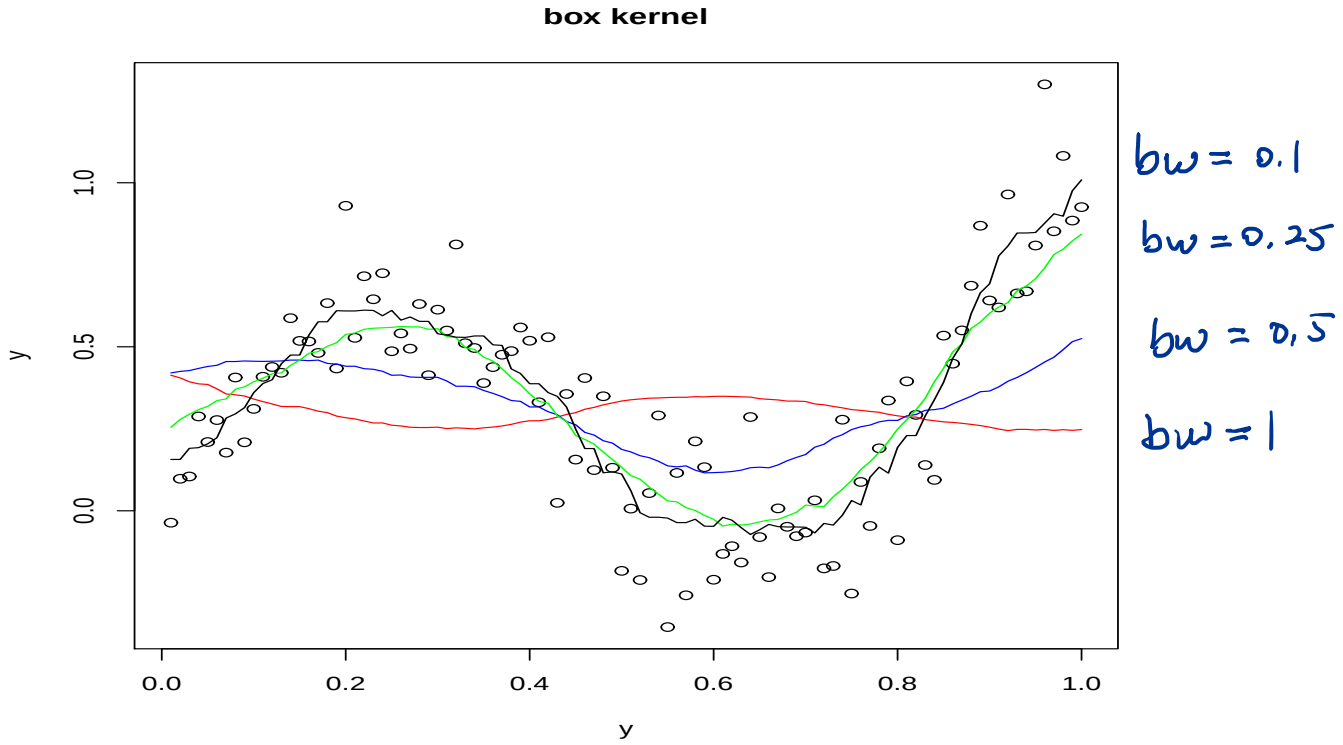
Kernel Smoothing

Some notes:

1. The further a point is from x^* the smaller the weight assigned to that point.
2. The smaller the value b the quicker the weights assigned to observations decline.
3. The larger the value of b the smoother the kernel.
4. The bandwidth is referred to as a "tuning parameter".

When b decreases, t increases, it is true that some observations will receive less weights, especially for those far away from x*. But for the observations very close to x*, the weights assigned to them are actually larger. Imagine the extreme case that b is very very small, all weights are assigned to the observation that closest to x*. So the weights decline for majority of observations but increase for the observation closest to x*. In this situation, we get a least smooth curve, e.g. essentially, no smoothing has been done. Therefore, a better way to illustrate this is if b becomes smaller, the weights will be more concentrated on the observations that are close to x*.

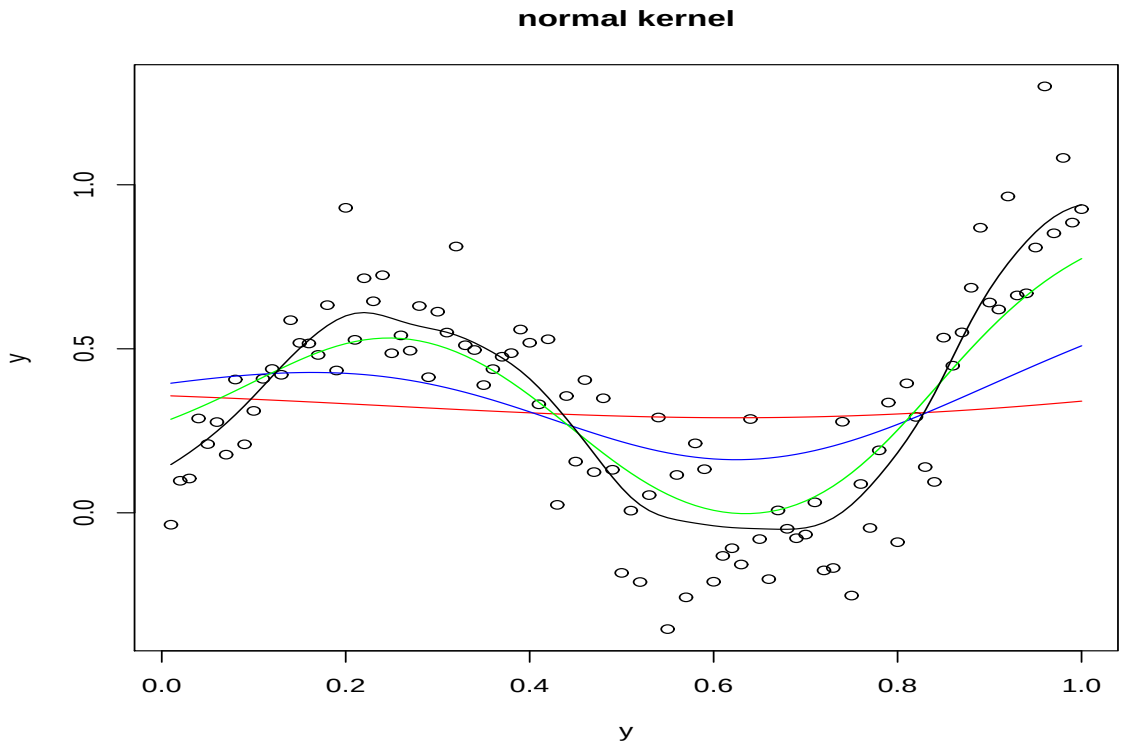
Example: Box Kernel Smoothing



Note: Looks "rough" - due to discrete nature of entry and exit of points.

```
plot(x,y,ylab="y",xlab="y",main="box kernel")
fit<-ksmooth(x,y,kernel="box",bandwidth=1)
lines(fit$x,fit$y,col="red")
fit<-ksmooth(x,y,kernel="box",bandwidth=0.5)
lines(fit$x,fit$y,col="blue")
fit<-ksmooth(x,y,kernel="box",bandwidth=0.25)
lines(fit$x,fit$y,col="green")
fit<-ksmooth(x,y,kernel="box",bandwidth=0.1)
lines(fit$x,fit$y)
```

Example: Normal Kernel Smoothing



Notes: Looks better than box kernel. With normal kernel weights decline smoothly.

```
plot(x,y,ylab="y",xlab="y",main="normal kernel")
fit<-ksmooth(x,y,kernel="normal",bandwidth=1)
lines(fit$x,fit$y,col="red")
fit<-ksmooth(x,y,kernel="normal",bandwidth=0.5)
lines(fit$x,fit$y,col="blue")
fit<-ksmooth(x,y,kernel="normal",bandwidth=0.25)
lines(fit$x,fit$y,col="green")
fit<-ksmooth(x,y,kernel="normal",bandwidth=0.1)
lines(fit$x,fit$y)
```

Spline Smoothing

- We will focus on natural cubic splines.
- Natural cubic splines estimate $f(x)$ by the use of piece-wise cubic polynomials.
- The cubic polynomials that are fitted are subject to "certain constraints."
- Fitting global polynomials of high order is not a good idea!

A introduction to piece-wise cubic polynomials is given by Figure 5.2 in the "Elements of Statistical learning". This figure is shown on the next slide.

Constraints

① pieces are connected

② smooth! the derivatives of endpoints of pieces should be the same.

③ The 2nd derivatives at the connection pts should be the same! (convexity)

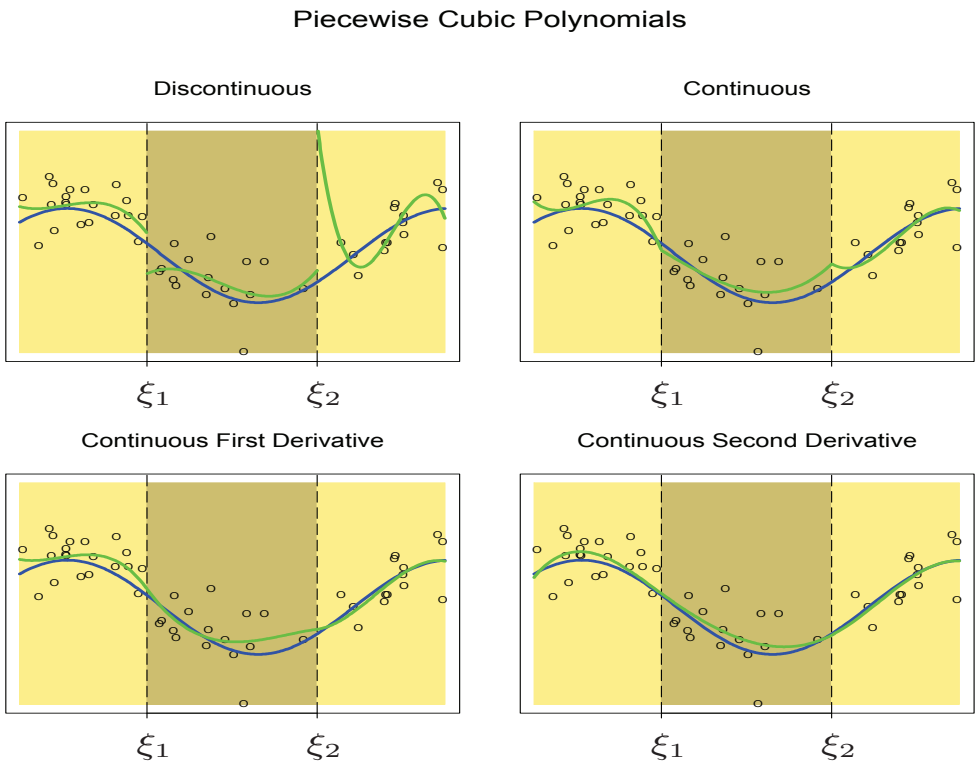


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Natural Cubic Spline Smoothing

- A set of "knots" are chosen.
- Within each region, defined by the knots, a cubic polynomial is fitted.
- At the knots, the polynomials are constrained to be continuous and to have continuous first and second derivatives. [Additionally, beyond the boundary knots linearity is assumed.] see last page.
- The number of knots is the "tuning parameter" for splines. The more knots the less smooth the resulting function.

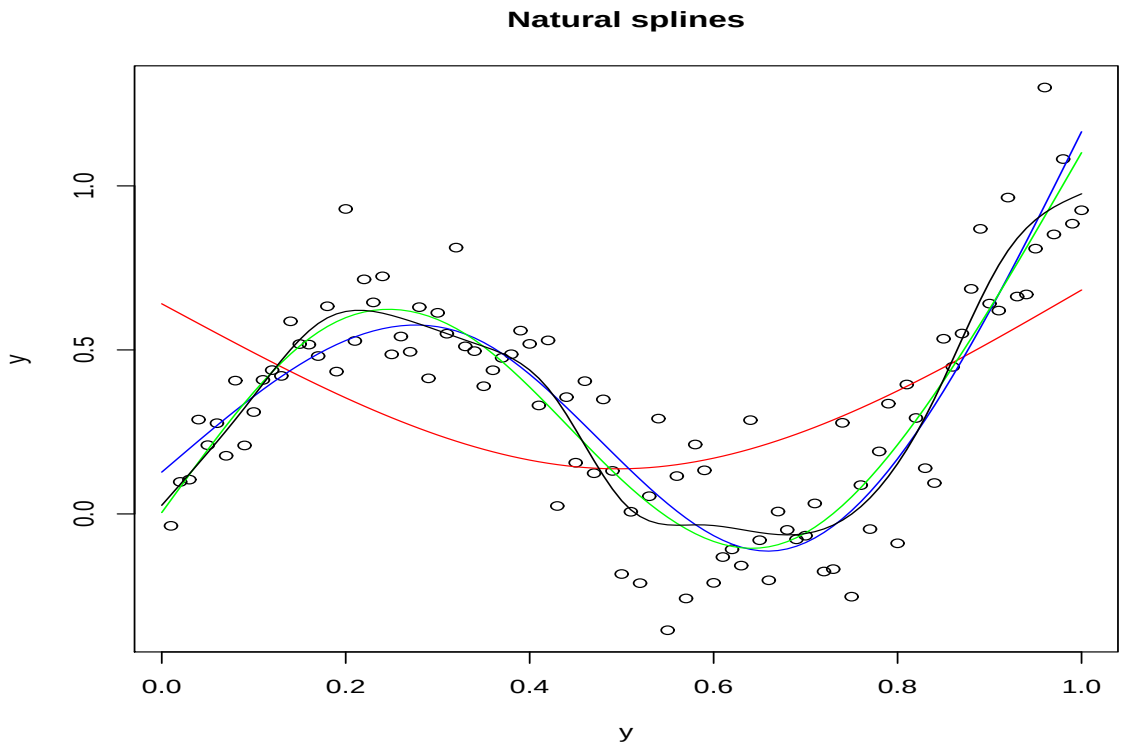
Example: Natural Cubic Splines

```
plot(x,y,ylab="y",xlab="y",main="Natural Splines")
library(splines)
# number of knots = df - 1

fit<-lm(y~ns(x,df=2))
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
fit<-predict(fit,temp)
lines(values,fit,col="red")

fit<-lm(y~ns(x,df=3))
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
fit<-predict(fit,temp)
```

```
lines(values,fit,col="blue")

fit<-lm(y~ns(x,df=6))
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
fit<-predict(fit,temp)
lines(values,fit,col="green")

fit<-lm(y~ns(x,df=12))
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
fit<-predict(fit,temp)
lines(values,fit)
```

dof ↑

# of knots ↑



The other option is to explicitly select the placement of the knots. For example

```
plot(x,y,ylab="y",xlab="y",main="Natural splines")
library(splines)
# Giving exact knot location
```

```
fit<-lm(y~ns(x,knots=c(0.33,0.66)))
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
fit<-predict(fit,temp)
lines(values,fit,col="red")
```

Modelling with Natural Spline Smoothing

- Natural Splines (and other forms of smoothing) are particularly useful for modelling the effect of covariates in a regression analysis.
- If the splines suggest a simple linear or quadratic relationship, then a simpler model can be fitted.
- Below we will see an example using the South African Heart Data that is described in the "Elements of Statistical Learning". This dataset can be obtained from http://www-stat.stanford.edu/ tibs/ElemStatLearn/ A description of the data is also provided (and is shown below):

```
"A retrospective sample of males in a heart-disease high-risk region
of the Western Cape, South Africa. There are roughly two controls per
case of CHD. Many of the CHD positive men have undergone blood
pressure reduction treatment and other programs to reduce their risk
factors after their CHD event. In some cases the measurements were
made after these treatments. These data are taken from a larger
dataset, described in  Rousseauw et al, 1983, South African Medical
```

Journal.

sbp systolic blood pressure
tobacco cumulative tobacco (kg)
ldl low densiity lipoprotein cholesterol
adiposity
famhist family history of heart disease (Present, Absent)
typea type-A behavior
obesity
alcohol current alcohol consumption
age age at onset
chd response, coronary heart disease

To read into R:
read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data",
sep=",",head=T,row.names=1) "

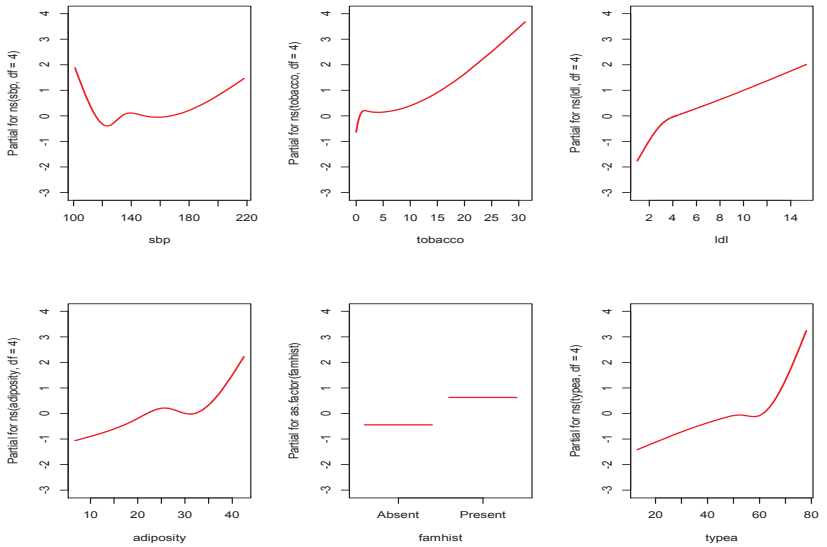
South African Heart Data

- We will model this data using logistic regression.
- The exact form of the relationship between some of the covariates and the response is potentially complicated.
- We will use natural cubic splines to model these potentially complex relationships

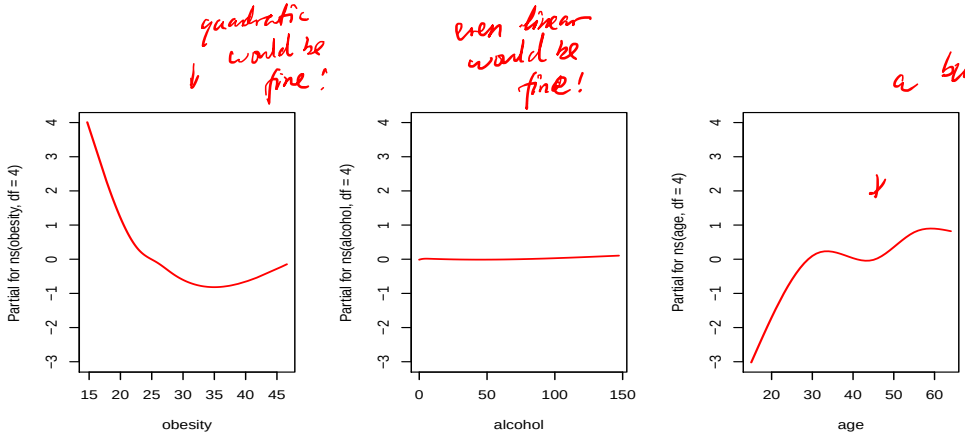
South African Heart Data

```
heart<-read.table("http://www-stat.stanford.edu/
~tibs/ElemStatLearn/datasets/SAheart.data",
sep=",",head=T,row.names=1)
names(heart)
> names(heart)
 [1] "sbp"       "tobacco"   "ldl"       "adiposity" "famhist"   "typea"
 [7] "obesity"   "alcohol"   "age"       "chd"

fit<-glm(chd~ns(sbp,df=4)+ns(tobacco,df=4)+
ns(ldl,df=4)+ns(adiposity,df=4)+as.factor(famhist)+
ns(typea,df=4)+ns(obesity,df=4)+ns(alcohol,df=4)
+ns(age,df=4),family="binomial")

par(mfrow=c(2,3))
termplot(fit)
```

South African Heart Data



South African Heart Data



The plots above suggest that linear or quadratic terms are probably appropriate for modelling the effects of each variable. Below we refit our model and do some very ad-hoc model selection.

South African Heart Data

```
fit1<-glm(chd~sbp+I(sbp^2)+tobacco+ldl+adiposity
+as.factor(famhist)+typea+obesity+I(obesity^2)
+alcohol+age+I(age^2),family="binomial")
summary(fit1)
> summary(fit1)

Call:
glm(formula = chd ~ sbp + I(sbp^2) + tobacco + ldl + adiposity +
as.factor(famhist) + typea + obesity + I(obesity^2) + alcohol +
age + I(age^2), family = "binomial")

Deviance Residuals:
Min       1Q   Median       3Q      Max
-1.8008  -0.7857  -0.4125   0.8504   2.4473

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept)                7.1889033  5.2489532   1.370 0.170816
sbp                       -0.0831110  0.0545694  -1.523 0.127751
I(sbp^2)                   0.0002937  0.0001790   1.641 0.100884
tobacco                    0.0805346  0.0272292   2.958 0.003100 **
```

```
ldl                          0.1868630  0.0606021   3.083 0.002046 **
adiposity                    0.0369953  0.0310429   1.192 0.233361
as.factor(famhist)Present    0.9595617  0.2314061   4.147 3.37e-05 ***
typea                        0.0394953  0.0124809   3.164 0.001554 **
obesity                     -0.5754962  0.2418426  -2.380 0.017330 *
I(obesity^2)                 0.0087517  0.0039770   2.201 0.027768 *
alcohol                      0.0013104  0.0045619   0.287 0.773924
age                          0.0452748  0.0122008   3.711 0.000207 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 464.58  on 450  degrees of freedom
AIC: 488.58

Number of Fisher Scoring iterations: 5

#some very rough model selection leads to fit2
fit2<-glm(chd~tobacco+ldl+as.factor(famhist)
+typea+obesity+I(obesity^2)
+age,family="binomial")
> summary(fit2)

Call:
```

```
glm(formula = chd ~ tobacco + ldl + as.factor(famhist) + typea +
obesity + I(obesity^2) + age, family = "binomial")

Deviance Residuals:
Min       1Q   Median       3Q      Max
-1.9292  -0.7907  -0.4294   0.8780   2.3935

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept)                -0.514835   2.914800  -0.177 0.859801
tobacco                     0.081354   0.026187   3.107 0.001892 **
ldl                         0.199392   0.059227   3.367 0.000761 ***
as.factor(famhist)Present   0.925414   0.227819   4.062 4.86e-05 ***
typea                       0.037843   0.012276   3.083 0.002051 **
obesity                    -0.424623   0.206184  -2.059 0.039452 *
I(obesity^2)                0.006812   0.003575   1.905 0.056732 .
age                         0.054138   0.010295   5.259 1.45e-07 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 470.37  on 454  degrees of freedom
AIC: 486.37
```

```
Number of Fisher Scoring iterations: 5
```

Knot Selection

- How can the number of knots be chosen?
- A common method is to choose the number of knots to minimize the AIC or BIC.
- The AIC for a particular model is defined as: $-2ln(L) + 2 \times p$.
- The BIC for a particular model is defined as: $-2ln(L) + log(n) \times p$.
- Both of these measures combine a measure of the goodness of fit of the model with a penalty for the number of parameters in the model.

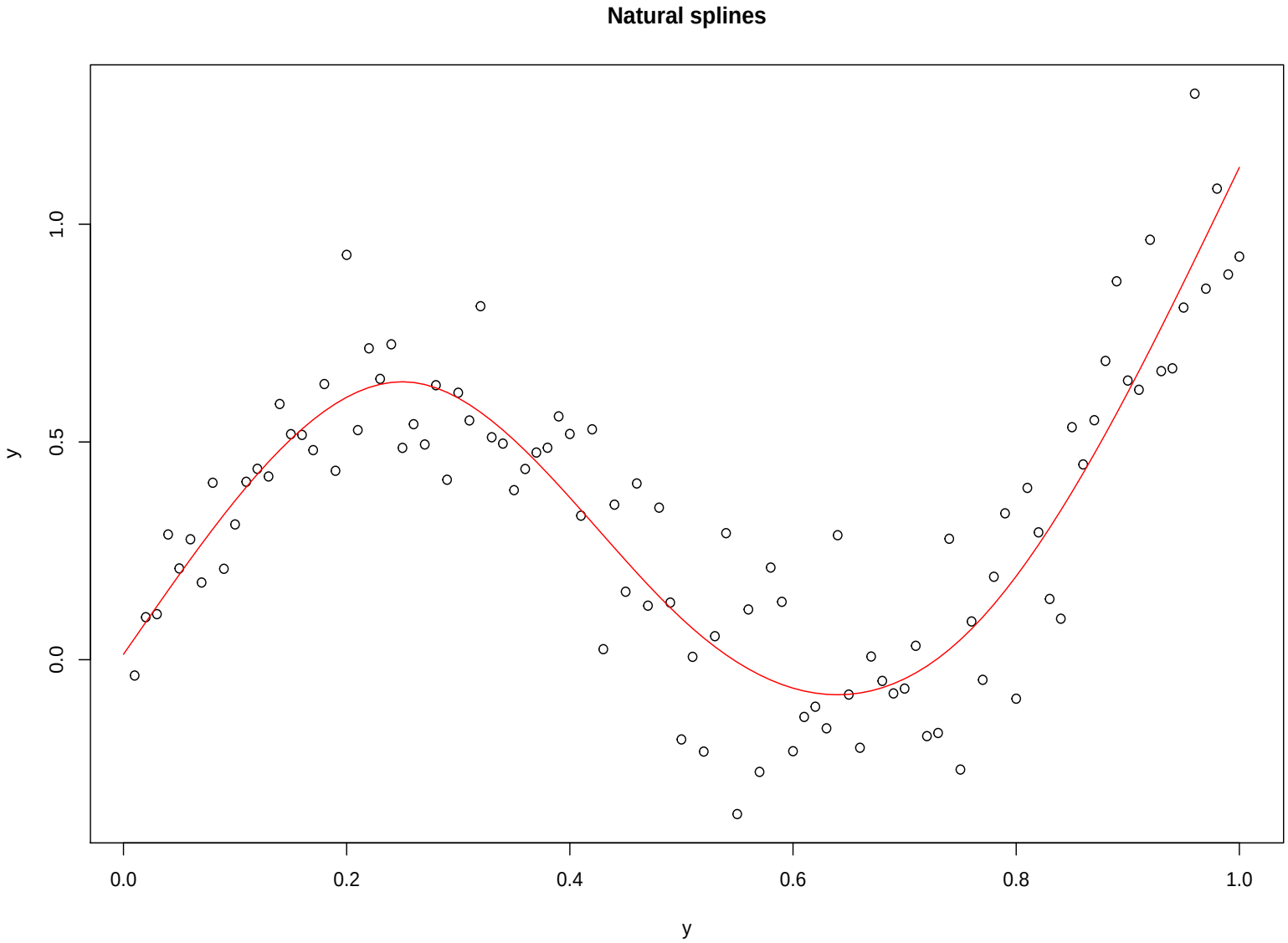
Example: Knot Selection

Using the motivating data from above. We will use the AIC to choose the optimal number of knots.

```
library(splines)

for(i in 1:20) {
fit<-lm(y~ns(x,df=i))
print(c(i,AIC(fit)))
}

#minimum when df=4

plot(x,y,ylab="y",xlab="y",main="Natural splines")
values<-seq(0,1,0.01)
temp<-data.frame("x"=values)
```

```
fit<-predict(fit,temp)
lines(values,fit,col="red")
```

The plot below shows a plot of the spline fitted using the optimal number of knots.

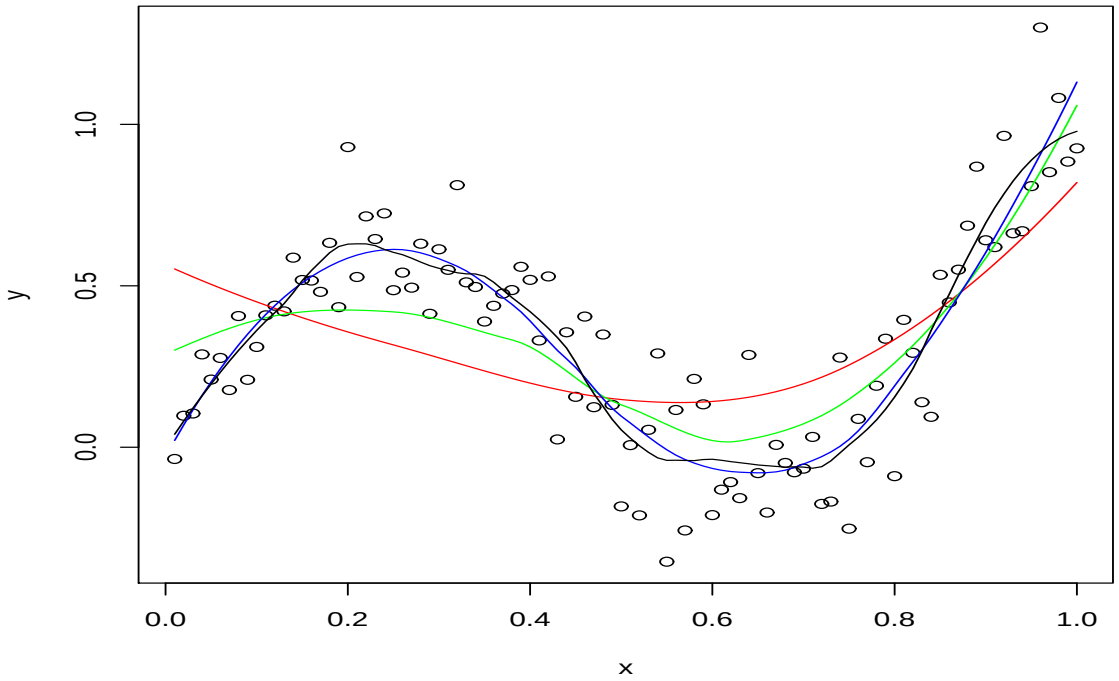

Natural splines
y
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
y

Other Smoothers - Local Polynomial Regression

locally weighted scatterplot smoothing

- In R the function "loess" implements local polynomial fitting.
- Using points that are "close" (in the neighborhood of) to the point of interest x^* a (weighted) polynomial is fitted. The smooth value at x^* is given by the fitted value from this polynomial.
- For loess the tuning parameter is the "span". The span controls the proportion of the points that influence the smooth at x^*.

Example: Loess




```
plot(x,y)
fit <- loess(y ~ x, span=2, degree=2)
```

```
lines(x,fit$fitted,col="red")
fit <- loess(y ~ x, span=1, degree=2)
lines(x,fit$fitted,col="green")
fit <- loess(y ~ x, span=0.5, degree=2)
lines(x,fit$fitted,col="blue")
fit <- loess(y ~ x, span=0.25, degree=2)
lines(x,fit$fitted)
```

looks smooth & fits well.