Regression For non-linear functions

2025-02-08

Overview

- in many situations our variable of interest does not have a straight line relationship with the outcome of interest (Y)
- when the data sets are moderate in size (say a hundred for linear regression and a thousand for logistic and other generalized linear models) the assumption of linearity becomes problematic
- there are a number of proposed solutions, piecewise linear models, polynomials (almost always a bad idea) and splines
- in this lecture I want to talk about splines and why they are a useful tool

Typical models in epidemiology and clinical trials

- we often want to model the prevelance of a disease, or risk for relapse in our models
- we generally want to account for important risk factors, and adjust for other variables (age, sex, BMI, pack years) that may have a big effect on the outcome, and our goal is to reduce variability due to these variables
- it is very unlikely that age has a linear relationship on pretty much any outcome, frailty tends to accumulate quickly in the elderly, while for those under 50 it remains stable over decades
- one of the bad things that happens when you model age (or many other continuous covariates as linear functions in your models) is that the lack of fit caused by inappropriate model choice will manifest as identifying covariates as related to the outcome when they are not

A simulation

- in this lecture we will explore how this can happen, and how you can prevent it
- my personal preference is natural splines, fit with the **ns** function in the splines package
- there are other choices, but this is pretty simple to both fit and interpret

Simulation

Here we are going to simulate some data from a non-linear model.
Just for ease of explanation, let's think of x as representing age and y will be some continuous measurement, say levels of serum cholestorol (in some unspecified units).

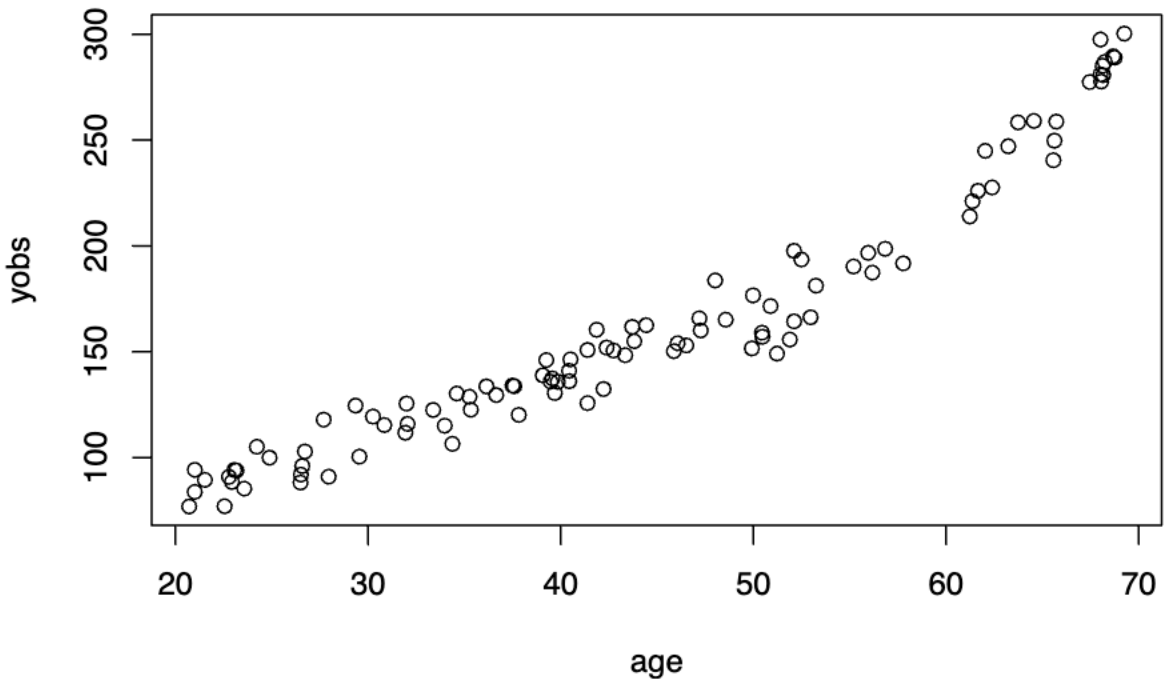
```
y = function(x) {
  y= ifelse(x < 60, y=20+3*x, 200+10*(x-60))
  return(y)
}

##set a seed, so that the random number generation is the same for every iteration
set.seed(333)

age = runif(100, min=20, max=70)
#we need a big-ish sd to get variation around the line
```

```r
#what happens if we increase the sd below?
yobs = y(age) + rnorm(100,sd=10)

plot(age,yobs)
```



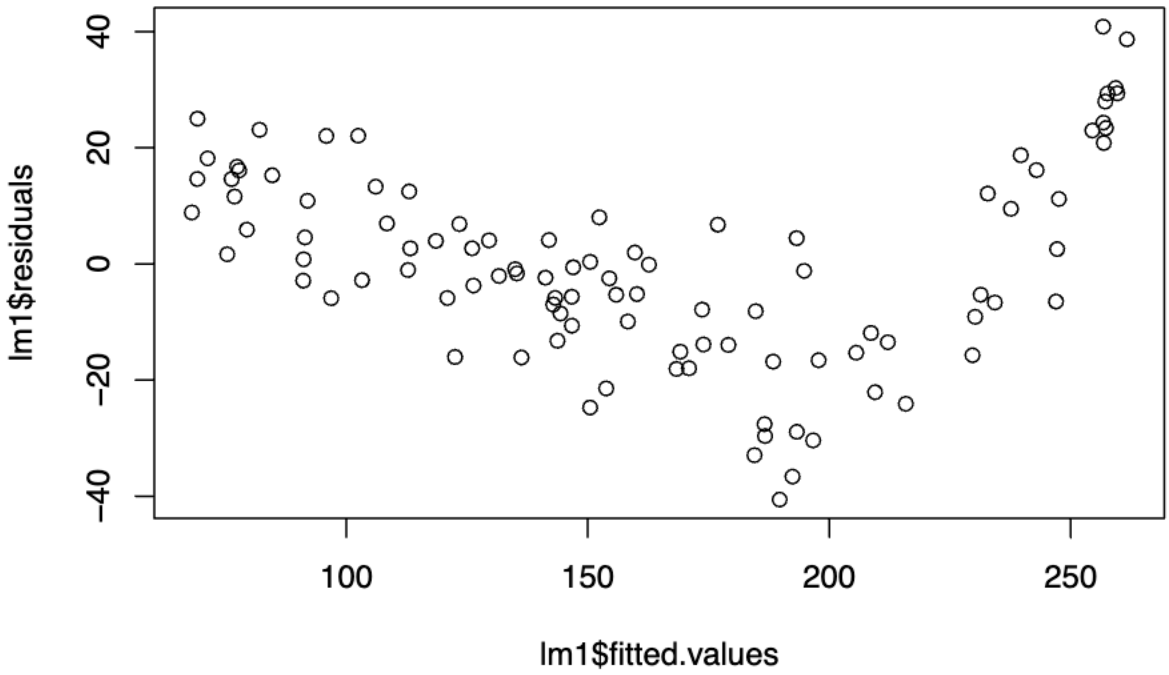

```r
lm1 = lm(yobs~age)
summary(lm1)
```

```
##
## Call:
## lm(formula = yobs ~ age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.588 -10.953  -0.984  11.725  40.856
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -14.678      5.463  -2.687  0.00847 **
## age            3.990      0.117  34.094  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.96 on 98 degrees of freedom
## Multiple R-squared:  0.9222, Adjusted R-squared:  0.9215
## F-statistic:  1162 on 1 and 98 DF,  p-value: < 2.2e-16
```

This model appears to fit well. The Multiple R-squared value is high, both coefficients seem to be significant. But...

And we can look at the residuals from this model to see what we thing about the goodness of fit.

```r
plot(lm1$fitted.values, lm1$residuals)
```



So now we see that there is a pattern in the residuals that might cause us concern. What can go wrong in this with a linear model?

- our predictions are going to be wrong for many of the values of x
- if we examine other variables for significance we will find associations where none exists, and the only reason they appear interesting is because they fix the defects in our model.

You can see that the residuals are large for large values of our predictions (the fitted values). And these are associated with large values of x due to the nature of the model. So suppose there is some other variable z where large values of x associate with large values of z. Example, prostate cancer affects men over 65 (think of x as age here), and taking a medicine for hypertension is also associated with age, older people are more likely to need medication.

So next we simulate whether someone takes a hypertensive medication. For people under 60, we will assume only about 10% take a medicine and for people over 60 it will be 50%.

```
onmeds=rep(NA,length(age))
for(i in 1:length(age)) {
  if(age[i]<60) p=.1 else p=.7
  onmeds[i] = rbinom(1,1,prob=p)
}
##onmeds = ifelse(x<50, rbinom(1,1,prob=.1), rbinom(1,1,prob=.5))
table(onmeds)
```

```
## onmeds
##  0  1
## 80 20
```

```
table(onmeds, age<60)
```

```
##
## onmeds FALSE TRUE
##      0     7   73
##      1    14    6
```

We generated this `onmeds` variable without any regard to our response, so it is independent of the response

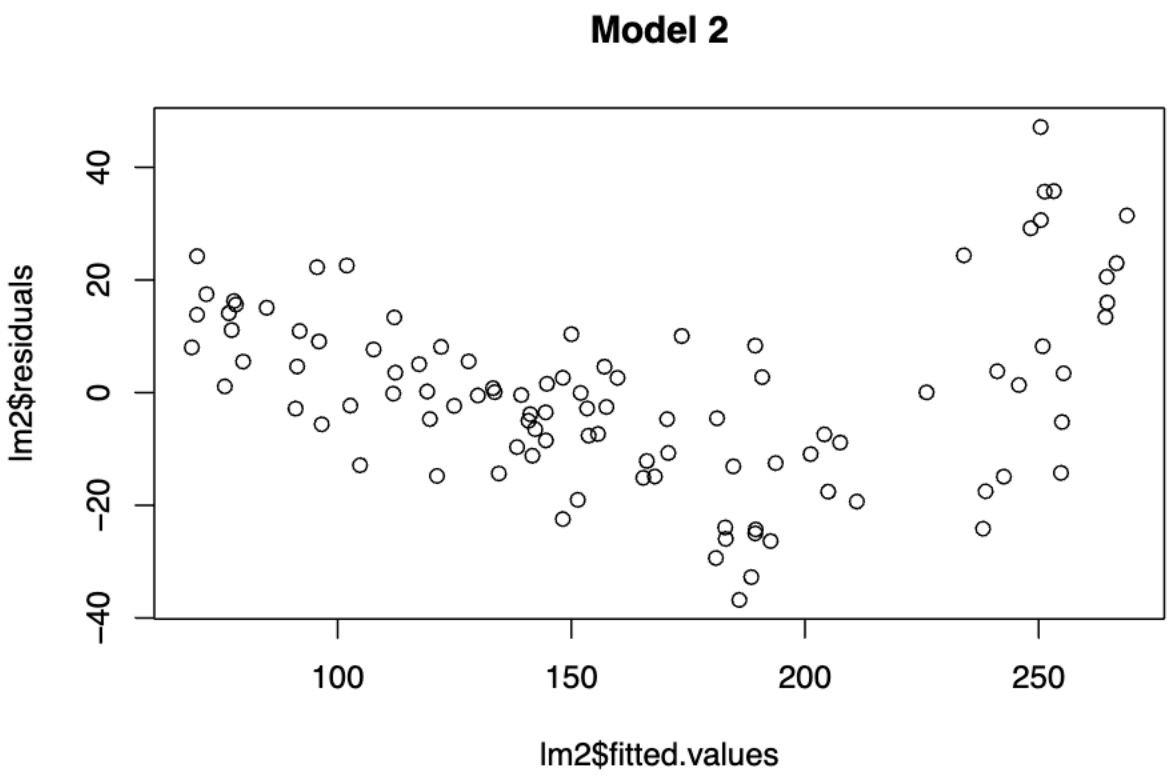
and there should be no relationship between it and our response.

```
lm2 = lm(yobs~age+onmeds)
summary(lm2)
```

```
##
## Call:
## lm(formula = yobs ~ age + onmeds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.811 -10.984  -0.115   9.326  47.164
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.7335     5.3909  -1.991  0.04929 *
## age           3.8396     0.1223  31.386  < 2e-16 ***
## onmeds       13.7197     4.4319   3.096  0.00257 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.26 on 97 degrees of freedom
## Multiple R-squared:  0.9292, Adjusted R-squared:  0.9278
## F-statistic: 636.9 on 2 and 97 DF,  p-value: < 2.2e-16
```

And again, the residual plots is fairly informative, there remain issues with the fit.

```
plot(lm2$fitted.values, lm2$residuals, main="Model 2")
```



And here we see that the model seems to be good, we have significant effects for both age, which is really associated with response, and for our made up variable, that does not. These kinds of associations are quite common, especially in large data sets. And their impact is becoming more substantial and can lead to

incorrect conclusions.

One of the problems is that as data sets get larger then the impact of our modeling assumptions will change. In some cases issues that exist with small data sets will be lessened, but in other cases the impacts will increase. Fitting straight line models to continuous data, especially health related data is almost always incorrect. But that doesn't matter so much when you only have a hundred or so data points. When you have more than a thousand then this problems can occur (and of course the can occur at smaller numbers).

We can alleviate the problem but fitting more flexible models. And a reasonable choice is to fit natural splines. The **splines** package implements natural splines using the function **ns**. The function generates a basis matrix to represent piecewas-cubic splines. You can add flexibility through either increasing the degrees of freedom (**df**), or by specifying the number and location of the *knots*. In general you should look at the fit and assess whether or not you think you have enough flexibility. If the underlying function you are trying to approximate has a lot of variation in it, you will need more **df**. For most relationships in epidemiology, like those with age, BMI, height etc the underlying relationships are often quite smooth.

```
lm3 = lm(yobs~ ns(age,df=5) + onmeds)
summary(lm3)
```

```
##
## Call:
## lm(formula = yobs ~ ns(age, df = 5) + onmeds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.0090  -5.5355   0.1207   5.0871  23.5097
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        83.9177     3.6619  22.917   <2e-16 ***
## ns(age, df = 5)1   54.4336     4.7019  11.577   <2e-16 ***
## ns(age, df = 5)2   80.2160     6.5554  12.237   <2e-16 ***
## ns(age, df = 5)3 107.0379     6.4259  16.657   <2e-16 ***
## ns(age, df = 5)4 206.3211     9.7248  21.216   <2e-16 ***
## ns(age, df = 5)5 196.0401     4.3048  45.539   <2e-16 ***
## onmeds             0.5783     2.8613   0.202     0.84
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.46 on 93 degrees of freedom
## Multiple R-squared:  0.977,  Adjusted R-squared:  0.9756
## F-statistic: 659.7 on 6 and 93 DF,  p-value: < 2.2e-16
```

We can compare the spline model, **lm3** to the linear model using standard statistical methods. The linear coefficient is a submodel of the spline model so one can simply use the **anova** method. The **ns** model, **lm3** contains (in the sense that it could be identical to) **lm2** and so the inference is valid.

```
##use LRT to compare models
anova(lm2, lm3, test ="LRT")
```

```
## Analysis of Variance Table
##
## Model 1: yobs ~ age + onmeds
## Model 2: yobs ~ ns(age, df = 5) + onmeds
##   Res.Df     RSS Df Sum of Sq  Pr(>Chi)
## 1     97 25654.3
## 2     93  8322.2  4     17332 < 2.2e-16 ***
```

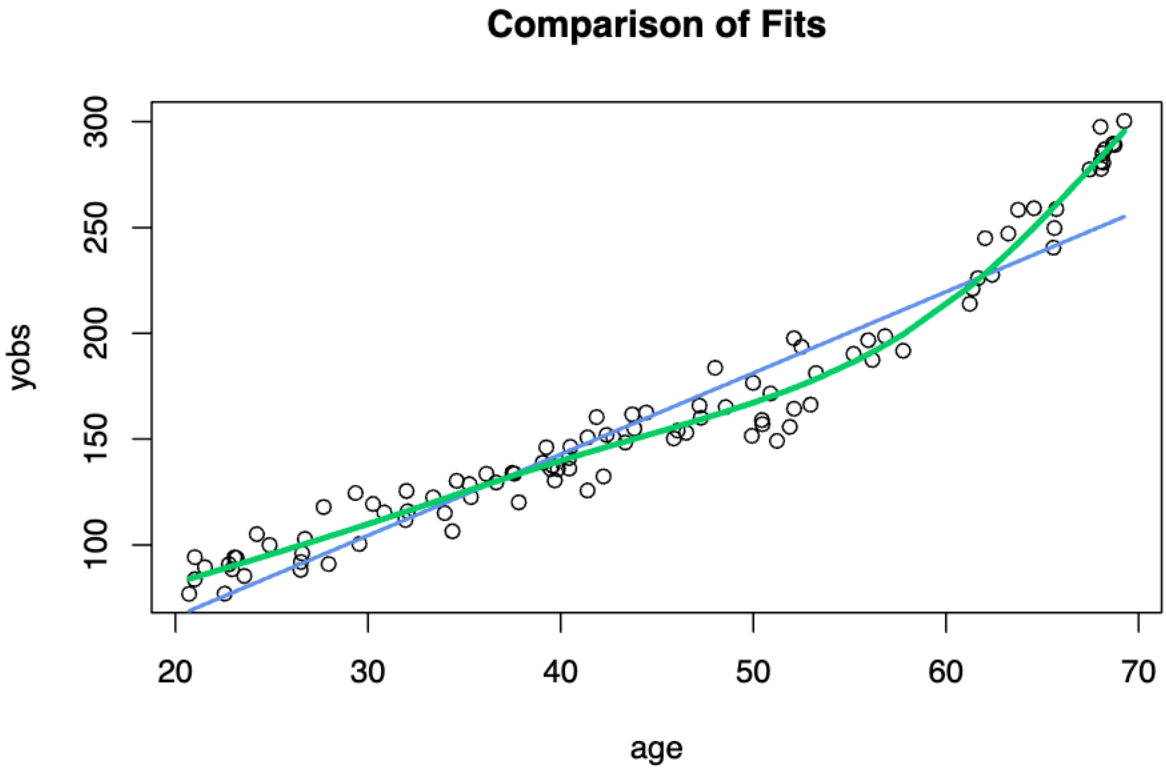
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we see that the model fits much better - the multiple R squared is high, and `onmeds` is no longer significant. We will need to do some work to better understand the output.

We can examine the fitted spline together with the regression line from model `lm2` above. We use the predict function, for the spline model we need to sort age to get the plot to look right. Alternatively you can just generate a sequence of `x` values and use that. You do need to specifiy a value of `onmeds` in order to get a prediction.

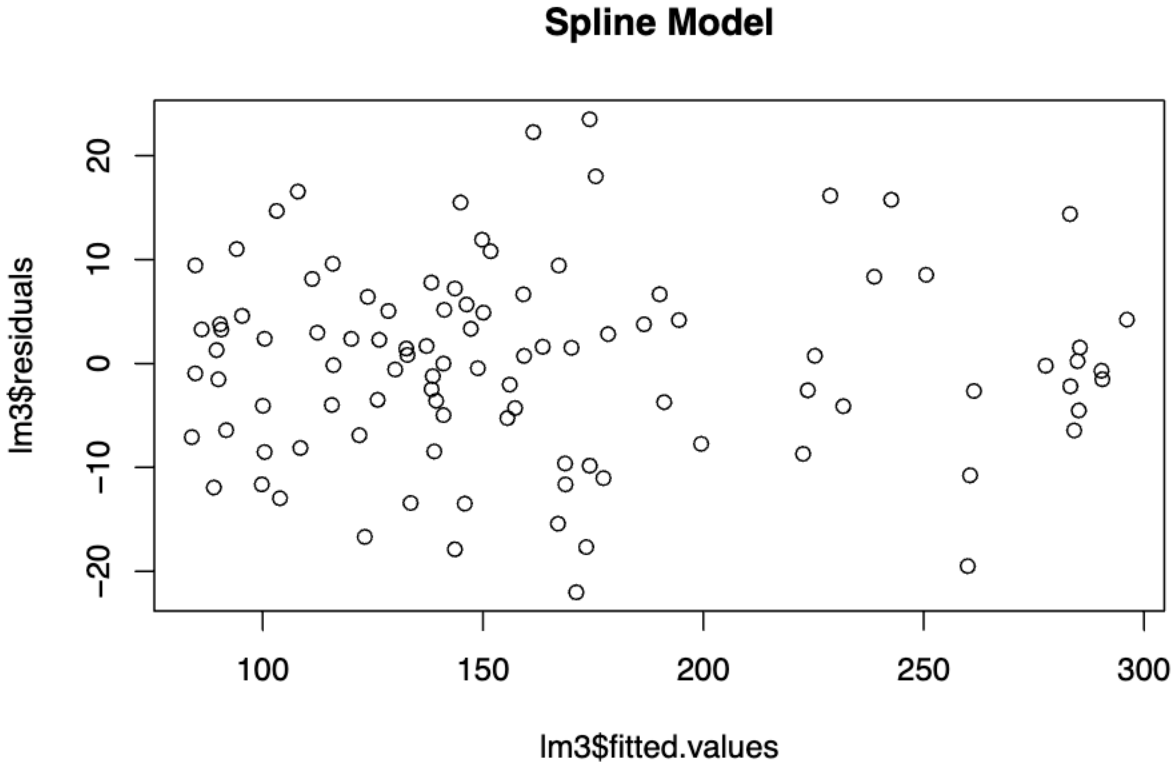
As an exercise see if you can figure out how to add prediction intervals.

```
plot(age, yobs, main="Comparison of Fits")
coefs2 = lm2$coefficients
prlm2 = predict(lm2,newdata=data.frame(age=age, onmeds=0))
lines(age, prlm2, lwd=2, col="cornflowerblue")
prlm3 = predict(lm3,newdata=data.frame(age=sort(age), onmeds=0))
lines(sort(age), prlm3, lwd=3, col="springgreen3")
```



And the plot of residuals against fitted values now looks much better.

```
plot(lm3$fitted.values, lm3$residuals, main="Spline Model")
```


Spline Model
lm3$residuals
20
10
0
−10
−20
100
150
200
250
300
lm3$fitted.values