

Non-linear and tree models Tutorials

Repeat the R codes in 7.8Lab: Non-linear Modeling and 8.3 Lab: Decision Trees from 'An Introduction to Statistical Learning' (James et al., 2021), with detailed annotations.

Jason Huang

2025-12-14

目錄

7.8 Lab: Non-linear Modeling	1
7.8.1 Polynomial Regression and Step Functions	2
7.8.2 Splines	8
7.8.3 GAMs	13
8.3 Lab: Decision Trees	20
8.3.1 Fitting Classification Trees	20
8.3.2 Fitting Regression Trees	27
8.3.3 Bagging and Random Forests	30
8.3.4 Boosting	33
8.3.5 Bayesian Additive Regression Trees	36

7.8 Lab: Non-linear Modeling

```
library(ISLR2)
attach(Wage)
```

7.8.1 Polynomial Regression and Step Functions

```
# 使用 `lm()` 函數擬合線性模型以預測薪資
# poly(age, 4) 是 R 的預設多項式函數
# 它會生成 4 階的【正交多項式 (Orthogonal Polynomials)】
fit <- lm(wage ~ poly(age, 4), data = Wage)
# 取出並查看模型的係數矩陣
coef(summary(fit))
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	111.70361	0.7287409	153.283015	0.000000e+00
poly(age, 4)1	447.06785	39.9147851	11.200558	1.484604e-28
poly(age, 4)2	-478.31581	39.9147851	-11.983424	2.355831e-32
poly(age, 4)3	125.52169	39.9147851	3.144742	1.678622e-03
poly(age, 4)4	-77.91118	39.9147851	-1.951938	5.103865e-02

```
# raw = T (True) 這個參決定了你的自變數 (年齡 age ) 是
# 使用「正交多項式 (Orthogonal Polynomials)」還是「原始多項式 (Raw Polynomials)」
# raw = T 會強制 poly() 生成 4 階的【原始多項式 (Raw Polynomials)】。
fit2 <- lm(wage ~ poly(age, 4, raw = T), data = Wage)
# 取出並查看模型的係數矩陣
# fit 和 fit2 的預測曲線【完全相同】· 但係數數值和解釋意義【完全不同】。
coef(summary(fit2))
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.841542e+02	6.004038e+01	-3.067172	0.0021802539
poly(age, 4, raw = T)1	2.124552e+01	5.886748e+00	3.609042	0.0003123618
poly(age, 4, raw = T)2	-5.638593e-01	2.061083e-01	-2.735743	0.0062606446
poly(age, 4, raw = T)3	6.810688e-03	3.065931e-03	2.221409	0.0263977518
poly(age, 4, raw = T)4	-3.203830e-05	1.641359e-05	-1.951938	0.0510386498

```
# 為了讓 R 能夠計算 age 的平方 (age^2)、立方 (age^3) 等 · 必須使用 I() 函數 (AsIs 函數) 。
# I() 告訴 R :「請按字面意義計算括號內的表達式 (如 age^2) · 不要進行公式符號的特殊解讀。」
fit2a <- lm(wage ~ age + I(age^2) + I(age^3) + I(age^4),
data = Wage)
# coef() 僅提取係數的數值 · 不像 coef(summary()) 會顯示標準誤和 P 值等 。
# 這個模型 (fit2a) 的係數和 fit2 的係數【完全相同】。
coef(fit2a)
```

(Intercept)	age	I(age^2)	I(age^3)	I(age^4)
-1.841542e+02	2.124552e+01	-5.638593e-01	6.810688e-03	-3.203830e-05

```
# 這是擬合多項式的第三種寫法，使用 cbind() 創建一個矩陣作為單一自變數。
# cbind(age, age^2, age^3, age^4) 將 age 的 1 到 4 次方組合為一個新的矩陣變數。
# 這種方法樣等價於 fit2 和 fit2a，但它會將矩陣的名稱（此例中為 cbind(...)）作為係數名稱的一部分。
fit2b <- lm(wage ~ cbind(age, age^2, age^3, age^4),
data = Wage)
```

這段程式碼以更簡潔的方式實現了相同的功能，它使用 `cbind()` 函數從一組向量建立矩陣；公式中任何類似 `cbind()` 的函數呼叫也起到了 wrapper(包裝器) 的作用。

現在，我們建立一個包含需要進行預測的年齡值的網格，然後呼叫通用的 `predict()` 函數，並指定我們需要標準誤差。

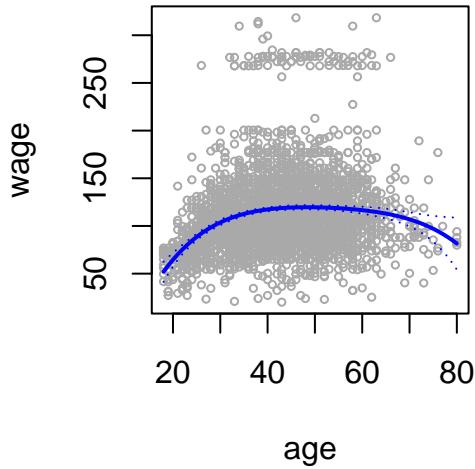
```
# 取得 'age' 變數的最小值和最大值（即年齡範圍）
agelims <- range(age)
# 建立一個從最小年齡到最大年齡的等差序列（用於繪圖預測線）
age.grid <- seq(from = agelims[1], to = agelims[2])
# 使用模型 'fit'（正交多項式）在 age.grid 上計算預測值及其標準誤差
preds <- predict(fit, newdata = list(age = age.grid),
se = TRUE)
# 計算 95% 信賴區間的上下限（預測值 ± 兩倍標準誤差）
se.bands <- cbind(preds$fit + 2 * preds$se.fit,
preds$fit - 2 * preds$se.fit)

# 設定繪圖參數：將繪圖區域分為 1 行 2 列，並調整邊界大小。
par(mfrow = c(1, 2), mar = c(4.5, 4.5, 1, 1),
oma = c(0, 0, 4, 0))
# 繪製原始數據的散點圖（年齡 vs 薪資），設定 x 軸範圍和點的大小/顏色。
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
# 為整個繪圖區域 (outer = T) 設定一個標題
title("Degree -4 Polynomial", outer = T)
# 在散點圖上繪製模型 'fit' 的擬合曲線
lines(age.grid, preds$fit, lwd = 2, col = "blue")
# 繪製預測值的 95% 信賴區間上下限（使用虛線 lty = 3）
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)

# 使用模型 'fit2'（原始多項式）在 age.grid 上計算預測值及其標準誤差
preds2 <- predict(fit2, newdata = list(age = age.grid),
se = TRUE)
# 計算模型 'fit' 與 'fit2' 預測值之間的最大絕對值差異，驗證兩者曲線是否相同。
max(abs(preds$fit - preds2$fit))
```

```
[1] 6.842527e-11
```

Degree -4 Polynomial



現在我們擬合從線性到五次多項式的各種模型，並試圖確定足以解釋薪資和年齡之間關係的最簡單模型。我們使用 `anova()` 函數執行變異數分析 (ANOVA，使用 F 檢定)，以檢驗 null analysis (即模型 M1 足以解釋資料) 與 alternative hypothesis (即需要更複雜的模型 M2)。為了使用 `anova()` 函數，M1 和 M2 必須是 nested models 巢狀模型：M1 中的預測變數必須是 M2 中預測變數的子集。在本例中，我們擬合了五個不同的模型，並依序比較較簡單的模型和較複雜的模型。

```
# 擬合一階多項式模型 (簡單線性迴歸)：薪資 (wage) 對 年齡 (age) 的線性關係。
fit.1 <- lm(wage ~ age, data = Wage)
# 擬合二階多項式模型 (二次曲線)：使用 age 和 age^2 的正交項來預測薪資。
fit.2 <- lm(wage ~ poly(age, 2), data = Wage)
# 擬合三階多項式模型 (三次曲線)：在二階基礎上加入三次方正交項。
fit.3 <- lm(wage ~ poly(age, 3), data = Wage)
# 擬合四階多項式模型 (四次曲線)：在三階基礎上加入四次方正交項。
fit.4 <- lm(wage ~ poly(age, 4), data = Wage)
# 擬合五階多項式模型 (五次曲線)：在四階基礎上加入五次方正交項。
fit.5 <- lm(wage ~ poly(age, 5), data = Wage)

# 對這五個巢狀 (Nested) 模型進行變異數分析 (ANOVA) 檢定
# 比較加入更高階多項式項後，模型擬合的改善是否具有統計顯著性。
anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

Analysis of Variance Table

```

Model 1: wage ~ age
Model 2: wage ~ poly(age, 2)
Model 3: wage ~ poly(age, 3)
Model 4: wage ~ poly(age, 4)
Model 5: wage ~ poly(age, 5)

  Res.Df   RSS Df Sum of Sq    F    Pr(>F)
1   2998 5022216
2   2997 4793430  1   228786 143.5931 < 2.2e-16 ***
3   2996 4777674  1   15756   9.8888  0.001679 **
4   2995 4771604  1   6070   3.8098  0.051046 .
5   2994 4770322  1   1283   0.8050  0.369682
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

線性模型 1 與二次模型 2 的 p 值接近零 ($< 10^{-16}$)，顯示線性擬合不足以解釋資料。同樣，二次模型 2 與三次模型 3 的 p 值也非常低 (0.0017)，因此二次擬合也不足以解釋資料。三次多項式模型 3 與四次多項式模型 4 的 p 值約為 5%，而五次多項式模型 5 的 p 值為 0.37，因此似乎沒有必要。由此可見，三次或四次多項式似乎都能較好地擬合數據，但更高階或更低階的模型則沒有必要。

在這種情況下，與其使用 `anova()` 函數，不如利用 `poly()` 函數產生正交多項式的特性，更簡潔地獲得這些 p 值。

```

# 取出並查看模型的係數矩陣
coef(summary(fit.5))

```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	111.70361	0.7287647	153.2780243	0.000000e+00
poly(age, 5)1	447.06785	39.9160847	11.2001930	1.491111e-28
poly(age, 5)2	-478.31581	39.9160847	-11.9830341	2.367734e-32
poly(age, 5)3	125.52169	39.9160847	3.1446392	1.679213e-03
poly(age, 5)4	-77.91118	39.9160847	-1.9518743	5.104623e-02
poly(age, 5)5	-35.81289	39.9160847	-0.8972045	3.696820e-01

```

# t 統計量的平方等於 anova() 函數計算的 F 統計量
(-11.983)^2

```

[1] 143.5923

我們可以使用 `anova()` 函數來比較以下三個模型

```

# 擬合線性模型：使用教育程度 (education) 和年齡 (age) 來預測薪資 (wage)。
# 這是最簡單的線性關係模型，假設年齡和薪資呈一條直線關係。
fit.1 <- lm(wage ~ education + age, data = Wage)

```

```
# 在 fit.1 的基礎上，將年齡項替換為二階多項式 (poly(age, 2))。
# 允許薪資隨年齡的變化呈現二次曲線（例如拋物線）關係
fit.2 <- lm(wage ~ education + poly(age, 2), data = Wage)
# 在 fit.2 的基礎上，將年齡項提高到三階多項式 (poly(age, 3))。
# 允許薪資隨年齡的變化呈現更複雜的三次曲線關係
fit.3 <- lm(wage ~ education + poly(age, 3), data = Wage)

# 對這三個巢狀 (Nested) 模型進行變異數分析 (ANOVA) 檢定。
# 用於判斷增加更高階年齡項對模型擬合的改善是否具有統計顯著性
anova(fit.1, fit.2, fit.3)
```

Analysis of Variance Table

	Model 1: wage ~ education + age	Model 2: wage ~ education + poly(age, 2)	Model 3: wage ~ education + poly(age, 3)		
Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	2994	3867992			
2	2993	3725395	1	142597	114.6969 <2e-16 ***
3	2992	3719809	1	5587	4.4936 0.0341 *

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

除了假設檢定和變異數分析之外，我們還可以採用交叉驗證法來選擇多項式階數，如第五章所述。接下來，我們考慮預測個人年收入是否超過 25 萬美元的問題。

我們的步驟與之前基本上相同，只是先創建合適的響應向量，然後應用 `glm()` 函數，並使用 `family = "binomial"` 來擬合多項式邏輯迴歸模型。

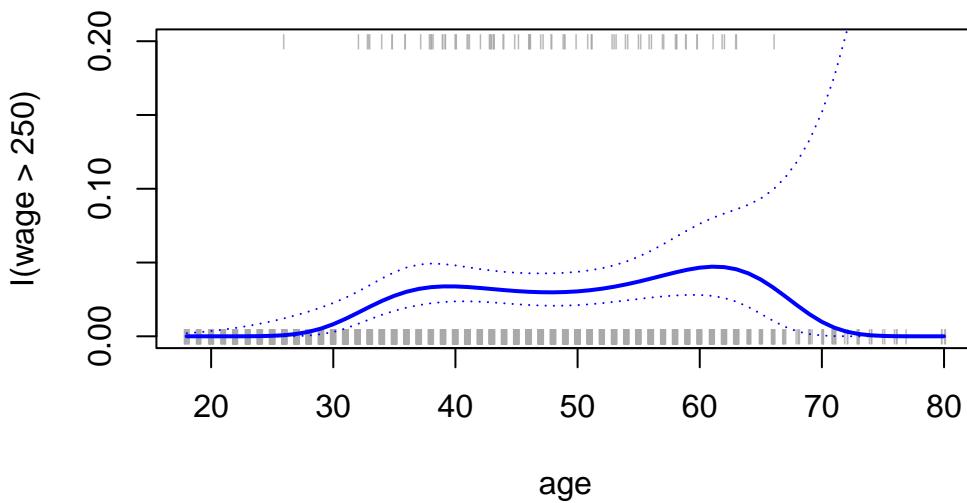
```
# 擬合廣義線性模型 (GLM)：預測薪資是否大於 250
# 使用四階年齡正交多項式，並指定二項式 (binomial) 族群。
fit <- glm(I(wage > 250) ~ poly(age, 4), data = Wage,
family = binomial)
# 使用模型 'fit' 在年齡網格上計算預測值，但輸出的是【Logit 尺度】。
preds <- predict(fit, newdata = list(age = age.grid), se = T)
# 將 Logit 尺度的預測值 (preds$fit) 轉換回【機率尺度】(Probability)
# 用於繪製擬合曲線
pfit <- exp(preds$fit) / (1 + exp(preds$fit))
# 在 Logit 尺度上，計算 95% 信賴區間的上下限（預測值 ± 兩倍標準誤差）。
se.bands.logit <- cbind(preds$fit + 2 * preds$se.fit,
preds$fit - 2 * preds$se.fit)
```

```

# 將 Logit 尺度的信賴區間上下限
# 使用 Sigmoid 函數轉換回【機率尺度】，確保數值在 0 到 1 之間。
se.bands <- exp(se.bands.logit) / (1 + exp(se.bands.logit))
# 再次計算預測值，直接要求輸出【機率尺度】(type = "response")，用於說明其缺點。
preds <- predict(fit, newdata = list(age = age.grid),
type = "response", se = T)

# 然而相應的置信區間是不合理的，因為最終會得到負機率！
# 初始化繪圖：建立一個空的散點圖，設置 x 軸範圍和 y 軸範圍。
plot(age, I(wage > 250), xlim = agelims, type = "n",
ylim = c(0, .2))
# 繪製原始數據的二元結果
# 使用 jitter() 在垂直方向上微幅擾動，並將點位 (0 和 1) 壓縮至 y 軸範圍內顯示。
points(jitter(age), I((wage > 250) / 5), cex = .5, pch = "|", col
= "darkgrey")
# 繪製模型在【機率尺度】上的擬合曲線（即 Logit 轉換後的 pfit）
lines(age.grid, pfit, lwd = 2, col = "blue")
# 繪製【機率尺度】的 95% 信賴區間上下限（使用 Logit 轉換得到的 se.bands）
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)

```



我們在圖表的頂部用灰色標記標出了工資值高於 250 的觀測值對應的年齡值，在圖表的底部用灰色標記標出了工資值低於 250 的觀測值對應的年齡值。

```
# 將連續變數 'age' 切割成 4 個等寬的區間 (Bins)。
# table() 函數計算並顯示每個區間內的資料點個數 (頻率)
table(cut(age, 4))
```

(17.9,33.5]	(33.5,49]	(49,64.5]	(64.5,80.1]
750	1399	779	72

```
# 擬合一個線性模型，使用切割後的年齡區間作為唯一的預測變數。
fit <- lm(wage ~ cut(age, 4), data = Wage)
# 查看此模型的係數矩陣
coef(summary(fit))
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	94.158392	1.476069	63.789970	0.000000e+00
cut(age, 4)(33.5,49]	24.053491	1.829431	13.148074	1.982315e-38
cut(age, 4)(49,64.5]	23.664559	2.067958	11.443444	1.040750e-29
cut(age, 4)(64.5,80.1]	7.640592	4.987424	1.531972	1.256350e-01

這裡，`cut()` 函數自動選擇了 33.5 歲、49 歲和 64.5 歲這三個年齡分界點。我們也可以使用 `breaks` 選項直接指定自己的分界點。

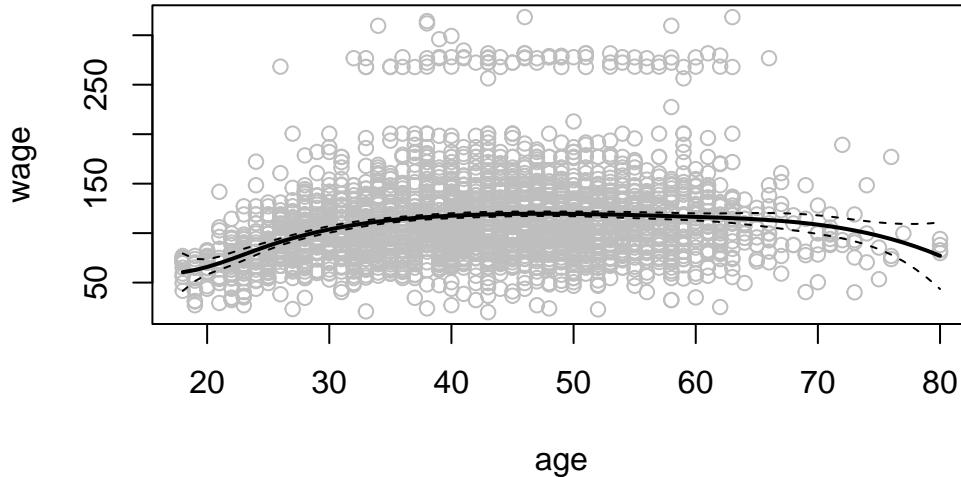
`lm()` 函數會建立一組虛擬變數用於迴歸分析。年齡小於 33.5 歲的類別被排除在外，因此截距係數 94,160 美元可以解釋為 33.5 歲以下人群的平均工資，其他係數可以解釋為其他年齡組人群的平均額外工資。我們可以像多項式擬合那樣產生預測結果和圖表。

7.8.2 Splines

為了在 R 中擬合迴歸樣條，我們使用 `splines` 套件。`bs()` 函數會產生具有指定節點集的樣條的完整基底函數矩陣。預設情況下產生的是三次樣條。

```
# 載入 'splines' 套件，這是用於生成樣條基函數的核心函式庫。
library(splines)
# 擬合線性模型：使用 bs() 函數創建 三次 B 樣條 基函數矩陣作為預測變數。
fit <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
# 使用樣條模型對年齡網格 (age.grid) 進行預測，並計算預測的標準誤 (se.fit)。
pred <- predict(fit, newdata = list(age = age.grid), se = T)
# 繪製原始數據的散點圖 (年齡 vs 薪資)，用灰色點表示，並初始化圖表。
plot(age, wage, col = "gray")
# 在圖上繪製樣條模型的擬合曲線 (預測值)
lines(age.grid, pred$fit, lwd = 2)
```

```
# 繪製擬合曲線的 95% 信賴區間【上限】(預測值 + 兩倍標準誤差) · 使用虛線。
lines(age.grid, pred$fit + 2 * pred$se, lty = "dashed")
# 繪製擬合曲線的 95% 信賴區間【下限】(預測值 - 兩倍標準誤差) · 使用虛線。
lines(age.grid, pred$fit - 2 * pred$se, lty = "dashed")
```



這裡我們預先設定了 25 歲、40 歲和 60 歲這三個節點，這將產生一個具有六個基底函數的樣條曲線。我們也可以使用 df 選項來產生一個節點位於資料均勻分位數的樣條曲線。

```
library(boot)
# 訓練 GLM 模型
glm.fit <- glm(mpg ~ horsepower, data = Auto)
# `cv.glm()` 函數會產生一個包含多個元素的清單，`delta` 向量中的兩個數字包含交叉驗證結果。
cv.err <- cv.glm(Auto, glm.fit)
cv.err$delta
```

[1] 24.23151 24.23114

我們可以對越來越複雜的多項式擬合重複此過程。

為了實現自動化，我們使用 `for()` 函數啟動一個 `for` 循環，該循環迭代地擬合階數為 $i = 1$ 到 $i = 10$ 的多項式迴歸，計算對應的交叉驗證誤差，並將其儲存在向量 `cv.error` 的第 i 個元素中。

```
# 建立一個 B 樣條 (Basis Spline) 的基函數矩陣 · 指定在 25, 40, 60 歲處有節點 (knots)。
# dim() 輸出矩陣的維度：列數是資料點個數，欄數是基函數的數量
# f = knots 數量 + 自由度
dim(bs(age, knots = c(25, 40, 60)))
```

```
[1] 3000     6
```

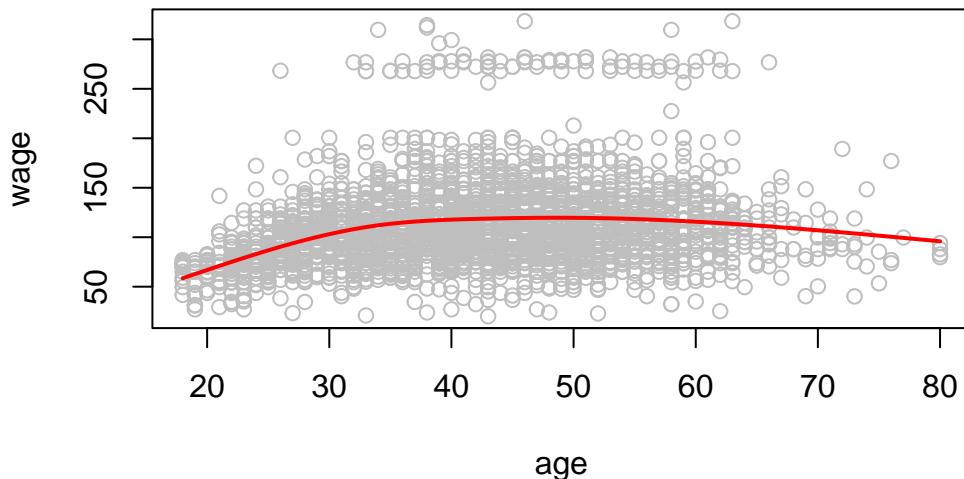
```
# 建立一個 B 樣條的基函數矩陣，指定模型有 6 個自由度 (df = 6)。  
# R 會根據 df 自動選擇節點 (knots) 的位置 (通常是等分點)。  
dim(bs(age, df = 6))
```

```
[1] 3000     6
```

```
# 查詢當自由度 (df) 設為 6 時，R 自動選擇的內部節點 (knots) 實際位置。  
attr(bs(age, df = 6), "knots")
```

```
[1] 33.75 42.00 51.00
```

```
plot(age, wage, col = "gray")  
# 擬合線性模型，使用自然樣條 (Natural Spline, ns) 基函數來預測薪資  
# 模型有 4 個自由度 (df=4)  
fit2 <- lm(wage ~ ns(age, df = 4), data = Wage)  
# 使用自然樣條模型 (fit2) 在年齡網格上計算預測值及其標準誤差  
pred2 <- predict(fit2, newdata = list(age = age.grid),  
se = T)  
# 在圖上繪製自然樣條 (Natural Spline, ns) 模型的擬合曲線 (紅線)  
lines(age.grid, pred2$fit, col = "red", lwd = 2)
```



```
# 繪製原始數據的散點圖 (年齡 vs 薪資)，準備繪製平滑樣條曲線。  
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")  
# 圖表標題
```

```

title("Smoothing Spline")
# 擬合第一個平滑樣條模型：手動指定自由度 (df) 為 16 (一個較為靈活的曲線)。
fit <- smooth.spline(age, wage, df = 16)
# 擬合第二個平滑樣條模型：使用【交叉驗證 (Cross-Validation, CV)】
# 選擇最佳的平滑度參數 (即選擇最佳自由度)
fit2 <- smooth.spline(age, wage, cv = TRUE)
# 顯示由交叉驗證 (CV) 自動選擇出來的最佳自由度 (df) 數值
fit2$df

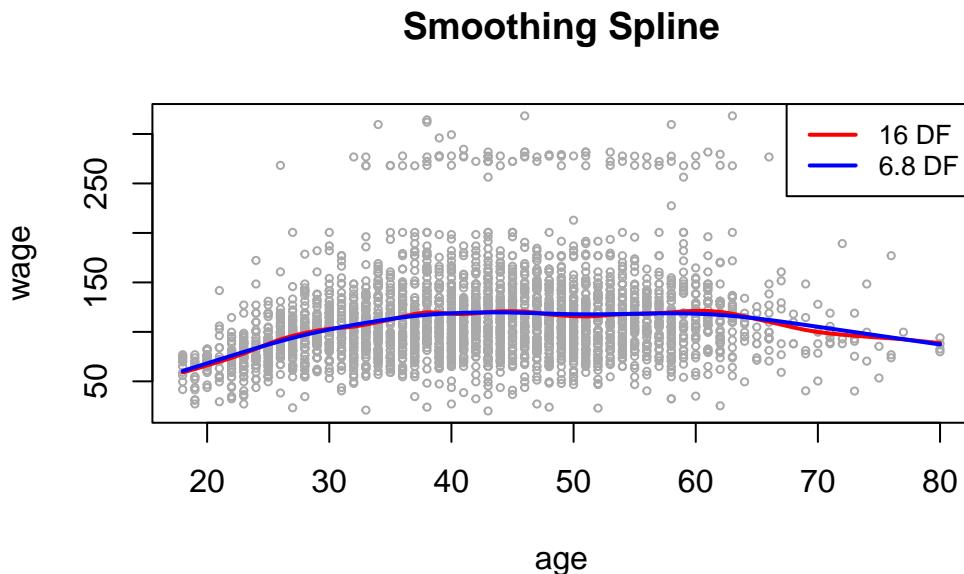
```

[1] 6.794596

```

# 在圖上繪製手動指定 df=16 的平滑樣條曲線 (紅線)
lines(fit, col = "red", lwd = 2)
# 在圖上繪製交叉驗證自動選擇 df (fit2$df) 的平滑樣條曲線 (藍線)
lines(fit2, col = "blue", lwd = 2)
# 在圖右上角添加圖例，標註紅線 (16 DF) 和藍線 (CV 選出的 6.8 DF) 所代表的模型。
legend("topright", legend = c("16 DF", "6.8 DF"),
col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)

```



在第一次呼叫 `smooth.spline()` 時，我們指定了 `df = 16`。此函數隨後確定哪個 λ 值能夠使自由度達到 16。在第二次呼叫 `smooth.spline()` 時，我們透過交叉驗證來選擇平滑度等級；這得到的 λ 值能夠使自由度達到 6.8。

```

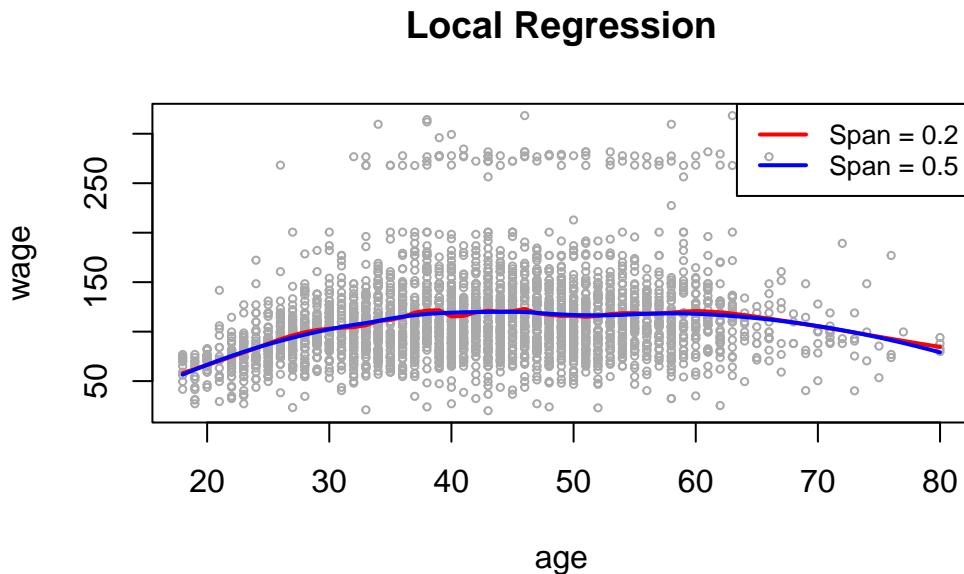
# 繪製原始數據的散點圖 (年齡 vs 薪資)，準備繪製局部迴歸曲線。
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")

```

```

# 圖表標題
title("Local Regression")
# 擬合第一個局部加權迴歸模型 (LOESS)
# 使用較小的平滑參數 (span = 0.2) · 得到一條較為【不平滑/靈活】的曲線。
fit <- loess(wage ~ age, span = .2, data = Wage)
# 擬合第二個局部加權迴歸模型 (LOESS)
# 使用較大的平滑參數 (span = 0.5) · 得到一條較為【平滑/僵硬】的曲線。
fit2 <- loess(wage ~ age, span = .5, data = Wage)
# 繪製第一個模型 (span=0.2) 在年齡網格上的預測曲線 (紅線)
lines(age.grid, predict(fit, data.frame(age = age.grid)),
col = "red", lwd = 2)
# 繪製第二個模型 (span=0.5) 在年齡網格上的預測曲線 (藍線)
lines(age.grid, predict(fit2, data.frame(age = age.grid)),
col = "blue", lwd = 2)
# 在圖的右上角添加圖例 · 解釋紅線和藍線分別代表的平滑參數 (Span) 大小。
legend("topright", legend = c("Span = 0.2", "Span = 0.5"),
col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)

```

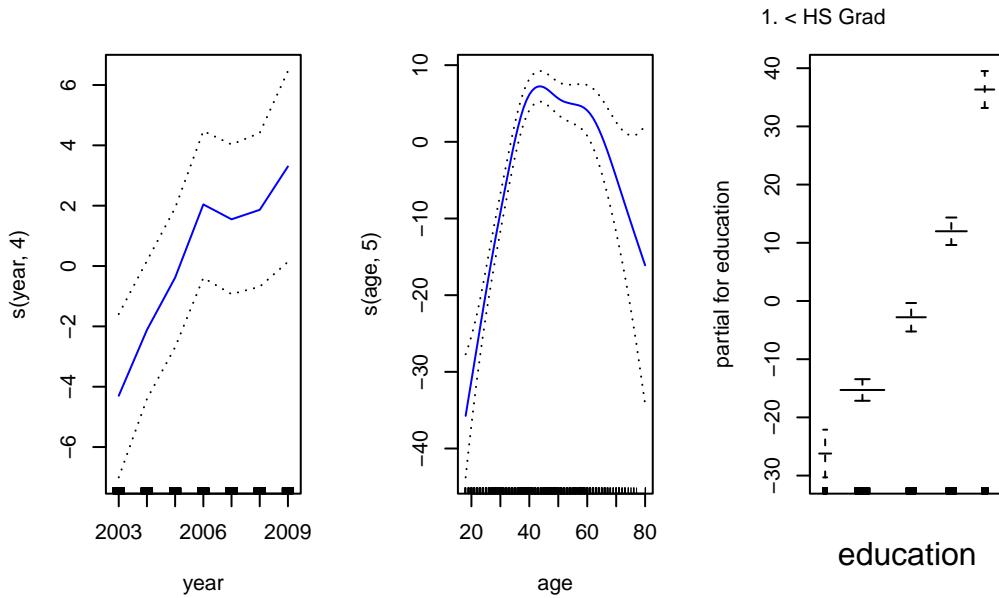


這裡我們使用 0.2 和 0.5 的跨度進行了局部線性迴歸 · 也就是說 · 每個鄰域包含 20% 或 50% 的觀測值。跨度越大 · 擬合效果越平滑。

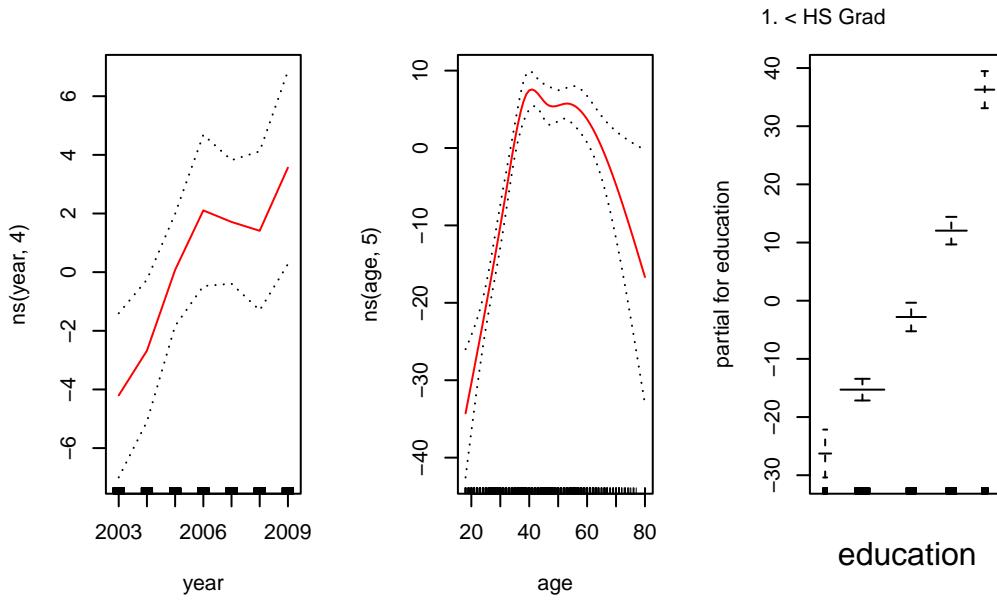
7.8.3 GAMs

現在我們使用年份和年齡的自然樣條函數來擬合一個廣義加性模型 (GAM) 來預測工資，並將教育程度視為定性預測變量。由於這只是一個使用適當選擇的基底函數的大型線性迴歸模型，我們可以直接使用 `lm()` 函數來實現。

```
# 使用標準的 lm() 函數擬合一個模型，其中 'year' 和 'age' 使用【自然樣條 (ns)】基函數作為平滑項。
# 這個模型本質上是一個迴歸樣條模型，是 GAM 的近似。
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education ,
data = Wage)
# 載入 'gam' 套件，這是用於擬合廣義加法模型 (GAMs) 的核心套件。
library(gam)
# 使用 gam() 函數擬合一個真正的 GAM 模型。
# s() 函數表示使用【平滑樣條 (Smoothing Spline)】來擬合 'year' 和 'age'，df 為 4 和 5。
gam.m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education ,
data = Wage)
# 將繪圖區域分為 1 行 3 列，以便繪製每個平滑項的圖。
par(mfrow = c(1, 3))
# 繪製 GAM 模型 (gam.m3) 中每個平滑項 (year 和 age) 的曲線圖，並顯示標準誤區間。
plot(gam.m3, se = TRUE, col = "blue")
```



```
# 繪製近似 GAM 模型 (gam1) 中每個樣條項的曲線圖。
# 兩個繪圖結果 (plot vs. plot.Gam) 應該非常相似，但 gam.m3 的 s() 曲線會更平滑。
plot.Gam(gam1, se = TRUE, col = "red")
```



```
# 擬合第一個 GAM 模型 (M1)：只對年齡使用平滑項，不包含 year 變數。
gam.m1 <- gam(wage ~ s(age, 5) + education, data = Wage)
# 擬合第二個 GAM 模型 (M2)：對年齡使用平滑項，但對 year 僅使用【線性項】(year)。
gam.m2 <- gam(wage ~ year + s(age, 5) + education,
               data = Wage)
# 對三個巢狀的 GAM 模型 (M1, M2, M3) 進行 ANOVA 檢定，使用 F 統計量來比較加入額外項的改善是否顯著。
anova(gam.m1, gam.m2, gam.m3, test = "F")
```

Analysis of Deviance Table

	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						

```
# 顯示完整 GAM 模型 (gam.m3) 的統計摘要，包括每個平滑項的顯著性檢定和自由度。
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: NA

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

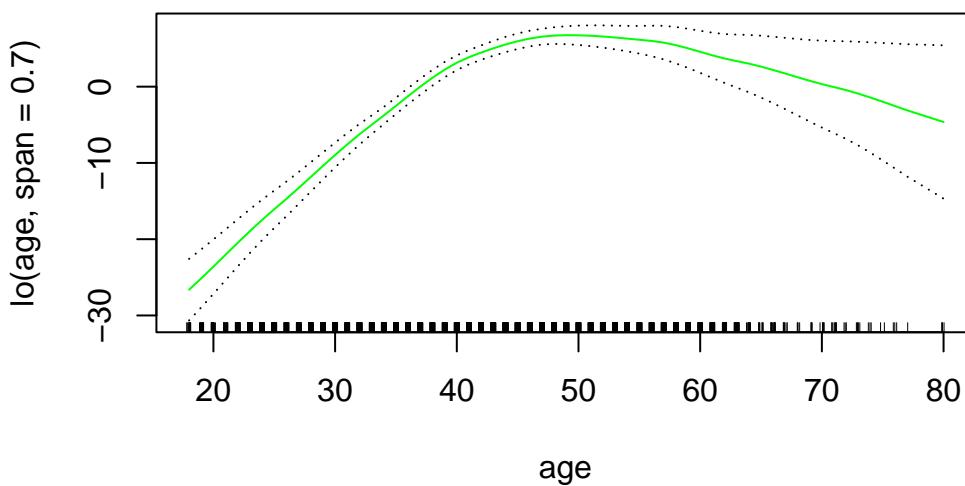
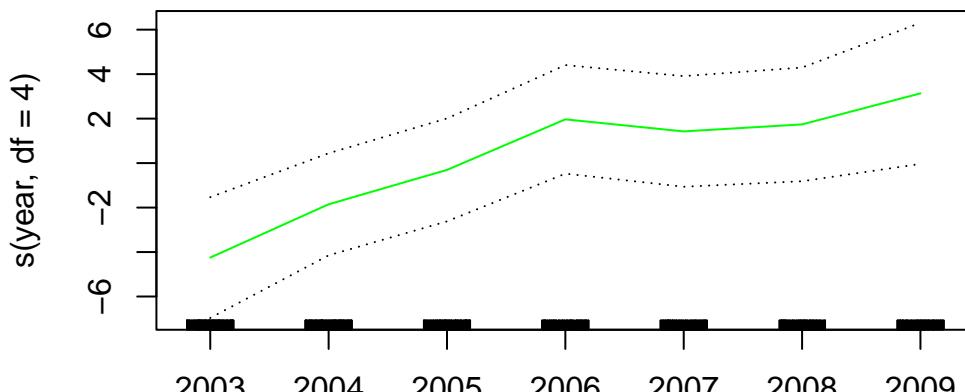
	Npar	Df	Npar	F	Pr(F)						
(Intercept)											
s(year, 4)	3	1.086	0.3537								
s(age, 5)	4	32.380	<2e-16	***							
education											

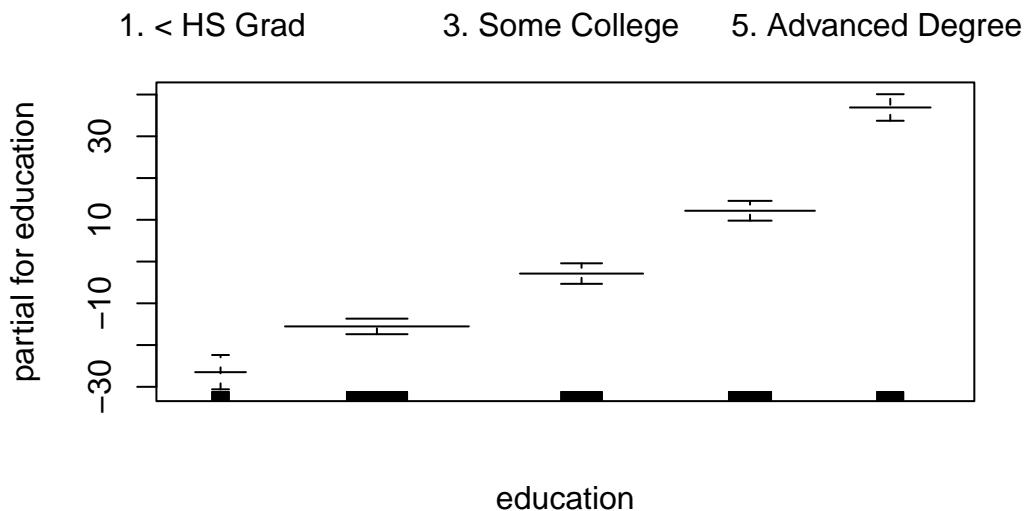
Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'..'	0.1	' '	1

「參數效應變異數分析」的 p 清楚地表明，即使僅假設線性關係，年份、年齡和教育程度也都具有高度統計意義。另一方面，「非參數效應變異數分析」中年份與年齡的 p 值對應於線性關係的零假設與非線性關係的備擇假設。年份較大的 p 值強化了我們從變異數分析中得出的結論，即線性函數足以描述該項。

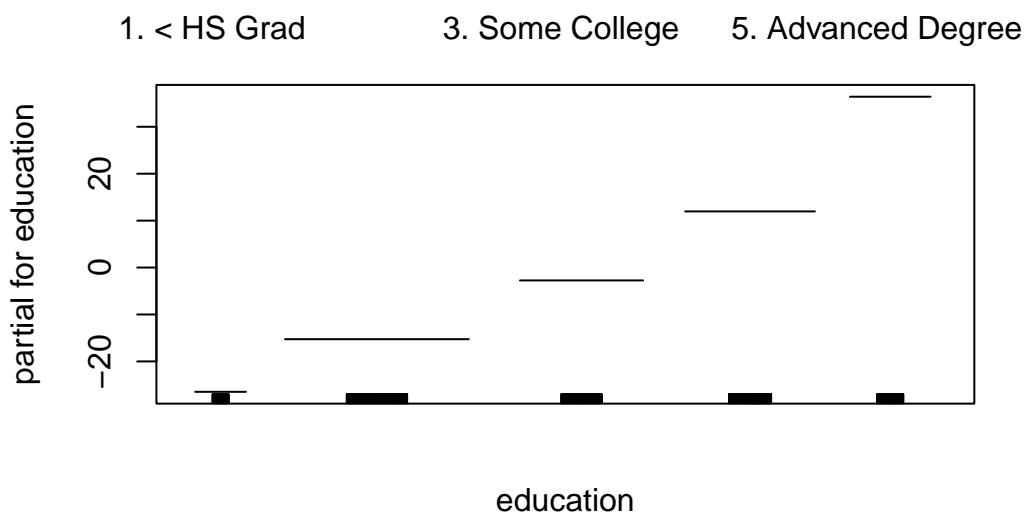
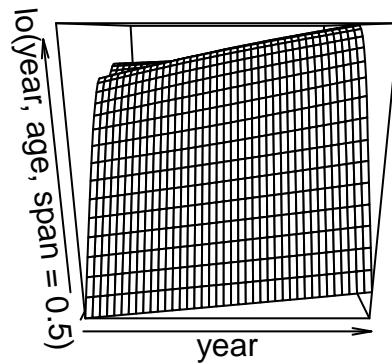
```
# 使用先前擬合的 GAM 模型 (gam.m2) 來預測原始數據集 (Wage) 中的薪資值
preds <- predict(gam.m2, newdata = Wage)
# 擬合一個新的 GAM 模型 (gam.lo)
# 對 'year' 使用平滑樣條 s()，但對 'age' 使用【局部迴歸 (lo())】作為平滑項，span 參數為 0.7。
gam.lo <- gam(
```

```
wage ~ s(year, df = 4) + lo(age, span = 0.7) + education ,  
data = Wage  
)  
# 繪製模型 gam.lo 中每個平滑項的擬合曲線，並顯示標準誤差。  
plot(gam.lo, se = TRUE, col = "green")
```



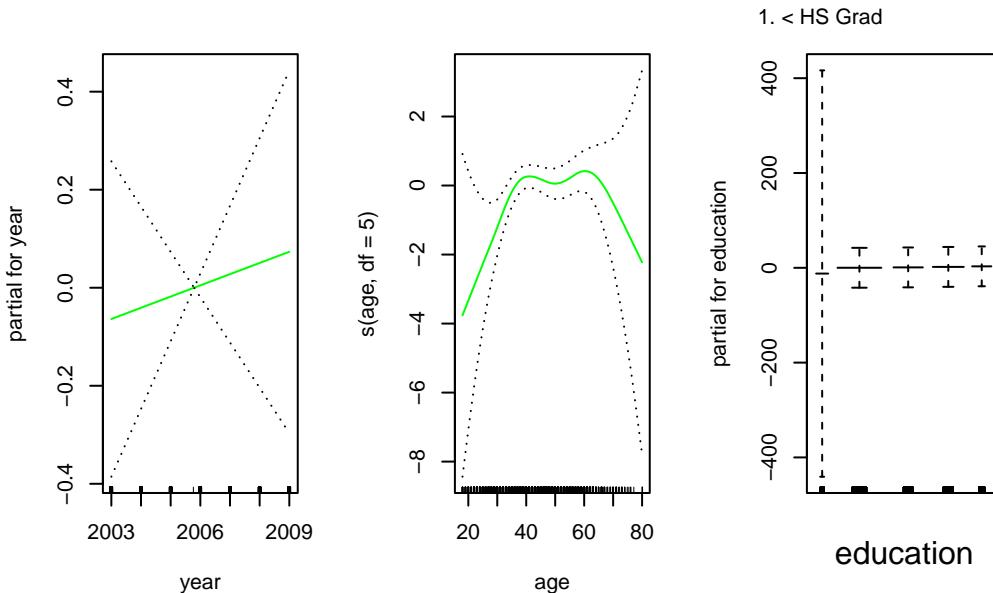


```
# 擬合另一個 GAM 模型 (gam.lo.i)
# 使用 lo(year, age, span = 0.5) 創建【二維的局部迴歸平滑項】
# 捕捉 'year' 和 'age' 之間的非線性交互作用
gam.lo.i <- gam(wage ~ lo(year, age, span = 0.5) + education ,
data = Wage)
# 載入 'akima' 套件，該套件提供了用於二維平滑圖形的可視化功能。
library(akima)
# 繪製帶有二維局部迴歸交互作用項 (lo(year, age)) 的 GAM 模型圖
plot(gam.lo.i)
```



```
# 擬合【邏輯斯迴歸 (Logistic Regression) 的 GAM 模型】
# 目標是預測薪資是否大於 250，使用 year (線性)、age (平滑樣條) 和 education (類別) 作為預測變數。
gam.lr <- gam(
  I(wage > 250) ~ year + s(age, df = 5) + education ,
  family = binomial, data = Wage
)
# 設定繪圖參數：將繪圖區域分為 1 行 3 列。
```

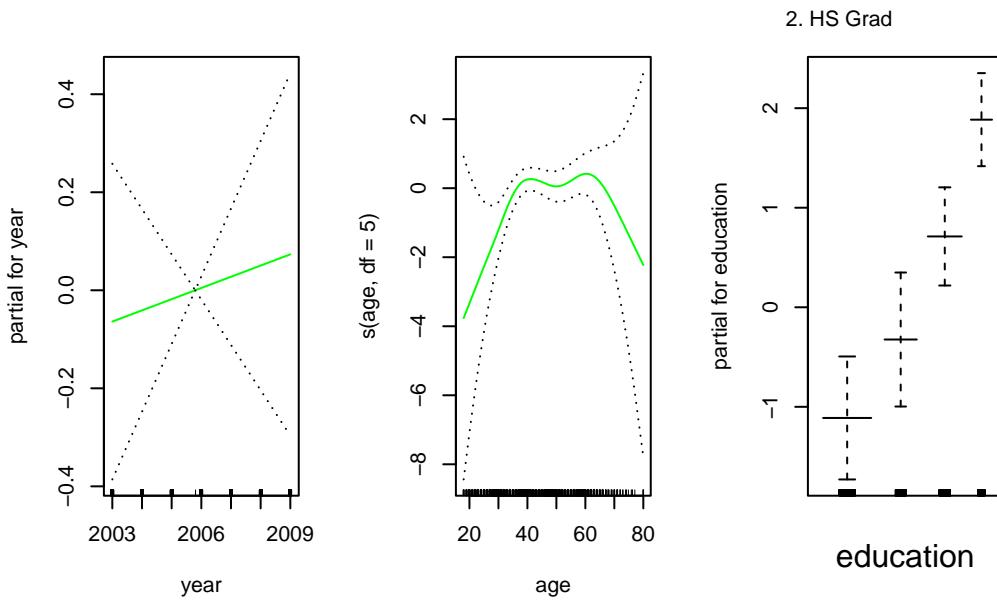
```
par(mfrow = c(1, 3))
# 繪製邏輯斯迴歸 GAM 模型 (gam.lr) 中每個平滑項對 Log-odds 尺度下的影響曲線
plot(gam.lr, se = T, col = "green")
```



```
# 建立一個交叉表，顯示不同教育程度組別中薪資是否大於 250 (二元結果) 的頻率分佈。
table(education, I(wage > 250))
```

education	FALSE	TRUE
1. < HS Grad	268	0
2. HS Grad	966	5
3. Some College	643	7
4. College Grad	663	22
5. Advanced Degree	381	45

```
# 擬合另一個邏輯斯迴歸 GAM 模型 (gam.lr.s)
# 但只使用【排除教育程度為 "1. < HS Grad"】的子集數據
gam.lr.s <- gam(
  I(wage > 250) ~ year + s(age, df = 5) + education ,
  family = binomial , data = Wage,
  subset = (education != "1. < HS Grad")
)
# 繪製在排除最低教育程度組別後的 GAM 模型的平滑項曲線
plot(gam.lr.s, se = T, col = "green")
```



8.3 Lab: Decision Trees

8.3.1 Fitting Classification Trees

我們首先使用分類樹分析 Carseats 資料集。在該資料集中銷售額是一個連續變量，因此我們首先將其重新編碼為二元變數。我們使用 `ifelse()` 函數建立一個名為 High 的變量，如果銷售額超過 8，則該變數的值為 Yes，否則為 No。

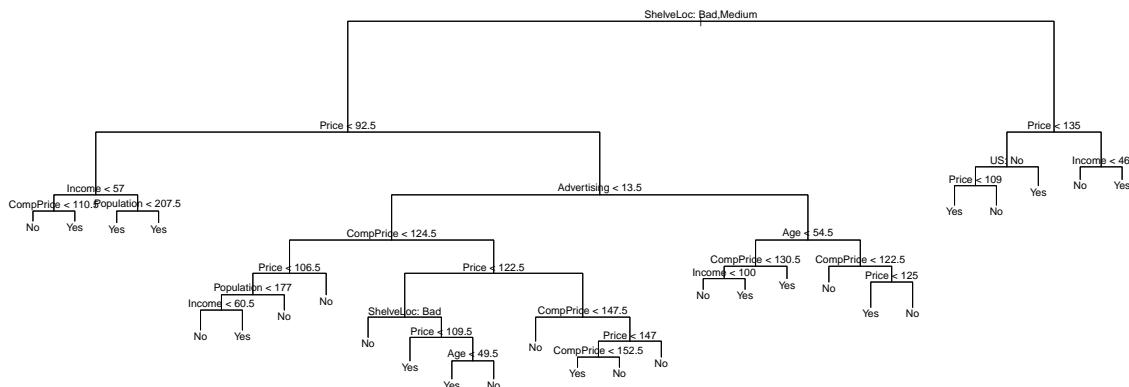
```
# 載入 'tree' 套件，用於擬合和處理決策樹模型的函式庫。
library(tree)
# 載入 'ISLR2' 套件，其中包含用於教學和練習的資料集。
library(ISLR2)
# 將 'Carseats' 資料集附加到 R 的搜尋路徑中，後續可以直接使用資料集中的變數名稱。
attach(Carseats)
# 創建一個新的【二元類別變數】
# 'High' (高銷量)：如果 Sales 銷售額大於 8 則為 "Yes"，否則為 "No"。
High <- factor(ifelse(Sales <= 8, "No", "Yes"))
# 將新創建的 'High' 變數新增回原始的 'Carseats' 資料集中。
Carseats <- data.frame(Carseats, High)
# 使用 tree() 函數擬合一個分類樹模型。
# 目標變數是 'High'，預測變數是【所有其他變數(.)】
# 排除 Sales (- Sales)】(因為 High 是基於 Sales 創建的)
```

```
tree.carseats <- tree(High ~ . - Sales, Carseats)
# 顯示擬合分類樹的摘要資訊
summary(tree.carseats)
```

Classification tree:

```
tree(formula = High ~ . - Sales, data = Carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"    "Price"        "Income"        "CompPrice"     "Population"
[6] "Advertising"   "Age"          "US"
Number of terminal nodes: 27
Residual mean deviance: 0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

```
# 樹狀圖最吸引人的特性之一是它們可以以圖形方式顯示
plot(tree.carseats)
# 繪製已擬合的分類樹的結構圖
text(tree.carseats, pretty = 0)
```



```
node), split, n, deviance, yval, (yprob)
* denotes terminal node

1) root 400 541.500 No ( 0.59000 0.41000 )
2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
4) Price < 92.5 46 56.530 Yes ( 0.30435 0.69565 )
8) Income < 57 10 12.220 No ( 0.70000 0.30000 )
```

```

16) CompPrice < 110.5 5  0.000 No ( 1.00000 0.00000 ) *
17) CompPrice > 110.5 5  6.730 Yes ( 0.40000 0.60000 ) *
9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
19) Population > 207.5 20  7.941 Yes ( 0.05000 0.95000 ) *
5) Price > 92.5 269  299.800 No ( 0.75465 0.24535 )
10) Advertising < 13.5 224  213.200 No ( 0.81696 0.18304 )
20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
160) Income < 60.5 6  0.000 No ( 1.00000 0.00000 ) *
161) Income > 60.5 6  5.407 Yes ( 0.16667 0.83333 ) *
81) Population > 177 26  8.477 No ( 0.96154 0.03846 ) *
41) Price > 106.5 58  0.000 No ( 1.00000 0.00000 ) *
21) CompPrice > 124.5 128  150.200 No ( 0.72656 0.27344 )
42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
84) ShelveLoc: Bad 11  6.702 No ( 0.90909 0.09091 ) *
85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
170) Price < 109.5 16  7.481 Yes ( 0.06250 0.93750 ) *
171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
343) Age > 49.5 11  6.702 No ( 0.90909 0.09091 ) *
43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
348) CompPrice < 152.5 7  5.742 Yes ( 0.14286 0.85714 ) *
349) CompPrice > 152.5 5  5.004 No ( 0.80000 0.20000 ) *
175) Price > 147 7  0.000 No ( 1.00000 0.00000 ) *
11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
89) Income > 100 5  0.000 Yes ( 0.00000 1.00000 ) *
45) CompPrice > 130.5 11  0.000 Yes ( 0.00000 1.00000 ) *
23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
46) CompPrice < 122.5 10  0.000 No ( 1.00000 0.00000 ) *
47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
94) Price < 125 5  0.000 Yes ( 0.00000 1.00000 ) *
95) Price > 125 5  0.000 No ( 1.00000 0.00000 ) *

```

```

3) ShelveLoc: Good 85 90.330 Yes ( 0.22353 0.77647 )
6) Price < 135 68 49.260 Yes ( 0.11765 0.88235 )
12) US: No 17 22.070 Yes ( 0.35294 0.64706 )
24) Price < 109 8 0.000 Yes ( 0.00000 1.00000 ) *
25) Price > 109 9 11.460 No ( 0.66667 0.33333 ) *
13) US: Yes 51 16.880 Yes ( 0.03922 0.96078 ) *
7) Price > 135 17 22.070 No ( 0.64706 0.35294 )
14) Income < 46 6 0.000 No ( 1.00000 0.00000 ) *
15) Income > 46 11 15.160 Yes ( 0.45455 0.54545 ) *

```

銷售額最重要的指標是貨架位置，因為第一個分支區分了「好」位置、「差」位置和「中等」位置。

如果我們直接輸入樹物件的名稱，R 會列印出與樹的每個分支對應的輸出。R 會顯示分割標準（例如，價格 < 92.5）、該分支中的觀測值數量、偏差、該分支的總體預測結果（是或否），以及該分支中取值為「是」和「否」的觀測值比例，通往終端節點的分支以星號標記。

為了正確評估分類樹在這些資料上的表現，我們必須估計測試誤差，而不僅僅是計算訓練誤差。我們將觀測值分為訓練集和測試集，使用訓練集建立樹，並在測試資料上評估其效能。

```

# 確保每次執行程式碼時隨機抽樣的結果都相同，以便結果可重現。
set.seed(2)
# 從 Carseats 資料集的行索引中，隨機抽取 200 個作為【訓練集 (Training Set)】的索引。
train <- sample(1:nrow(Carseats), 200)
# 創建【測試集 (Test Set)】選取未被抽中作為訓練集的剩餘數據
Carseats.test <- Carseats[-train, ]
# 提取測試集數據中對應的目標變數 'High' (高銷量) 實際值
High.test <- High[-train]
# 僅使用訓練集 (subset = train) 的數據進行訓練
tree.carseats <- tree(High ~ . - Sales, Carseats,
subset = train)
# 使用訓練好的模型對測試集進行預測，並要求輸出預測的類別標籤。
tree.pred <- predict(tree.carseats, Carseats.test,
type = "class")
# 建立一個【混淆矩陣 (Confusion Matrix)】
# 比較模型預測的結果 (tree.pred) 和實際結果 (High.test) 的分佈
table(tree.pred, High.test)

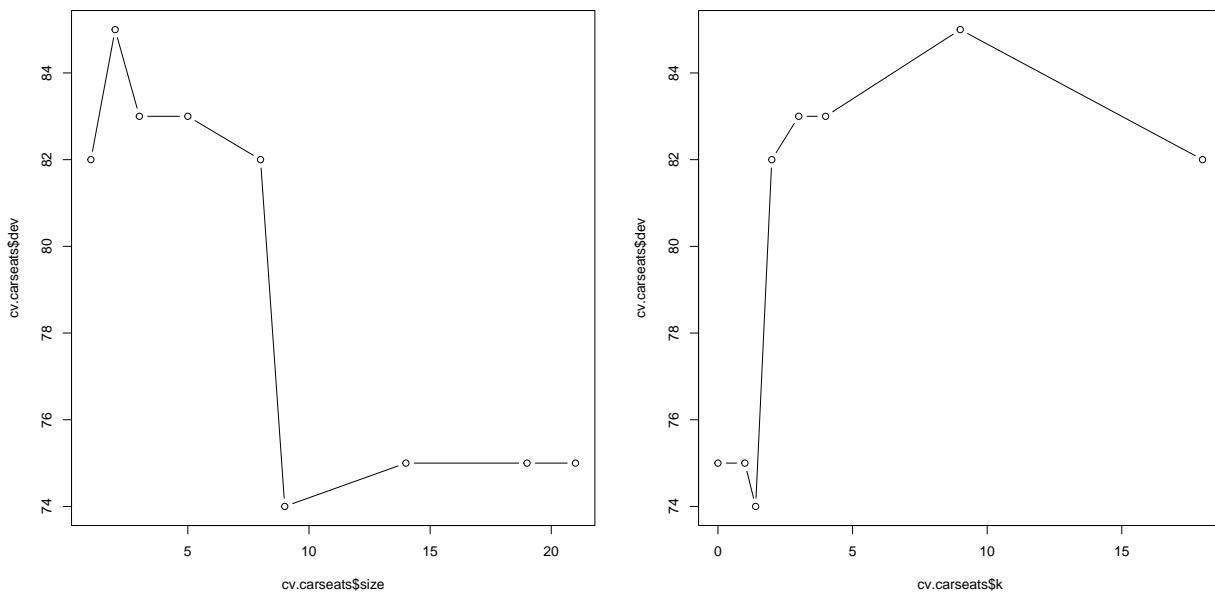
```

	High.test	
tree.pred	No	Yes
No	104	33
Yes	13	50

接下來，我們考慮剪枝是否能帶來更好的結果。`cv.tree()` 函數執行交叉驗證，以確定樹的最佳複雜

度；使用成本複雜度剪枝來選擇一系列待考慮的樹。

```
# 設定隨機數種子為 7，確保交叉驗證的結果是可重現的。  
set.seed(7)  
# 對完整的分類樹 (tree.carseats) 執行交叉驗證 (Cross-Validation)。  
# FUN = prune.misclass 指定使用 Misclassification Error  
# 作為修剪準則來評估每個子樹的大小。  
cv.carseats <- cv.tree(tree.carseats, FUN = prune.misclass)  
# 顯示交叉驗證結果 (cv.carseats) 清單中的所有元素名稱  
names(cv.carseats)  
  
[1] "size"    "dev"     "k"       "method"  
  
# 輸出交叉驗證結果的詳細列表  
cv.carseats  
  
$size  
[1] 21 19 14  9  8  5  3  2  1  
  
$dev  
[1] 75 75 75 74 82 83 83 85 82  
  
$k  
[1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0  
  
$method  
[1] "misclass"  
  
attr("class")  
[1] "prune"           "tree.sequence"  
  
# 將繪圖區域分為 1 行 2 列  
par(mfrow = c(1, 2))  
# 繪製【子樹大小 (size)】與【錯誤率 (dev)】關係圖，以線條和點 (type="b") 顯示。  
plot(cv.carseats$size, cv.carseats$dev, type = "b")  
# 繪製【成本複雜度參數 (k)】與【錯誤率 (dev)】關係圖  
plot(cv.carseats$k, cv.carseats$dev, type = "b")
```



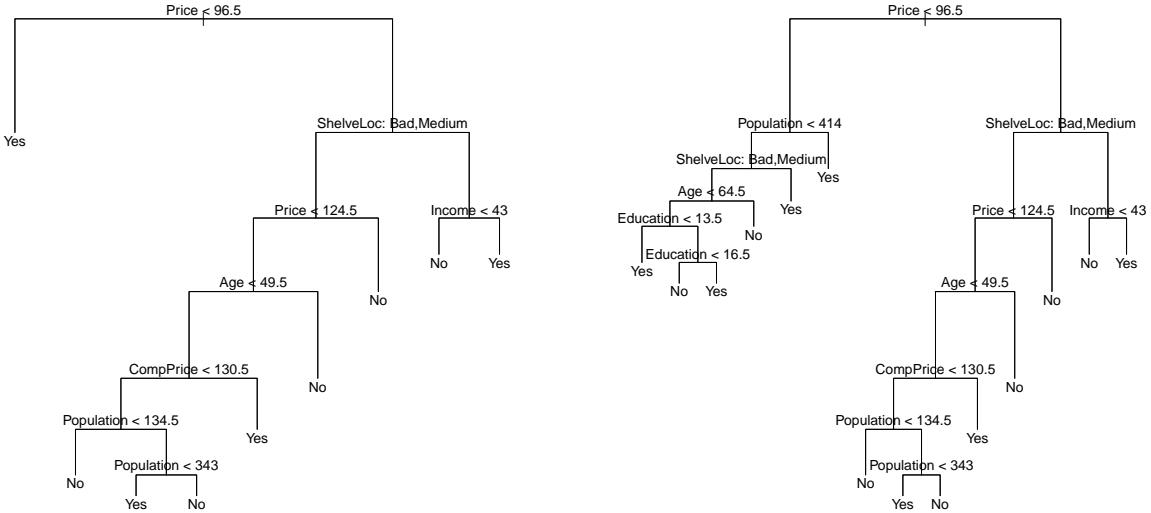
```
# 現在我們應用 prune.misclass() 函數來修剪樹，以獲得九節點樹。
# 使用分類錯誤率標準，將完整的樹修剪成具有 9 個終端節點 (best = 9) 的子樹。
prune.carseats <- prune.misclass(tree.carseats, best = 9)
# 繪製修剪後的 9 節點分類樹結構圖
plot(prune.carseats)
# 在 9 節點樹上添加文字標籤和分割條件
text(prune.carseats, pretty = 0)
# 使用 9 節點樹模型對測試集 (Carseats.test) 進行預測，輸出類別標籤。
tree.pred <- predict(prune.carseats, Carseats.test,
type = "class")
# 建立混淆矩陣，比較 9 節點樹模型的預測結果與實際測試集結果。
table(tree.pred, High.test)
```

		High.test	
		tree.pred	No Yes
No	97	25	
Yes	20	58	

```
# 計算 9 節點樹模型在測試集上的【分類準確率】(Accuracy)
# 即是正確預測的樣本數佔總樣本數的比例
(97 + 58) / 200
```

[1] 0.775

```
# 重新修剪完整的樹，這次選擇具有 14 個終端節點 (best = 14) 的子樹。
prune.carseats <- prune.misclass(tree.carseats, best = 14)
# 繪製修剪後的 14 節點分類樹結構圖。
plot(prune.carseats)
# 在 14 節點樹上添加文字標籤和分割條件。
text(prune.carseats, pretty = 0)
```



```
# 使用 14 節點樹模型對測試集進行預測，輸出類別標籤。
tree.pred <- predict(prune.carseats, Carseats.test,
type = "class")
# 建立混淆矩陣，比較 14 節點樹模型的預測結果與實際測試集結果。
table(tree.pred, High.test)
```

		High.test	
		tree.pred	No Yes
	No	102	31
	Yes	15	52

```
# 計算 14 節點樹模型在測試集上的【分類準確率】(Accuracy)
(102 + 52) / 200
```

[1] 0.77

8.3.2 Fitting Regression Trees

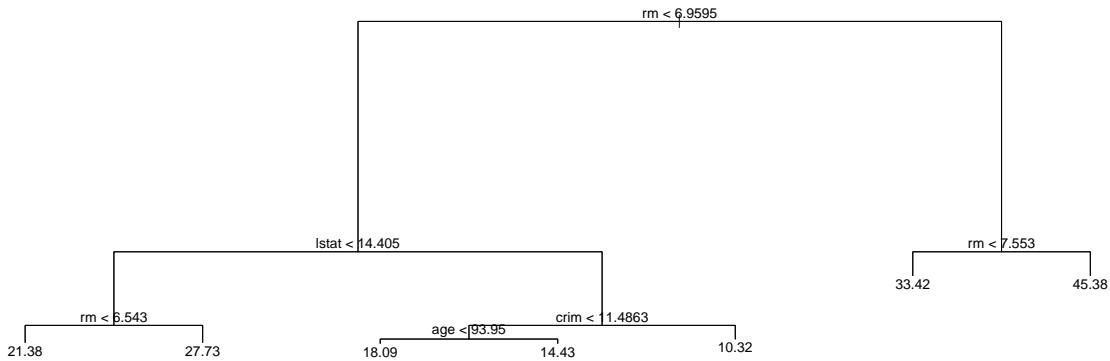
這裡我們對波士頓資料集擬合一個迴歸樹。首先，我們創建一個訓練集，並將迴歸樹擬合到訓練資料上。

```
# 確保每次執行程式碼時，隨機抽樣的結果都相同，以便結果可重現。
set.seed(1)
# 從 'Boston' 資料集的行索引中，隨機抽取一半的樣本作為訓練集 (Training Set)。
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
# 使用 tree() 函數擬合迴歸樹模型
# 目標變數是 'medv' (房價中位數)，預測變數是【所有其他變數(.)】
# 只使用訓練集 (subset = train) 的數據
tree.boston <- tree(medv ~ ., Boston, subset = train)
# 顯示擬合迴歸樹的摘要資訊
summary(tree.boston)
```

```
Regression tree:
tree(formula = medv ~ ., data = Boston, subset = train)
Variables actually used in tree construction:
[1] "rm"      "lstat"   "crim"    "age"
Number of terminal nodes:  7
Residual mean deviance:  10.38 = 2555 / 246
Distribution of residuals:
    Min.  1st Qu.  Median  Mean  3rd Qu.  Max.
-10.1800 -1.7770 -0.1775  0.0000  1.9230 16.5800
```

summary() 函數的輸出表明在建立決策樹時只使用了四個變數。在迴歸樹的脈絡中，偏差就是該決策樹的誤差平方和，現在我們繪製決策樹。

```
# 繪製已擬合的迴歸樹 (tree.boston) 的結構圖
plot(tree.boston)
# 在樹狀圖的節點和分支上添加文字標籤。
# pretty = 0 確保在每個終端節點上顯示準確的【預測值】(即該節點內 medv 的平均值)
text(tree.boston, pretty = 0)
```

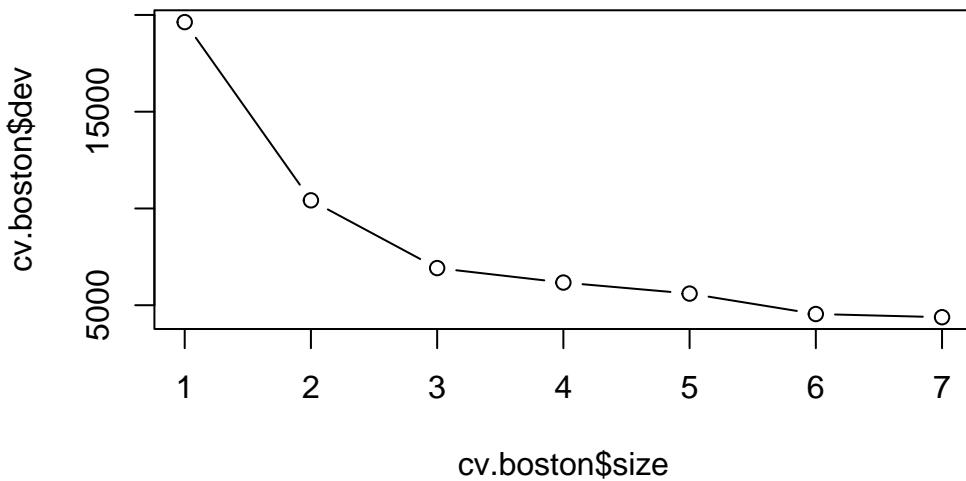


變數 $lstat$ 衡量社會經濟地位較低個體的百分比，而變數 rm 對應於平均房間數。決策樹表明， rm 值越大或 $lstat$ 值越小，對應的房屋價格越高。

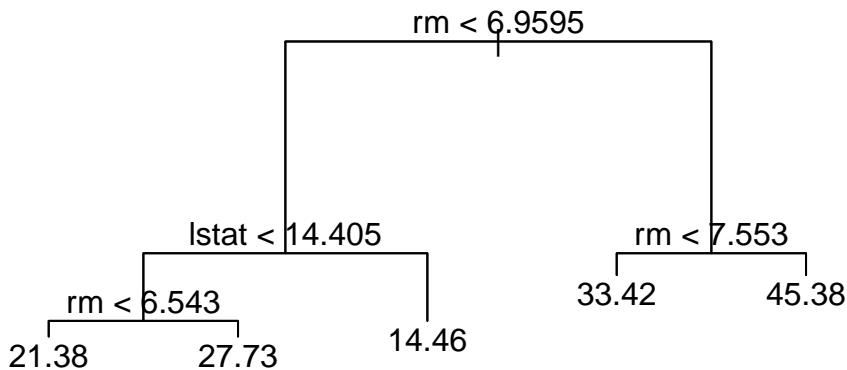
值得注意的是，我們可以透過將 `control = tree.control(nobs = length(train), mindev = 0)` 傳遞給 `tree()` 函數來建立一個更大的決策樹。現在我們使用 `cv.tree()` 函數來查看修剪決策樹是否能提高效能。

```

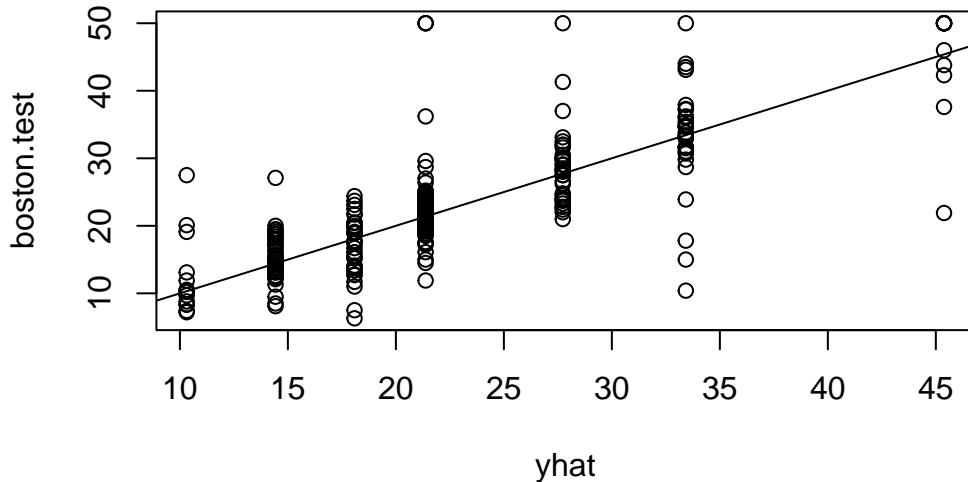
# 對完整的迴歸樹 (tree.boston) 執行交叉驗證 (Cross-Validation)。
# 預設使用【殘差平方和】作為評估準則
cv.boston <- cv.tree(tree.boston)
# 繪製【子樹大小 (size)】與【殘差偏差/錯誤率 (dev)】關係圖，用於選擇最佳子樹大小。
plot(cv.boston$size, cv.boston$dev, type = "b")
  
```



```
# 使用殘差平方和標準，將完整的迴歸樹修剪成具有 5 個終端節點 (best = 5) 的子樹。
prune.boston <- prune.tree(tree.boston, best = 5)
# 繪製修剪後的 5 節點迴歸樹結構圖
plot(prune.boston)
# 在 5 節點迴歸樹上添加文字標籤和分割條件，並顯示每個節點的預測值。
text(prune.boston, pretty = 0)
```



```
# 使用【未修剪】的完整迴歸樹 (tree.boston) 對【測試集】(Boston[-train, ]) 進行預測
yhat <- predict(tree.boston, newdata = Boston[-train, ])
# 提取測試集數據中對應的目標變數 'medv' (房價中位數) 的【實際值】
boston.test <- Boston[-train, "medv"]
# 繪製散點圖：X 軸為預測值 (yhat) · Y 軸為實際值 (boston.test)。
plot(yhat, boston.test)
# 在圖上繪製一條斜率為 1、截距為 0 的【理想線 (y=x)】
# 用於視覺化預測值與實際值的吻合程度
abline(0, 1)
```



```
# 計算測試集上的【均方誤差 (MSE)】
mean((yhat - boston.test)^2)
```

[1] 35.28688

換句話說，與迴歸樹相關的測試集均方誤差 (MSE) 為 35.29。因此 MSE 的平方根約為 5.941，顯示該模型的測試預測結果（平均而言）與人口普查區房屋中位數的真實值相差約 5,941 美元。

8.3.3 Bagging and Random Forests

這裡我們使用 R 語言中的 randomForest 包，對波士頓資料集應用 bagging 和隨機森林演算法。bagging 只是隨機森林的特例，其中 $m = p$ 。因此 randomForest() 函數既可以用來執行隨機森林，也可以用來執行 bagging。

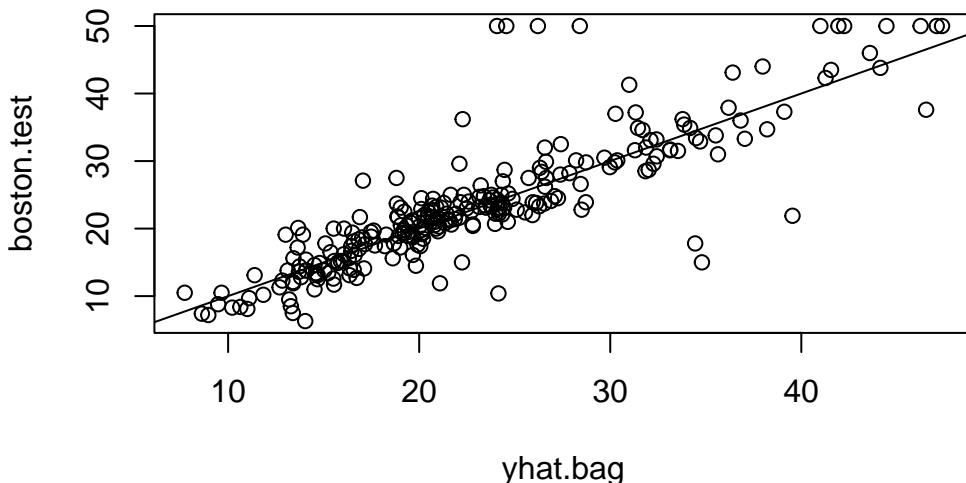
```
# 載入 'randomForest' 套件，這是用於 Bagging 和 Random Forest 模型的函式庫。
library(randomForest)
# 設定隨機數種子為 1，確保每次運行時模型結果的可重現性。
set.seed(1)
# 擬合第一個集成模型：執行 Bagging（套袋法）。
# Bagging 實質上是隨機森林的一個特例：mtry = 12（總變數個數，表示在每次分割時考慮所有預測變數）。
# importance = TRUE 表示計算並存儲變數重要性度量。
bag.boston <- randomForest(medv ~ ., data = Boston, subset = train, mtry = 12, importance = TRUE)
# 輸出 Bagging 模型的摘要資訊，包括樹的數量、解釋的變異量百分比等。
bag.boston
```

Call:

```
randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE, subset = train)
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 12

Mean of squared residuals: 11.40162
% Var explained: 85.17
```

```
# 使用 Bagging 模型對測試集 (Boston[-train, ]) 進行房價 (medv) 預測
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
# 繪製預測值 (yhat.bag) 與實際值 (boston.test) 的散點圖
plot(yhat.bag, boston.test)
# 在散點圖上繪製理想線 (y=x) 用於視覺化模型的預測準確度
abline(0, 1)
```



```
# 計算 Bagging 模型在測試集上的【均方誤差 (MSE)】
mean((yhat.bag - boston.test)^2)
```

[1] 23.41916

```
# 重新擬合 Bagging 模型，但這次將樹的數量 (ntree) 限制為 25 棵 (較少)。
bag.boston <- randomForest(medv ~ ., data = Boston,
subset = train, mtry = 12, ntree = 25)
```

```
# 使用新的 25 棵樹的 Bagging 模型對測試集進行預測
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
# 計算 25 棵樹的 Bagging 模型在測試集上的【均方誤差 (MSE)】
mean((yhat.bag - boston.test)^2)
```

[1] 25.75055

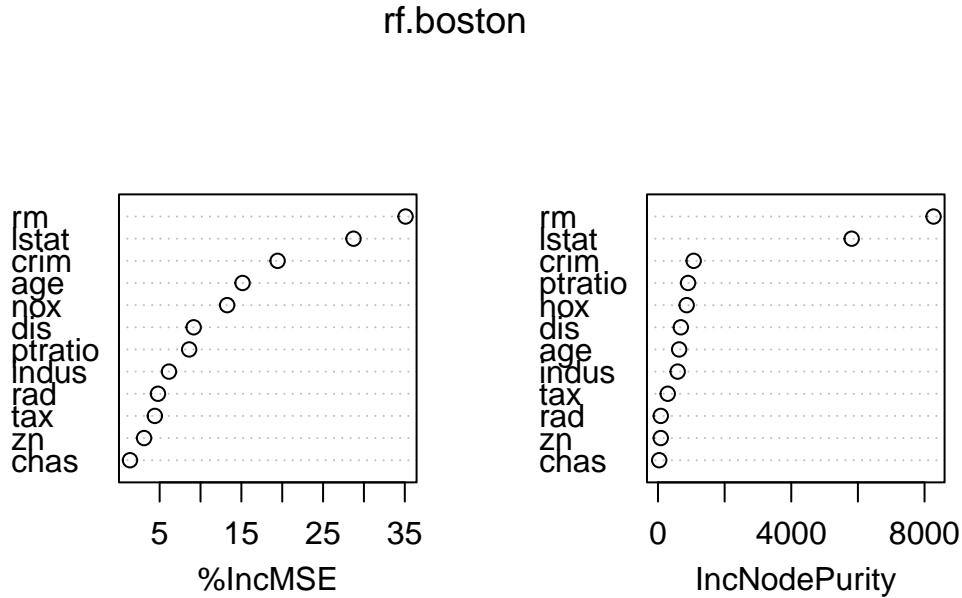
```
# 重新設定隨機數種子
set.seed(1)
# 擬合第二個集成模型：執行 Random Forest (隨機森林)。
# mtry = 6，總變數的平方根約等於 3.6，這裡使用 6 或 p/3
# 這是迴歸的常見選擇，表示每次分割時只考慮隨機抽取的 6 個變數
rf.boston <- randomForest(medv ~ ., data = Boston,
subset = train, mtry = 6, importance = TRUE)
# 使用隨機森林模型對測試集進行預測
yhat.rf <- predict(rf.boston, newdata = Boston[-train, ])
# 計算隨機森林模型在測試集上的【均方誤差 (MSE)】
mean((yhat.rf - boston.test)^2)
```

[1] 20.06644

```
# 輸出隨機森林模型中每個預測變數的【重要性度量】數值
importance(rf.boston)
```

	%IncMSE	IncNodePurity
crim	19.435587	1070.42307
zn	3.091630	82.19257
indus	6.140529	590.09536
chas	1.370310	36.70356
nox	13.263466	859.97091
rm	35.094741	8270.33906
age	15.144821	634.31220
dis	9.163776	684.87953
rad	4.793720	83.18719
tax	4.410714	292.20949
ptratio	8.612780	902.20190
lstat	28.725343	5813.04833

```
# 繪製隨機森林模型中預測變數的【重要性排序圖】(柱狀圖)
varImpPlot(rf.boston)
```



本文報告了兩種變數重要性量測。第一種基於給定變數置換後，out of bag samples 預測準確率的平均下降值。第二種測量是所有樹中，因該變數分裂而導致的節點不純度總下降值的平均值。對於回歸樹，節點不純度透過訓練 RSS 來衡量；對於分類樹，節點不純度透過偏差來衡量，可以使用 varImpPlot() 函數繪製這些重要性度量的圖表。

結果表明在隨機森林中考慮的所有決策樹中，社區財富 (lstat) 和房屋面積 (rm) 是迄今為止最重要的兩個變數。

8.3.4 Boosting

這裡我們使用 gbm 套件及其中的 gbm() 函數，將提升迴歸樹擬合到波士頓資料集。由於這是一個迴歸問題，我們使用選項 distribution = "gaussian" 來執行 gbm()；如果是二分類問題，則使用選項 distribution = "bernoulli"。

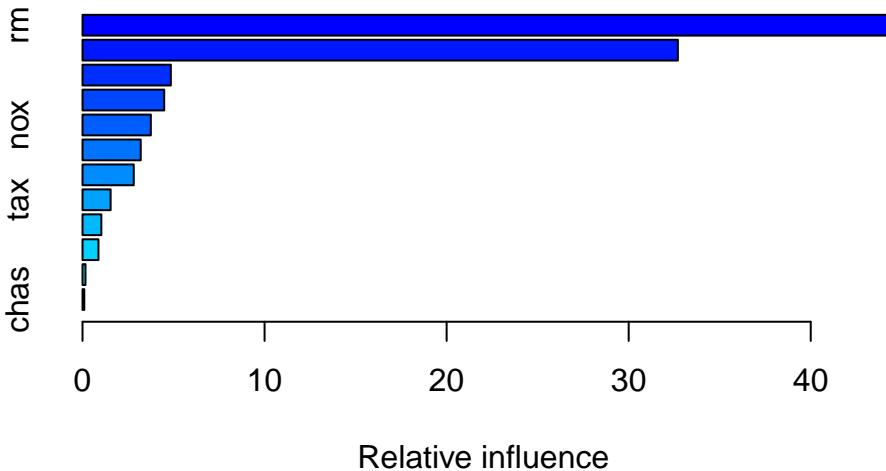
```
# 載入 'gbm' 套件，用於擬合梯度提升模型 (Gradient Boosting Machine) 的函式庫。
library(gbm)

# 設定隨機數種子為 1，確保每次運行時模型結果的可重現性。
set.seed(1)

# 擬合梯度提升模型 (Boosting)
# 目標變數 'medv' 對所有預測變數 '.'

# 僅使用訓練集 (Boston[train, ]) 進行訓練
# distribution = "gaussian" 指定這是【迴歸問題】
# n.trees = 5000 指定要建立的【樹的數量】(即提升迭代次數)
# interaction.depth = 4 指定每棵樹的【最大深度】(允許的最高階交互作用)
```

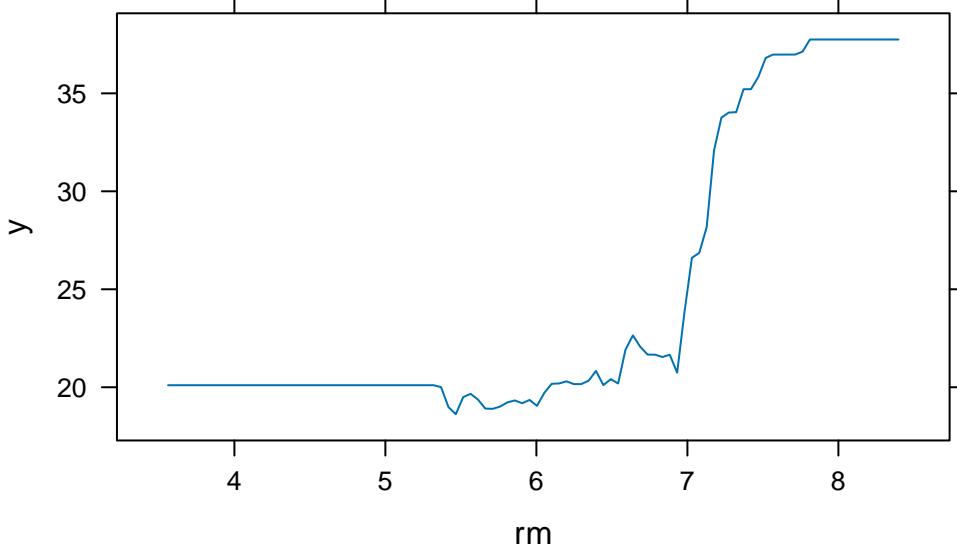
```
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
distribution = "gaussian", n.trees = 5000,
interaction.depth = 4)
# 輸出提升模型 (boost.boston) 的摘要資訊
summary(boost.boston)
```



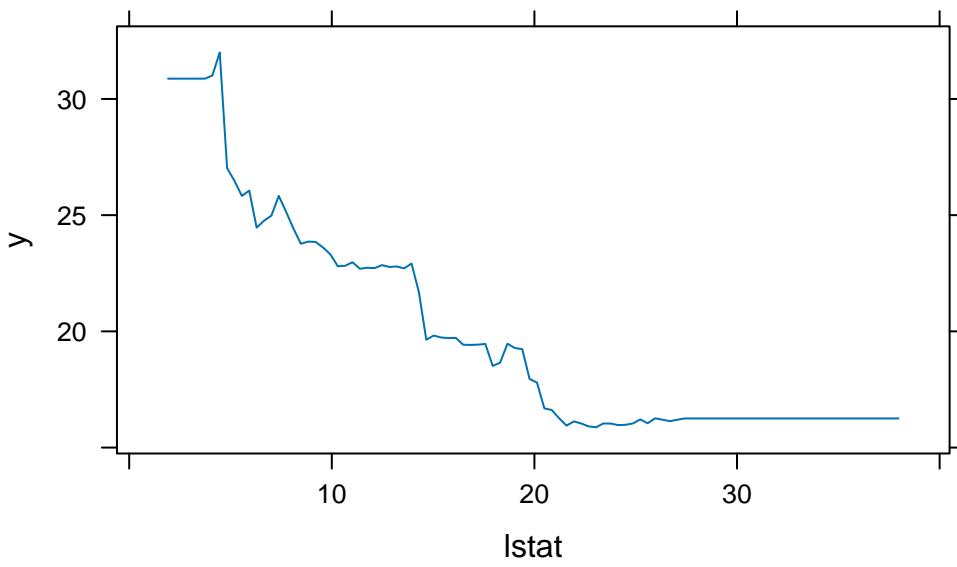
var	rel.inf
rm	rm 44.48249588
lstat	lstat 32.70281223
crim	crim 4.85109954
dis	dis 4.48693083
nox	nox 3.75222394
age	age 3.19769210
ptratio	ptratio 2.81354826
tax	tax 1.54417603
indus	indus 1.03384666
rad	rad 0.87625748
zn	zn 0.16220479
chas	chas 0.09671228

我們發現 lstat 和 rm 是迄今為止最重要的變數，我們也可以繪製這兩個變數的偏相關圖。這些圖表顯示了在剔除其他變數的影響後，所選變數對反應變數的邊際效應。在本例中，正如我們所預期的，房價中位數隨 rm 的增加而增加，並隨 lstat 的增加而減少。

```
# 繪製模型中變數 'rm' (平均房間數) 的【部分依賴圖 (Partial Dependence Plot)】  
# 顯示在控制其他變數不變的情況下，'rm' 如何影響預測目標 ('medv')。  
plot(boost.boston, i = "rm")
```



```
# 繪製提升模型中變數 'lstat' (低收入人口百分比) 的【部分依賴圖】  
# 顯示在控制其他變數不變的情況下，'lstat' 如何影響預測目標 ('medv')。  
plot(boost.boston, i = "lstat")
```



```
# 使用提升模型對測試集進行預測，指定使用全部 5000 棵樹來得到最終預測值。
yhat.boost <- predict(boost.boston,
newdata = Boston[-train, ], n.trees = 5000)
# 計算提升模型在測試集上的【均方誤差 (MSE)】
mean((yhat.boost - boston.test)^2)
```

[1] 18.39057

所得的測試均方誤差 (MSE) 為 18.39，優於隨機森林和 bagging 的測試 MSE。如果需要我們可以用不同的收縮參數 λ 值進行 boosting 操作。預設值為 0.001，這裡我們取 $\lambda = 0.2$ 。

```
# 擬合梯度提升模型 (Boosting)
# 在原模型的基礎上增加了 shrinkage = 0.2 (學習率)
# 上述用來控制每次迭代對模型的貢獻程度 (較小的收縮率通常提高準確性)
# 增加了 verbose = F，關閉模型訓練過程中的詳細輸出
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
distribution = "gaussian", n.trees = 5000,
interaction.depth = 4, shrinkage = 0.2, verbose = F)
# 使用這個優化後的提升模型對測試集進行預測，指定使用全部 5000 棵樹。
yhat.boost <- predict(boost.boston,
newdata = Boston[-train, ], n.trees = 5000)
# 計算優化後提升模型在測試集上的【均方誤差 (MSE)】
mean((yhat.boost - boston.test)^2)
```

[1] 16.54778

在這種情況下使用 $\lambda = 0.2$ 會導致測試均方誤差低於使用 $\lambda = 0.001$ 的情況。

每個圖的第一行都包含一個黑色方塊，代表根據與該統計量關聯的最優模型選擇的每個變數。

8.3.5 Bayesian Additive Regression Trees

在本節中，我們使用 BART 套件及其中的 gbart() 函數，將 Bayesian additive regression tree model 貝葉斯加性迴歸樹模型擬合到波士頓住房資料集。gbart() 函數專為定量結果變數設計，對於二元結果變量，可以使用 lbart() 和 pbart() 函數。

要執行 gbart() 函數，我們必須先建立訓練集和測試集的預測變數矩陣，我們使用預設設定運行 BART。

```
# 載入 'BART' 套件，這是用於擬合貝葉斯加法迴歸樹模型的核心函式庫。
library(BART)
# 從 'Boston' 資料集中選取前 12 欄 (即所有預測變數) 作為自變數矩陣 x
```

```

x <- Boston[, 1:12]
# 從 'Boston' 資料集中選取 'medv' (房價中位數) 作為目標變數向量 y
y <- Boston[, "medv"]
# 根據先前定義的訓練集索引 (train) · 從 x 中提取【訓練集自變數】。
xtrain <- x[train, ]
# 根據訓練集索引 · 從 y 中提取【訓練集目標變數】。
ytrain <- y[train]
# 根據先前定義的測試集索引 (-train) · 從 x 中提取【測試集自變數】。
xtest <- x[-train, ]
# 根據測試集索引 · 從 y 中提取【測試集目標變數】。
ytest <- y[-train]
# 設定隨機數種子為 1 · 確保 BART 模型的 MCMC 過程結果是可重現的。
set.seed(1)
# 擬合貝葉斯加法迴歸樹 (BART) 模型
# 使用訓練數據 (xtrain, ytrain) 進行訓練 · 並同時對測試數據進行預測。
bartfit <- gbart(xtrain, ytrain, x.test = xtest)

```

```

*****Calling gbart: type=1
*****Data:
data:n,p,np: 253, 12, 253
y1,yn: 0.213439, -5.486561
x1,x[n*p]: 0.109590, 20.080000
xp1,xp[np*p]: 0.027310, 7.880000
*****Number of Trees: 200
*****Number of Cut Points: 100 ... 100
*****burn,nd,thin: 100,1000,1
*****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.795495,3,3.71636,21.7866
*****sigma: 4.367914
*****w (weights): 1.000000 ... 1.000000
*****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,12,0
*****printevery: 100

```

MCMC

```

done 0 (out of 1100)
done 100 (out of 1100)
done 200 (out of 1100)
done 300 (out of 1100)
done 400 (out of 1100)
done 500 (out of 1100)

```

```
done 600 (out of 1100)
done 700 (out of 1100)
done 800 (out of 1100)
done 900 (out of 1100)
done 1000 (out of 1100)
time: 8s
trcnt,tecnt: 1000,1000
```

```
# 從 BART 模型結果中提取測試集預測值的【後驗平均值】(yhat.test.mean) 作為最終預測值
yhat.bart <- bartfit$yhat.test.mean
# 計算 BART 模型在測試集上的【均方誤差 (MSE)】
mean((ytest - yhat.bart)^2)
```

```
[1] 15.94718
```

```
# 根據模型輸出的【變數使用次數的平均值 (varcount.mean)】，創建一個降序排列的索引。
ord <- order(bartfit$varcount.mean, decreasing = T)
# 輸出變數使用次數的平均值，按照降序排列顯示，衡量【變數重要性】。
bartfit$varcount.mean[ord]
```

```
nox      lstat      tax      rad      rm      indus      chas      ptratio      age      zn
22.952   21.329   21.250   20.781   19.890   19.825   19.051   18.976   18.274   15.952
dis      crim
14.457   11.007
```