样条

林双全

2016 年 *12* 月 *11* 日

在日常数据的处理中，经常需要对数据进行拟合，例如探究因变量与自变量之间的关系，比较简单的方法就是用多项式回归进行拟合. 但是在实际生活中如此好单纯不做作的数据实在是可遇而不可求的，这时可以用一些非参数方法去估计和拟合. 常用的方法有核方法、局部多项式、样条方法等等。

| 在介绍样条回归之前简要回顾下什么是分段线性回归，就是在不同的区间上有着不同的线性回归形式：

$$y = \begin{cases} \alpha_1 + \beta_1 x + \epsilon_1, & x \in [t_0, t_1] \\ \alpha_2 + \beta_2 x + \epsilon_1, & x \in [t_1, t_2] \\ \quad \vdots & \\ \alpha_k + \beta_k x + \epsilon_1, & x \in [t_{k-1}, t_k] \end{cases}$$

其中的$t_1, t_2, \ldots, t_{k-1}$ 就被称为节点 (knots).

而样条方法 (spline)，就是用分段的低阶多项式去逼近函数，简单地来说就是在节点 (knots) 光滑的分段回归，常用的样条有多项式样条 (Polynomial spline) 和光滑样条 (Smoothing spline). 多项式样条的样条基很多，其中最为常用的是 B-spline basis(B 样条基) 和 truncated power basis(截断幂基), 今天要介绍的就是 B-spline 和 P-spline。

B-spline

概念

| B 样条 (B-spline,Basic-spline) 由 Isaac Jacob Schoenberg 于 1946 年提出来，是贝茨曲线 (Bezier Curve) 的一种一般化。其定义为：给定满足$t_0 < t_1 < \cdots < t_m$ 的 m+1 个节点，一个 p 次 B 样条是由 p 次 B 样条基组成：

$$f(t) = \sum_{i=0}^{m} P_i b_i^p(x)$$

其中P_i 被称为控制点或 de Boor 点，更一般地表示成

$$f(x) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + \sum_{j=1}^{m} \beta_{pj} b_j^p(x)$$

$b_i^p(t)$ 称为即为 B 样条基，可由 Cox-de Boor 递归公式定义：

$$b_i^0(x) = \begin{cases} 1, & t_i \le x \le t_I + 1 \\ 0, & \end{cases}$$

$$b_i^p(x) = \frac{x - t_i}{t_{t+p} - t_i} b_i^{p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} b_{i+1}^{p-1}(x)$$

| B 样条样条要求在每个节点直到 p-1 阶导数都是连续的。实际生活中，最常用的就是三次 B 样条去拟合数据。

| 当节点等距时，称 B 样条是均匀 (uniform) 否则就是非均匀的。可以证明的是，在 B 样条是均匀的时候且在同样的 p 次下，用截断幂基作为基函数得到的结果与用 B 样条基没有显著差异的，因此 B 样条也可以表示为

$$f(x) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + \sum_{k=1}^{m} \beta_k (x - t_k)_+^p$$

其中$(x-a)_+^p = \{max(0, z-a)\}^p$ 是截断幂函数。

节点个数和位置都对 B 样条的估计结果有影响，需要在多个节点但会引起曲线技术变化和选择较少的节点令函数变得更平稳之间做出权衡。实践证明，令节点在数据上呈现均匀分布是一种比较有效的节点选择方式，其一种实现方式是：首先确定需要的自由度，然后依靠软件自动在数据的均匀分位数上设置相应个数的节点。而节点的个数选择一种方法是尝试多个不同的节点个数，然后从中选择拟合“形状最理想”的曲线，另一种较为客观的方式就是使用交叉验证。

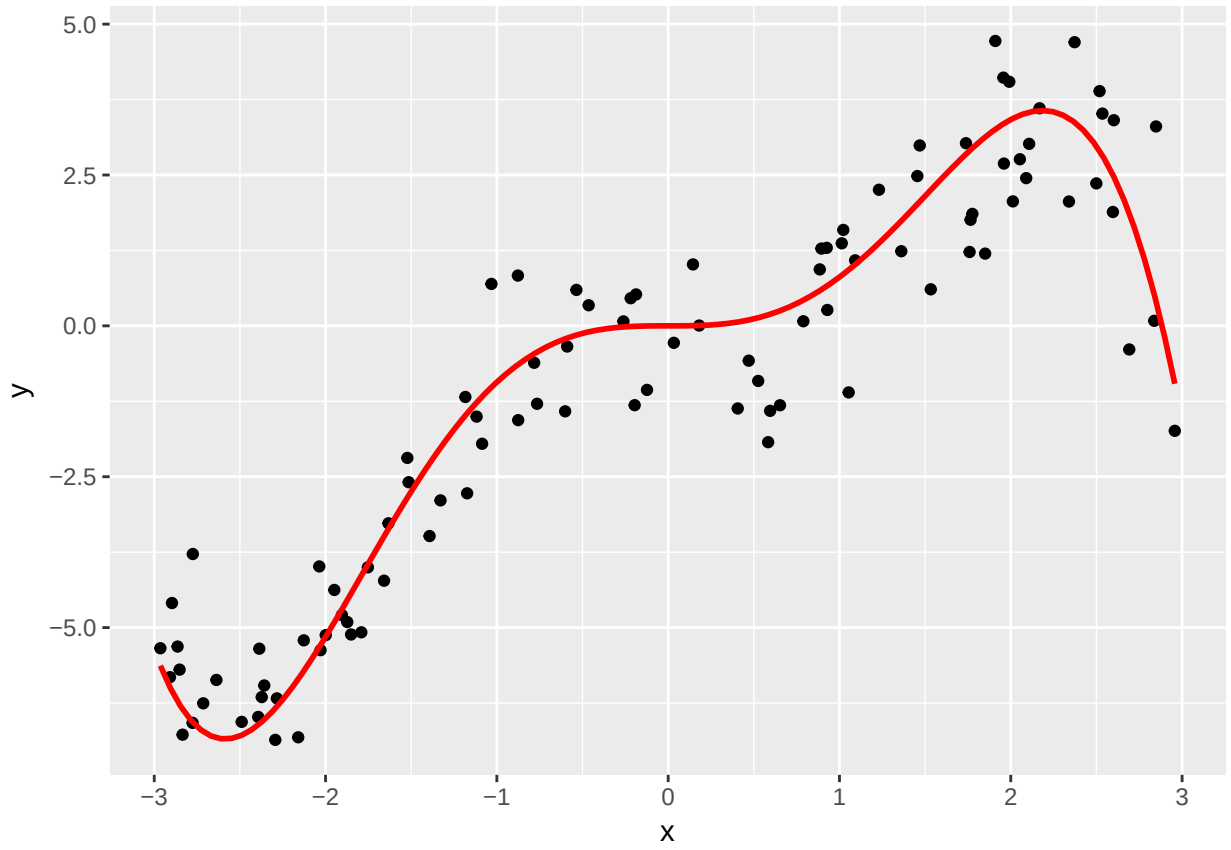
B 样条的系数用最小二乘法估计。关于 B 样条的公式推导、更多性质，以及其推广形式非均匀 B 样条，有兴趣的同学可以查阅 DeBoor(1978) 的 APractical Guide to Spline 和 Paul(1995) 的 Curve and Surface Fitting with Splines 等著作。

程序实现

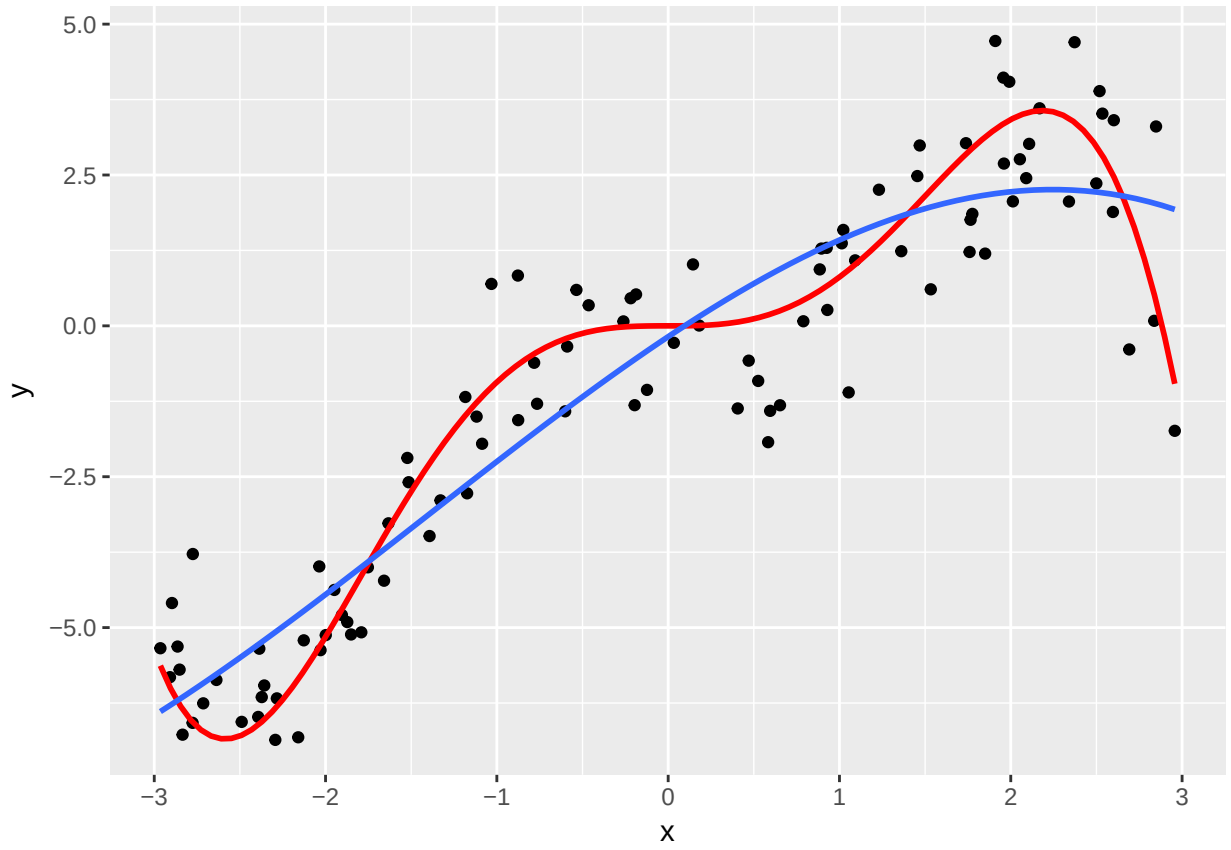
| 在 R 中 spline 包中的 bs() 函数提供了 B 样条估计, 其调用格式为：
bs(x, df = NULL, knots = NULL, degree = 3, intercept = FALSE, Boundary.knots = range(x)) - x：自变量
- df：自由度，df=degree+(节点个数)
- knots: 节点位置，可以自己指定
- degree：分段多项式的次数，即上文的 p，默认为 3
- intercept：默认为 FALSE，如果取 TRUE，则在基中包含截距项
- Boundary.knots：B 样条的边界节点，默认用 range(x)

以下用一个例子说明：

```
library(splines)
library(data.table)
library(ggplot2)
# 实际函数
s = function(x) {
  (x^3) * sin((x + 3.4)/2)
}
# 以实际函数生成一份随机数据
v <- rnorm(100,0,1)
dat <- data.table(x <- sort(runif(100,min = -3,max = 3)),y <- s(x)+v)
attach(dat)
# 数据的散点图
p <- ggplot(dat,aes(x,y))+geom_point()
p <- p+stat_function(fun = s,color="red",size=1)
p # 实际曲线
```

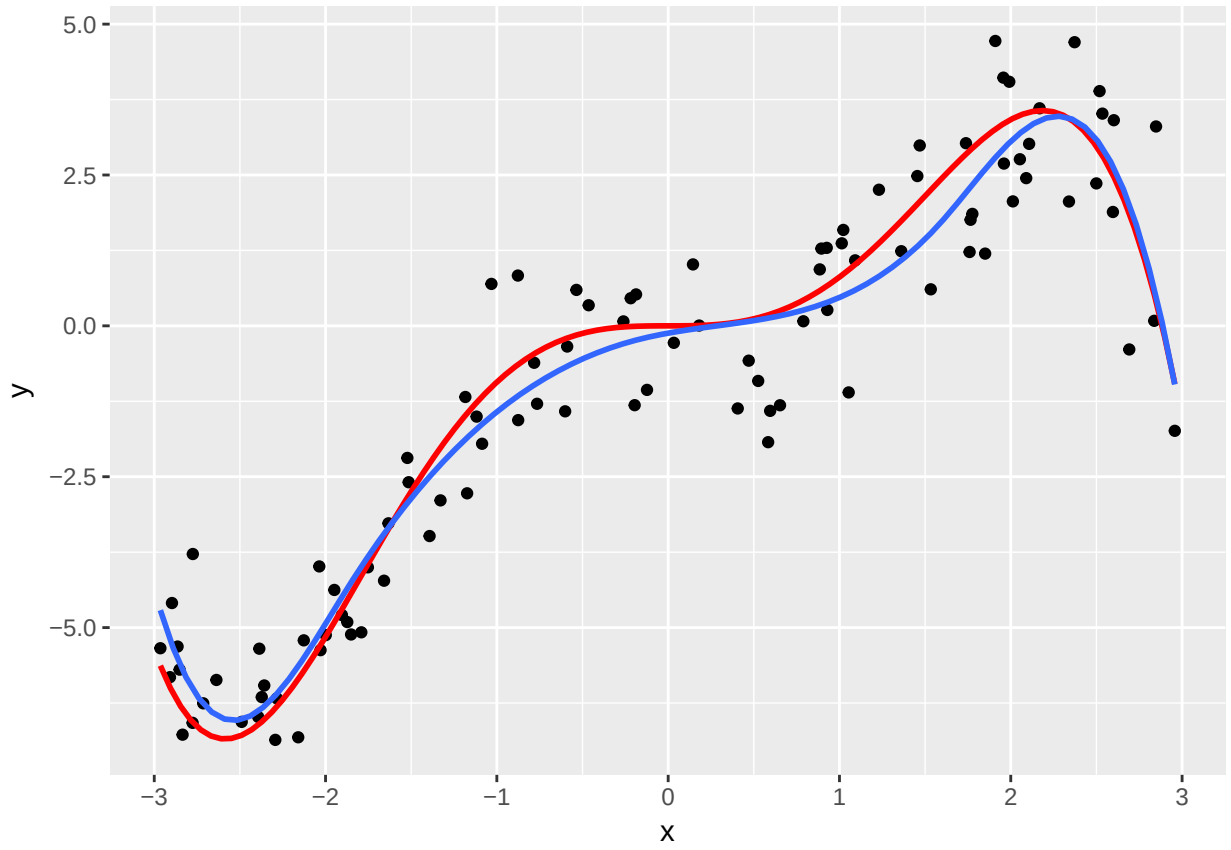



```
# 先用三次多项式去拟合
cubic.lm <- lm(y ~ poly(x, 3))
p+stat_smooth(method = ”lm”,formula = y ~ poly(x, 3),se=FALSE)
```




```
# 可以看到效果并不理想


# 用 B 样条进行拟合，取自由度为 6，即 p 为 3，节点个数也为 3
m.bsp <- lm(y ~ bs(x, df = 6))
p+geom_smooth(method = "lm",formula =  y~bs(x, df=6),se=FALSE)
```




```
# 可以看到拟合效果很好
attr(bs(x, df = 6), "knots") # 样条的节点
```

```
##       25%       50%       75%
## -1.8564418 -0.1914779  1.5844354
```

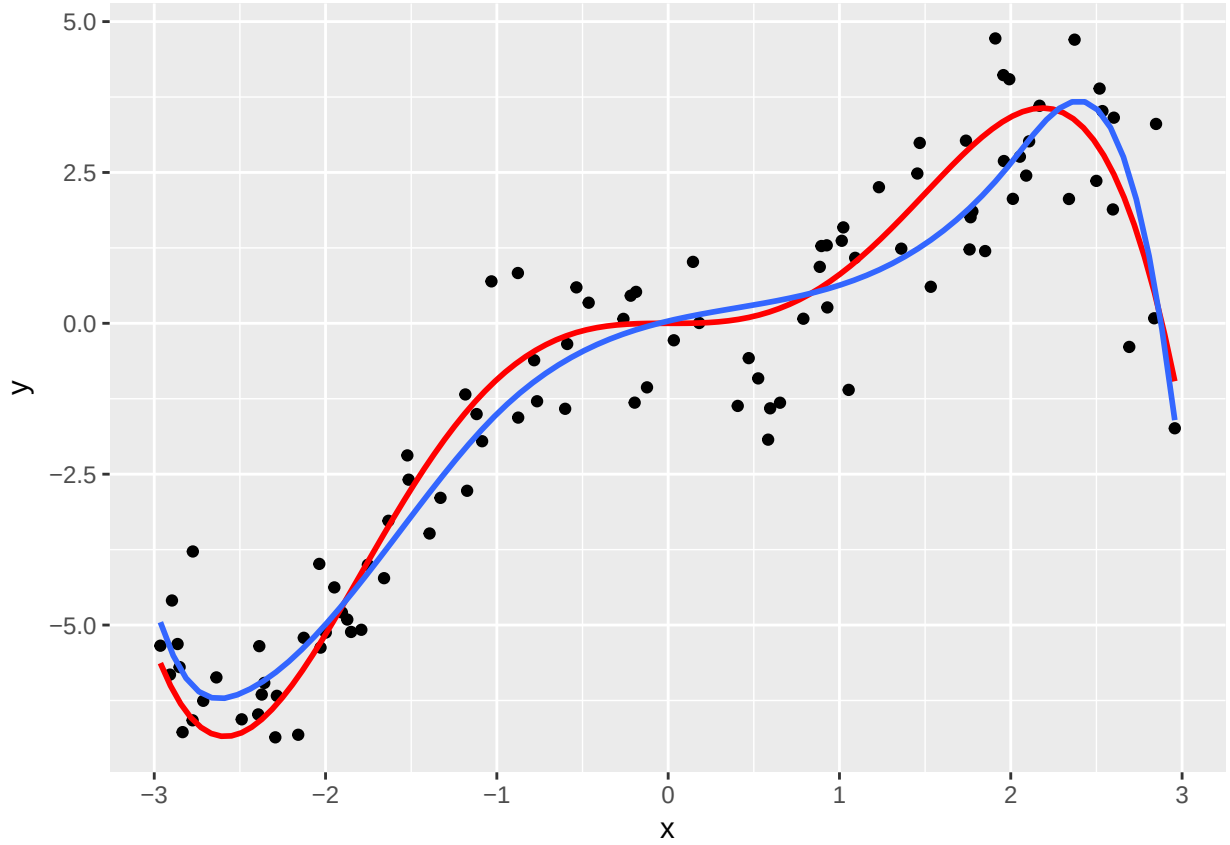
```
# 节点在不指定的情况下默认的是均匀样条
# 利用 summary() 可以得到系数
summary(m.bsp)
```

```
##
## Call:
## lm(formula = y ~ bs(x, df = 6))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40453 -0.73206 -0.09721  0.72541  2.81090
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -4.7118     0.5153  -9.143 1.33e-14 ***
## bs(x, df = 6)1  -3.4755     1.0012  -3.471 0.000787 ***
## bs(x, df = 6)2   2.1702     0.7175   3.025 0.003219 **
## bs(x, df = 6)3   5.2665     0.9341   5.638 1.83e-07 ***
## bs(x, df = 6)4   4.2522     0.8146   5.220 1.08e-06 ***
## bs(x, df = 6)5  10.6106     0.8571  12.379  < 2e-16 ***
## bs(x, df = 6)6   3.7400     0.8262   4.527 1.77e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.012 on 93 degrees of freedom
## Multiple R-squared:  0.9123, Adjusted R-squared:  0.9066
## F-statistic: 161.1 on 6 and 93 DF,  p-value: < 2.2e-16
```

```
# 将系数带入 f(x) 的 B 样条基展开中即可得到一个显式的表达式

# 也可以自己选取节点
m.bsp1 <- lm(y ~ bs(x, df = 6, knots = c(-2.5, -1, 2)))
p+geom_smooth(method = "lm",formula =  y~bs(x, df = 6, knots = c(-2.5, -1, 2)),se=FALSE)
```




```
detach(dat)
```

P-spline

概念

由上面的例子可以看出来用 B 样条估计的效果很好，但在数据的拟合就是要找到一个函数$g(x)$使得$RSS = \sum_{i=1}^{n}(y_i - g(x_i))^2$最小，但是如果对$g(x_i)$不施加任何约束条件，为了得到一个取值为零的 RSS，只要选择 g 在每个y_i处做插值即可，但这样的函数严重过拟合，会变得异常不光滑，因此是没有意义的。因此需要的目标函数是要在能够满足 RSS 尽量小的同时也尽量光滑。

| 解决过拟合的一个常用方法就是引入惩罚项，即最小化

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int [g''(t)^2]dt$$

其中λ 是非负的调节参数 (tuning parameter)，这样得到的$g(x)$ 就是一个光滑样条 (Smoothing spline)。

而当对上文的 B-spline 引入惩罚项进行光滑估计时，就可以得到 p-spline，也就是惩罚样条。

需要注意的是，在拟合 p 样条时，不需要实现选择节点的个数或位置，实际上在每一个训练观测点都有一个节点，但是关键就是要确定λ 的值，λ 越小，损失函数就越小，而。λ 越大，曲线就会越光滑，因此λ 就是控制光滑样条的偏差 -方差的权衡。确定λ 的一个很自然的解决方案就是利用交叉验证.

关于 P-spline 的更详细内容，有兴趣的同学可以自行阅读 Paul & Brian (1996) 的 Flexible Smoothing with B-spline and Penalities.

程序实现

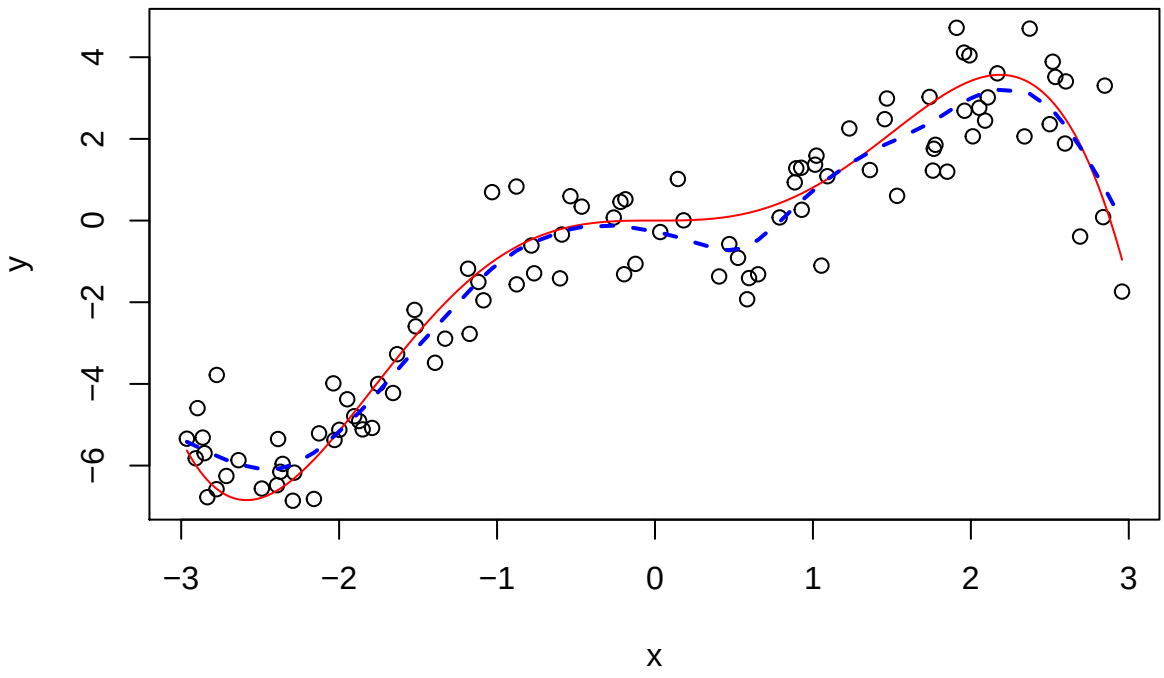
| 同样的，在 R 的 splines 包中提供了函数 smooth.spline 来拟合,q 其调用格式为：smooth.spline(x, y = NULL, w = NULL, df, spar = NULL,cv = FALSE,all.knots = FALSE, nknots =.nknots.smspl,…)
- x 和 y：自变量和因变量
- weight：x 的权重向量，默认为 1
- df：自由度
- spar：光滑参数
- cv：取 TRUE 时用交叉验证迭代，只当没有指定自由度时用到; 默认为 FALSE，用广义交叉验证 (GCV)
- all.knots：默认为 FALSE，取 TRUE 时代表将 x 中所有数都作为节点
- nknots：当 all.knots 取 FALSE 时指定节点个数

继续用上面的数据进行光滑样条的拟合：

```
library(splines)
attach(dat)
# 可以自己选取自由度
fit1 <- smooth.spline(x, y,df=12)
# 也可以用交叉验证
fit2 <- smooth.spline(x, y,cv=TRUE)
fit2
```

```
## Call:
## smooth.spline(x = x, y = y, cv = TRUE)
##
## Smoothing Parameter  spar= 0.6889239  lambda= 0.0001466626 (10 iterations)
## Equivalent Degrees of Freedom (Df): 11.00598
## Penalized Criterion: 82.44829
## PRESS: 1.099928
```

```
plot(x,y)
x.plot = seq(min(x), max(x), length.out = 1000)
y.plot = s(x.plot)
lines(x.plot,y.plot,col="red")  # 实际曲线
lines(fit2, lty = 2,lwd=2, col = "blue")
```




```
# 可以看到拟合效果也非常好

fit2$fit$nk  # 节点数
```

```
## [1] 64
```

```
coef <- fit2$fit$coef  # 系数
detach(dat)
```