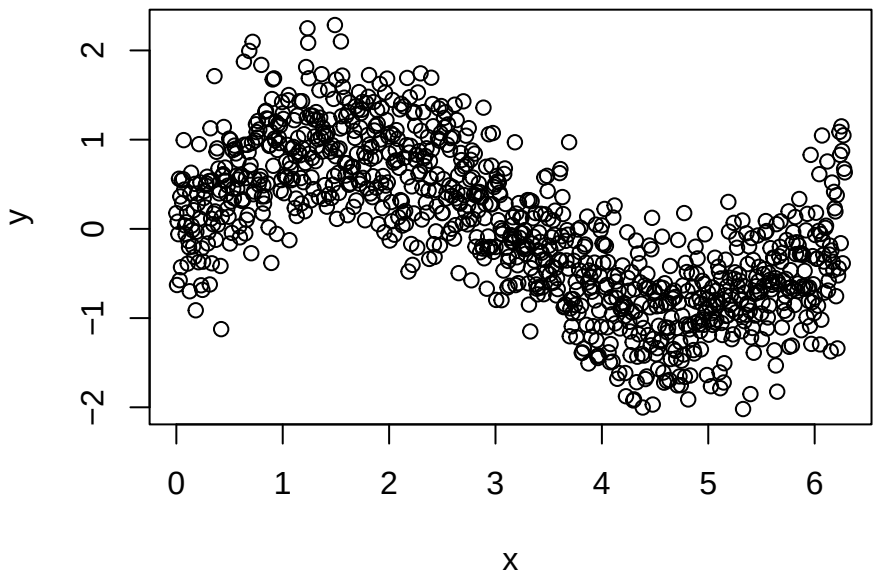
Working with splines in R

September 24, 2024

Here we provide a basic introduction to splines-fitting in R. We will use our own code to fit the models, rather than `mgcv`. We start by simulating some data and looking at it:

```
n <- 1e3
x <- seq(0, 2*pi, length.out = n)
y <- sin(x) + rnorm(n, 0, 0.5)
dat <- data.frame(x = x, y = y)
plot(x, y)
```

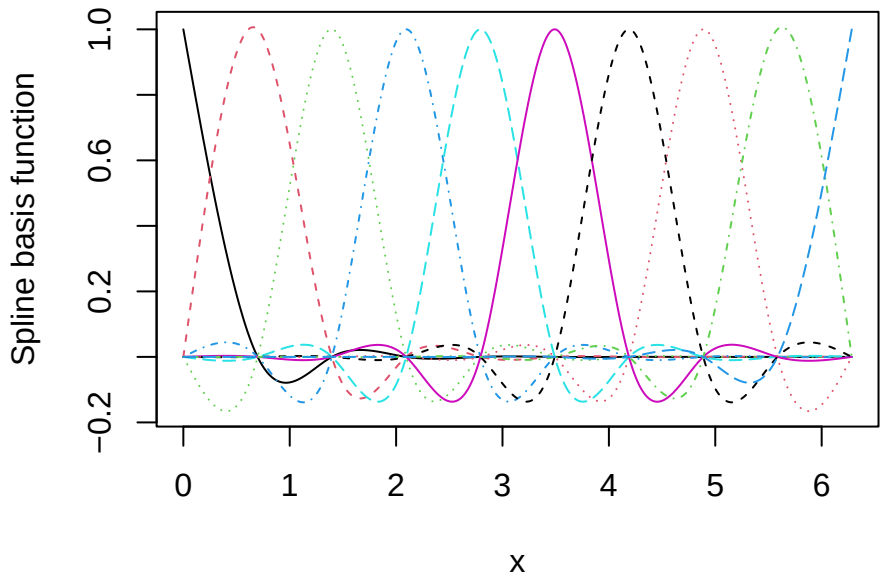


To models this data with splines, we first need to build the spline model matrix $\mathbf{X}$. We can do this with `mgcv`, via the `smoothCon` function. The function outputs a list, we will find the evaluated basis fuctions in the `$X` element of the list:

```
library(mgcv)
sm <- smoothCon(s(x, k = 10, bs = "cr"), data=dat, knots=NULL)[[1]]

X <- sm$X

matplot(X, x = x, type = 'l', ylab = "Spline basis function")
```



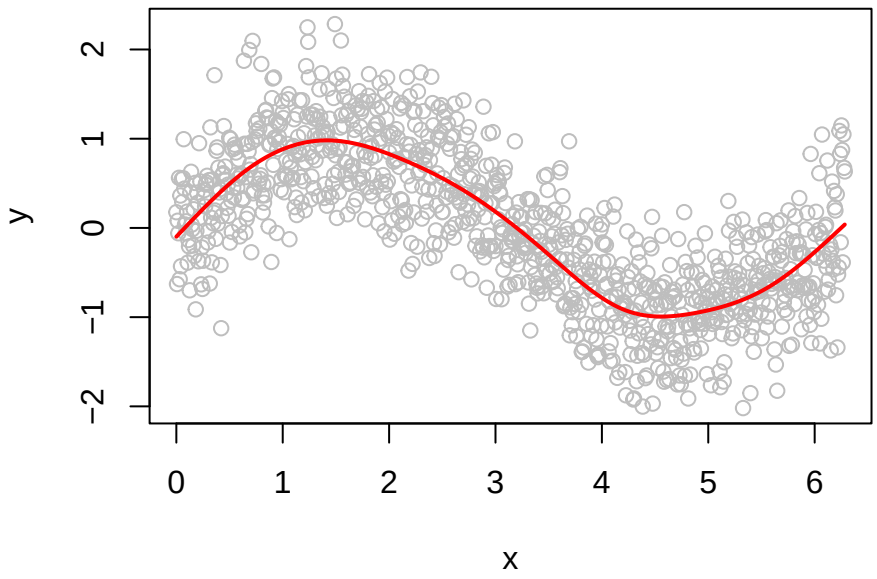
Above we are using 10 cubic regression splines basis functions. We can fit the corresponding regression model by least-squares, by using the `lm` function:

```
fit <- lm(y ~ X - 1, data = dat)

coef(fit)

##          X1          X2          X3          X4          X5          X6
## -0.09842079  0.68214881  0.98128403  0.78287406  0.35188979 -0.28931328
##          X7          X8          X9         X10
## -0.90753118 -0.95106981 -0.64961126  0.03796254

plot(x, y, col = "grey")
lines(x, X %*% coef(fit), type = 'l', col = "red", lwd = 2)
```

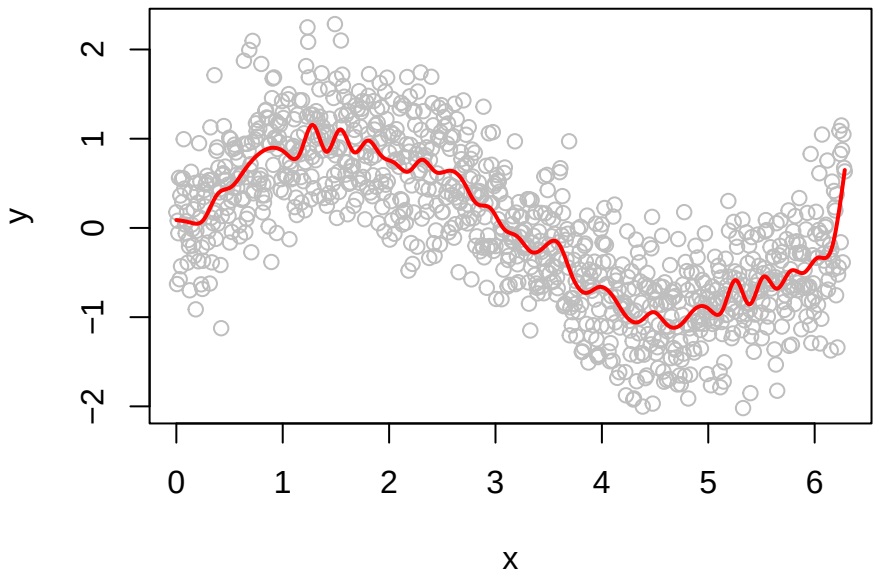


Note that `-1` is used to remove the intercept from the linear model. We do this because a columns of 1s is already in the span of the columns of the model matrix $\mathbf{X}$, hence adding a intercept (a column of ones to the model matrix) would lead to an unidentifiable model. The fit looks good here, but it depends on the number of basis functions used. Increasing k to 50 leads to overfitting:

```
sm2 <- smoothCon(s(x, k = 50, bs = "cr"), data=dat, knots=NULL)[[1]]
X2 <- sm2$X

fit2 <- lm(y ~ X2 - 1, data = dat)

plot(x, y, col = "grey")
lines(x, X2 %*% coef(fit2), type = 'l', col = "red", lwd = 2)
```



To address the overfitting, we can add a smoothing penalty that will be added to the sum-of-squares loss. That is we can fit the model by minimising:

$$\text{pen\_ssqr}(\boldsymbol{\beta}|\lambda) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$

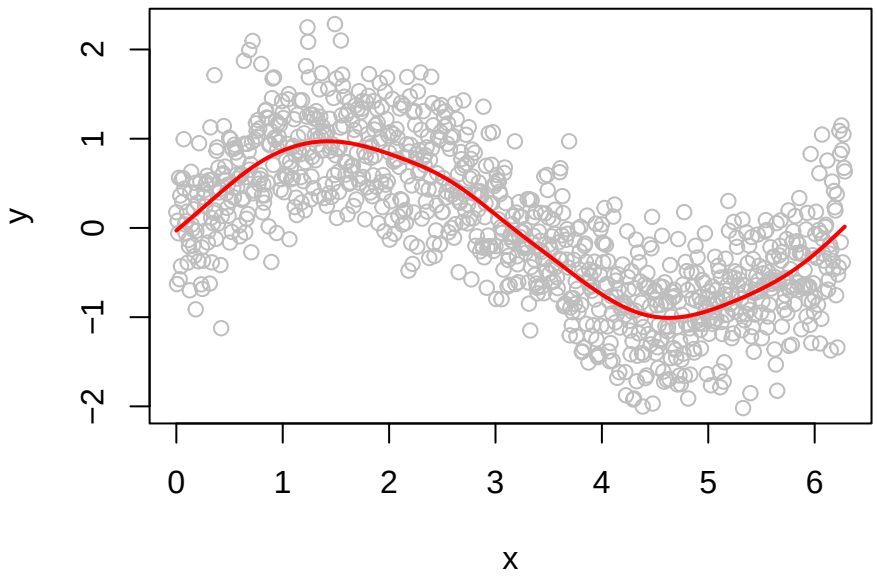
where $\mathbf{S}$ is a penalty matrix. We can find $\mathbf{S}$ in the `$S` element of the output of `smoothCon`. Here is our new objective function:

```
pen_ssqr <- function(beta, X, y, S, lambda){
  mu <- X %*% beta
  pen <- t(beta) %*% S %*% beta
  loss <- sum((y - mu)^2) + lambda * pen
  return(loss)
}
```

which we minimise here using `optim` (and via the BFGS optimiser):

```
fit_pen <- optim(par = fit2$coef, fn = pen_ssqr, y = y, X = X2,
                 S = sm2$S[[1]], lambda = 10000, method = "BFGS")

plot(x, y, col = "grey")
lines(x, X2 %*% fit_pen$par, type = 'l', col = "red", lwd = 2)
```



Here the fit looks good again. But note that the value `lambda = 10000` is hand-selected, so it would be much preferable to have an automatic method to select it.