Bài giảng R, số 6
-Regression và Prediction-

TS.Tô Đức Khánh

21/04/2024

1 Các bước xử lý cơ bản với mô hình hồi quy

Mô hình hồi quy tuyến tính được dùng để tiên đoán giá trị của biến phản hồi (response variable) Y dựa trên cơ sở của p biến giải thích khác nhau: $X_1, X_2, \ldots, X_p$. Mô hình được phát biểu tổng quát như sau:

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_p x_{pi} + \varepsilon_i,$$

trong đó, $x_{1i}, x_{2i}, \ldots, x_{pi}$ là các giá trị quan sát của các biến giải thích $X_1, X_2, \ldots, X_p$, Y_i là quan sát của biến Y, và ε_i là thành phần sai số, với $i = 1, 2, \ldots, n$.

Trong `R`, để ước lượng mô hình hồi quy tuyến tính, ta sử dụng hàm `lm()`, cụ thể như sau:

```
lm(formula = y ~ x1 + x2 + ..., data = ..., ...)
```

trong đó,

- `formula = y ~ x1 + x2 + ...` là biểu thức tuyến tính của mô hình, với `y` là tên của biến đáp ứng, `x1` và `x2` lần lượt là tên của các biến giải thích;
- `data` là tên của bộ dữ liệu chứa các biến cần cho việc ước lượng mô hình;
- `...` là các đối thêm, được sử dụng trong các trường hợp cụ thể.

Sau khi ước lượng mô hình, ta sử dụng các hàm sau:

- `summary()`: in ra bảng tổng hợp kết quả tổng hợp của phân tích mô hình;
- `coef()`: truy xuất ra ước lượng hệ số của mô hình;
- `confint()`: xác định khoảng tin cậy cho hệ số trong mô hình;
- `predict()`: tiên đoán giá trị của biến đáp ứng dựa vào một hoặc nhiều giá trị được cho trước của các biến đáp ứng;
- `residuals()`: truy xuất thặng dư của mô hình;
- `rstandard()`: truy xuất thặng dư chuẩn hóa của mô hình.

Ví dụ 1: Xét bộ dữ liệu `Advertising.csv` chứa dữ liệu về doanh số bán hàng và chi phí dành cho các hình thức quảng cáo: tivi, radio, newspaper. Mục tiêu là xây dựng mô hình dự đoán doanh số bán sản phẩm dựa trên các khoản tiền đầu tư quảng cáo, qua đó tìm ra chiến lược quảng cáo trong thời gian sắp tới.

```
data_adv <- read_csv(file = "datasets/Advertising.csv")
data_adv <- data_adv |> janitor::clean_names()
glimpse(data_adv)
```

```
## Rows: 200
## Columns: 5
## $ id        <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19~
## $ tv        <dbl> 230.1, 44.5, 17.2, 151.5, 180.8, 8.7, 57.5, 120.2, 8.6, 199.8, 66~
## $ radio     <dbl> 37.8, 39.3, 45.9, 41.3, 10.8, 48.9, 32.8, 19.6, 2.1, 2.6, 5.8, 24~
## $ newspaper <dbl> 69.2, 45.1, 69.3, 58.5, 58.4, 75.0, 23.5, 11.6, 1.0, 21.2, 24.2, ~
## $ sales     <dbl> 22.1, 10.4, 9.3, 18.5, 12.9, 7.2, 11.8, 13.2, 4.8, 10.6, 8.6, 17.~
```

Biến phản hồi được quan tâm là `sales` - doanh số sản phẩm được bán ra. Mô hình được đề xuất:

$$\texttt{sales}_i = \beta_0 + \beta_1 \texttt{tv}_i + \beta_2 \texttt{radio}_i + \beta_3 \texttt{newspaper}_i + \varepsilon_i$$

Ta ước lượng mô hình này như sau:

```
md_adv <- lm(sales ~ tv + radio + newspaper, data = data_adv)
```

Kết quả tổng hợp

```
summary(md_adv)
```

```
##
## Call:
## lm(formula = sales ~ tv + radio + newspaper, data = data_adv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## tv           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

Một số nhận xét nhanh:

- Tăng chi phí quảng cáo TV lên một đơn vị (1000$), doanh số bán hàng tăng 45.8 đơn vị sản phẩm.
- Nếu xét theo chi phí quảng cáo trên Radio, mức tăng doanh số khi tăng chi phí quảng cáo lên một đơn vị (1000$) là 188.5 đơn vị sản phẩm.
- Trong khi đó, tăng 1000$ cho chi phí quảng cáo trên báo in thì mức tăng doanh số bán hàng lại là -1 đơn vị sản phẩm! Tại sao lại như vậy? Theo biểu đồ tương quan, mức tăng phải dương.

Tiếp theo, ta áp dụng phương pháp bootstrap để ước lượng khoảng tin cậy và kiểm định giả thuyết $\beta_j = 0$.

Trước tiên, ta định hàm `fun_boot_md()` để thực hiện ước tính mỗi lần lặp lấy mẫu:

```
fun_boot_md <- function(data, ind, formula, ...){
  data_new <- data[ind,]
  out_md <- lm(formula = formula, data = data_new, ...)
  return(out_md$coefficients)
}
```

```r
set.seed(84)
out_boot_md_adv <- boot(data = data_adv, statistic = fun_boot_md, R = 1000,
                        formula = sales ~ tv + radio + newspaper)
out_boot_md_adv
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data_adv, statistic = fun_boot_md, R = 1000, formula = sales ~
##     tv + radio + newspaper)
##
##
## Bootstrap Statistics :
##         original        bias    std. error
## t1*  2.938889369 -7.493837e-03 0.353895448
## t2*  0.045764645  6.690041e-05 0.001977460
## t3*  0.188530017 -2.028853e-04 0.010886865
## t4* -0.001037493  9.586706e-06 0.006899563
```

Thực hành 1: Hãy vẽ histogram của các kết quả ước lượng bootstrap của hệ số, và nhận xét về phân phối mẫu của chúng.

Ta sử dụng hàm `boot.ci()` để ước tính khoảng tin cậy 95%, của các hệ số, tương ứng với đối số `index`:

```r
boot.ci(out_boot_md_adv, index = 1, type = "perc", conf = 0.95)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = out_boot_md_adv, conf = 0.95, type = "perc",
##     index = 1)
##
## Intervals :
## Level     Percentile
## 95%   ( 2.175,  3.612 )
## Calculations and Intervals on Original Scale
```

```r
boot.ci(out_boot_md_adv, index = 2, type = "perc", conf = 0.95)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = out_boot_md_adv, conf = 0.95, type = "perc",
##     index = 2)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0420,  0.0497 )
## Calculations and Intervals on Original Scale
```

```r
boot.ci(out_boot_md_adv, index = 3, type = "perc", conf = 0.95)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = out_boot_md_adv, conf = 0.95, type = "perc",
##     index = 3)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.1661,  0.2089 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(out_boot_md_adv, index = 4, type = "perc", conf = 0.95)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = out_boot_md_adv, conf = 0.95, type = "perc",
##     index = 4)
##
## Intervals :
## Level     Percentile
## 95%   (-0.0153,  0.0128 )
## Calculations and Intervals on Original Scale
```

Ta tính p-value cho kiểm định giả thuyết $\beta_j = 0$, như sau:

```
pvals_adv2 <- sapply(1:ncol(out_boot_md_adv$t),function(x) {
  qt0 <- mean(out_boot_md_adv$t[, x] <= 0)
  if (qt0 < 0.5) {
    return(2*qt0)
  } else {
    return(2*(1 - qt0))
  }
})
pvals_adv2
```

```
## [1] 0.000 0.000 0.000 0.866
```

Giả sử ta có thông tin rằng, công ty sẽ đầu tư 150.000$, 40.000$ và 55.000$ cho quảng cáo lần lượt trên trên tivi, radio và newspaper. Khi đó, dựa vào mô hình, ta có thể ước tính được trung bình doanh số bán hàng là

```
predict(md_adv, newdata = data.frame(tv = 150, radio = 40, newspaper = 55))
```

```
##        1
## 17.28772
```

tức là khoảng 17.287 ngàn đơn vị sản phẩm. Và để tìm khoảng tin cậy cho giá trị trung bình doanh số bán hàng, ta sử dụng phương pháp bootstrap. Vì, ta đã chạy bootstrap ở phần trước với 1000 lần lặp để ước tính các hệ số, ta có thể sử dụng kết quả này để tính các giá trị ước đoán trung bình doanh số bán hàng trong mỗi lần lặp mẫu:

```
x_adv <- c(1, 150, 40, 55)
y_adv <- apply(out_boot_md_adv$t, 1, function(x){
  x_adv %*% x
})
quantile(y_adv, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 16.82280 17.73671
```

Như vậy, trung bình doanh số bán hàng được tiên lượng theo mô hình này, có giá trị thay đổi từ (16.823, 17.737), với độ tin cậy 95%, tương ứng với mức trung bình (16.823, 17.737) ngàn sản phẩm.

Ngoài ra, ta còn có thể sử dụng phương pháp bootstrap để ước lượng khoảng tiên đoán cho doanh số sản phẩm, theo mức đầu tư của công ty đề ra, dựa trên mô hình:

```
resid_adv <- residuals(md_adv)
y_adv_pd_pci <- y_adv + sample(resid_adv, size = 1000, replace = TRUE)
quantile(y_adv_pd_pci, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 13.57307 19.63043
```

Chú ý: hàm `residuals()` được dùng để tính thặng dư của mô hình tại mọi điểm quan sát của dữ liệu.

Để đánh giá mô hình ta sử dụng phương pháp cross-validation và chỉ số RMSE (root mean square error):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2}$$

Thực hành 2: Sử dụng ý tưởng về validation set, chia bộ dữ liệu thành hai phần (theo tỷ lệ phù hợp), hãy tính RMSE của mô hình dựa trên test set.

Thực hành 3: Hãy viết đoạn code để áp dụng phương pháp leave-one-out cross-validation nhằm đánh giá mô hình trong ví dụ.

Thực hành 4: Hàm `errorest()` của thư viện `ipred` được thiết kế để ước tính sai số của mô hình thông qua phương pháp cross-validation (mặc định 10-fold cross-validation). Hãy tìm hiểu hàm này, và áp dụng cho việc đánh giá mô hình trong ví dụ.

Bài tập 1: Trong hàm `lm()`, để nhập thành phần tương tác của hai biến, ta dùng toán tử `:`, ví dụ `x1:x2` ám chỉ sự tương tác của `x1` và `x2`. Hãy xây dựng mô hình có sự tương tác của `tv` và `radio`. Hãy thực hiện

- ước lượng mô hình,
- khoảng tin cậy cho hệ số,
- khoảng tin cậy cho trung bình doanh số bán hàng dựa vào mô hình,
- khoảng tiên đoán cho doanh số bán hàng dựa vào mô hình,
- đánh giá mô hình.

Bài tập 2: (về nhà) Xét file dữ liệu `house_sales.csv` cung cấp dữ liệu về giá bán nhà ở King County (Seattle), Washington. Hãy đọc dữ liệu và

- thực hiện các mô tả tổng hợp cho các biến được ghi chép trong dữ liệu,
- xây dựng các mô hình tiên đoán giá bán nhà (`adj_sale_price`) theo các biến được thu thập trong dữ liệu,
- thực hiện các thống kê suy luận cho mô hình vừa xây dựng,
- đánh giá các mô hình được đề xuất.

2 Lựa chọn mô hình

2.1 Hồi quy từng bước

Trong R, hàm `regsubsets()` của thư viện `leaps`, được thiết kế cho nhiệm vụ hồi quy từng bước nhằm tìm tập con tốt nhất của mô hình hồi quy.

```
regsubsets(x = ..., data = ..., weights = NULL, nvmax = 8,
           method = c("exhaustive", "backward", "forward", "seqrep"))
```

trong đó,

- `x`: công thức xác định mô hình hồi quy;
- `data`: dữ liệu chứa dữ liệu cho việc phân tích;
- `weights`: trọng số được sử dụng trong mô hình (dành cho mô hình hồi quy trọng số);
- `nvmax`: số biến tối đa chứa trong tập con được xét;
- `method`: phương pháp hồi quy từng bước, với, `"exhaustive"` là hồi quy từng bước hỗn hợp, `"backward"` là hồi quy lùi từng bước, `"forward"` là hồi quy tiến từng bước.

Ví dụ 2: Xét dữ liệu `Hitters` trong thư viện `ISLR2`, cung cấp thông tin về của 322 cầu thủ bóng rổ chuyên nghiệp của giải nhà nghề Mỹ từ năm 1986 - 1987.

```
data(Hitters, package = "ISLR2")
Hitters <- Hitters |> janitor::clean_names()
glimpse(Hitters)
```

```
## Rows: 322
## Columns: 20
## $ at_bat     <int> 293, 315, 479, 496, 321, 594, 185, 298, 323, 401, 574, 202, 418,~
## $ hits       <int> 66, 81, 130, 141, 87, 169, 37, 73, 81, 92, 159, 53, 113, 60, 43,~
## $ hm_run     <int> 1, 7, 18, 20, 10, 4, 1, 0, 6, 17, 21, 4, 13, 0, 7, 3, 20, 2, 6, ~
## $ runs       <int> 30, 24, 66, 65, 39, 74, 23, 24, 26, 49, 107, 31, 48, 30, 29, 20,~
## $ rbi        <int> 29, 38, 72, 78, 42, 51, 8, 24, 32, 66, 75, 26, 61, 11, 27, 15, 7~
## $ walks      <int> 14, 39, 76, 37, 30, 35, 21, 7, 8, 65, 59, 27, 47, 22, 30, 11, 73~
## $ years      <int> 1, 14, 3, 11, 2, 11, 2, 3, 2, 13, 10, 9, 4, 6, 13, 3, 15, 5, 12,~
## $ c_at_bat   <int> 293, 3449, 1624, 5628, 396, 4408, 214, 509, 341, 5206, 4631, 187~
## $ c_hits     <int> 66, 835, 457, 1575, 101, 1133, 42, 108, 86, 1332, 1300, 467, 392~
## $ c_hm_run   <int> 1, 69, 63, 225, 12, 19, 1, 0, 6, 253, 90, 15, 41, 4, 36, 3, 177,~
## $ c_runs     <int> 30, 321, 224, 828, 48, 501, 30, 41, 32, 784, 702, 192, 205, 309,~
## $ crbi       <int> 29, 414, 266, 838, 46, 336, 9, 37, 34, 890, 504, 186, 204, 103, ~
## $ c_walks    <int> 14, 375, 263, 354, 33, 194, 24, 12, 8, 866, 488, 161, 203, 207, ~
## $ league     <fct> A, N, A, N, N, A, N, A, N, A, A, N, N, A, N, A, N, A, A, N, N, A~
## $ division   <fct> E, W, W, E, E, W, E, W, W, E, E, W, E, E, E, W, W, W, W, W, E, W~
## $ put_outs   <int> 446, 632, 880, 200, 805, 282, 76, 121, 143, 0, 238, 304, 211, 12~
## $ assists    <int> 33, 43, 82, 11, 40, 421, 127, 283, 290, 0, 445, 45, 11, 151, 45,~
## $ errors     <int> 20, 10, 14, 3, 4, 25, 7, 9, 19, 0, 22, 11, 7, 6, 8, 0, 10, 16, 9~
## $ salary     <dbl> NA, 475.000, 480.000, 500.000, 91.500, 750.000, 70.000, 100.000,~
## $ new_league <fct> A, N, A, N, N, A, A, A, N, A, A, N, N, A, N, A, N, A, A, N, N, N~
```

Quan sát biến `salary` cung cấp thông tin về tiền lương của các cầu thủ, ta nhận thấy có các giá trị khuyết

```
sum(is.na(Hitters$salary))
```

```
## [1] 59
```

Như vậy, ta cần loại bỏ các giá trị bị khuyết này ra khỏi bộ dữ liệu, trước khi thực hiện phân tích:

```r
Hitters <- na.omit(Hitters)
glimpse(Hitters)
```

```
## Rows: 263
## Columns: 20
## $ at_bat     <int> 315, 479, 496, 321, 594, 185, 298, 323, 401, 574, 202, 418, 239,~
## $ hits       <int> 81, 130, 141, 87, 169, 37, 73, 81, 92, 159, 53, 113, 60, 43, 158~
## $ hm_run     <int> 7, 18, 20, 10, 4, 1, 0, 6, 17, 21, 4, 13, 0, 7, 20, 2, 8, 16, 3,~
## $ runs       <int> 24, 66, 65, 39, 74, 23, 24, 26, 49, 107, 31, 48, 30, 29, 89, 24,~
## $ rbi        <int> 38, 72, 78, 42, 51, 8, 24, 32, 66, 75, 26, 61, 11, 27, 75, 8, 22~
## $ walks      <int> 39, 76, 37, 30, 35, 21, 7, 8, 65, 59, 27, 47, 22, 30, 73, 15, 14~
## $ years      <int> 14, 3, 11, 2, 11, 2, 3, 2, 13, 10, 9, 4, 6, 13, 15, 5, 8, 1, 1, ~
## $ c_at_bat   <int> 3449, 1624, 5628, 396, 4408, 214, 509, 341, 5206, 4631, 1876, 15~
## $ c_hits     <int> 835, 457, 1575, 101, 1133, 42, 108, 86, 1332, 1300, 467, 392, 51~
## $ c_hm_run   <int> 69, 63, 225, 12, 19, 1, 0, 6, 253, 90, 15, 41, 4, 36, 177, 5, 24~
## $ c_runs     <int> 321, 224, 828, 48, 501, 30, 41, 32, 784, 702, 192, 205, 309, 376~
## $ crbi       <int> 414, 266, 838, 46, 336, 9, 37, 34, 890, 504, 186, 204, 103, 290,~
## $ c_walks    <int> 375, 263, 354, 33, 194, 24, 12, 8, 866, 488, 161, 203, 207, 238,~
## $ league     <fct> N, A, N, N, A, N, A, N, A, A, N, N, A, N, N, A, N, N, A, N, A, A~
## $ division   <fct> W, W, E, E, W, E, W, W, E, E, W, E, E, E, W, W, W, E, W, W, E, E~
## $ put_outs   <int> 632, 880, 200, 805, 282, 76, 121, 143, 0, 238, 304, 211, 121, 80~
## $ assists    <int> 43, 82, 11, 40, 421, 127, 283, 290, 0, 445, 45, 11, 151, 45, 290~
## $ errors     <int> 10, 14, 3, 4, 25, 7, 9, 19, 0, 22, 11, 7, 6, 8, 10, 16, 2, 5, 2,~
## $ salary     <dbl> 475.000, 480.000, 500.000, 91.500, 750.000, 70.000, 100.000, 75.~
## $ new_league <fct> N, A, N, N, A, A, A, N, A, A, N, N, A, N, N, A, N, N, N, N, A, A~
```

Ta sử dụng hàm `regsubsets()` để tìm tập biến tốt nhất (chứa tối đa 8 biến) cho mô hình dự đoán tiền lương của 263 cầu thủ bóng rổ chuyên nghiệp:

```r
out_subset_hitters <- regsubsets(x = salary ~ ., data = Hitters, nvmax = 8,
                                 method = "exhaustive")
```

Chú ý: cú pháp `"salary ~ ."` có nghĩa là mô hình hồi quy cho `salary` dựa trên tất cả các biến của dữ liệu.

Ta in kết quả lựa chọn biến

```r
summary(out_subset_hitters)
```

```
## Subset selection object
## Call: regsubsets.formula(x = salary ~ ., data = Hitters, nvmax = 8,
##     method = "exhaustive")
## 19 Variables  (and intercept)
##             Forced in Forced out
## at_bat          FALSE      FALSE
## hits            FALSE      FALSE
## hm_run          FALSE      FALSE
## runs            FALSE      FALSE
## rbi             FALSE      FALSE
## walks           FALSE      FALSE
## years           FALSE      FALSE
## c_at_bat        FALSE      FALSE
## c_hits          FALSE      FALSE
## c_hm_run        FALSE      FALSE
## c_runs          FALSE      FALSE
## crbi            FALSE      FALSE
## c_walks         FALSE      FALSE
```

```
## leagueN         FALSE      FALSE
## divisionW       FALSE      FALSE
## put_outs        FALSE      FALSE
## assists         FALSE      FALSE
## errors          FALSE      FALSE
## new_leagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          at_bat hits hm_run runs rbi walks years c_at_bat c_hits c_hm_run c_runs
## 1  ( 1 ) " "    " "  " "    " "  " " " " " "   " "    " "      " "    " "      " "
## 2  ( 1 ) " "    "*"  " "    " "  " " " " " "   " "    " "      " "    " "      " "
## 3  ( 1 ) " "    "*"  " "    " "  " " " " " "   " "    " "      " "    " "      " "
## 4  ( 1 ) " "    "*"  " "    " "  " " " " " "   " "    " "      " "    " "      " "
## 5  ( 1 ) "*"    "*"  " "    " "  " " " " " "   " "    " "      " "    " "      " "
## 6  ( 1 ) "*"    "*"  " "    " "  " " " " "*"   " "    " "      " "    " "      " "
## 7  ( 1 ) " "    "*"  " "    " "  " " " " "*"   " "    "*"      "*"    "*"      " "
## 8  ( 1 ) "*"    "*"  " "    " "  " " " " "*"   " "    " "      " "    "*"      "*"
##          crbi c_walks leagueN divisionW put_outs assists errors new_leagueN
## 1  ( 1 ) "*"  " "     " "     " "       " "      " "     " "    " "
## 2  ( 1 ) "*"  " "     " "     " "       " "      " "     " "    " "
## 3  ( 1 ) "*"  " "     " "     " "       "*"      " "     " "    " "
## 4  ( 1 ) "*"  " "     " "     "*"       "*"      " "     " "    " "
## 5  ( 1 ) "*"  " "     " "     "*"       "*"      " "     " "    " "
## 6  ( 1 ) "*"  " "     " "     "*"       "*"      " "     " "    " "
## 7  ( 1 ) " "  " "     " "     "*"       "*"      " "     " "    " "
## 8  ( 1 ) " "  "*"     " "     "*"       "*"      " "     " "    " "
```

Để biết được tập nào là tốt nhất trong số các tập $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_8$, ta sử dụng chỉ số Mallows' Cp:

```
sum_out_subset_hitters <- summary(out_subset_hitters)
sum_out_subset_hitters$cp
```

```
## [1] 104.281319  50.723090  38.693127  27.856220  21.613011  14.023870  13.128474
## [8]   7.400719
```

Kết quả cho thấy tập kết quả thứ 8 (chứa 8 biến) là tốt nhất. Ta có thể truy xuất kết quả các biến được lựa chọn:

```
sum_out_subset_hitters$which[which.min(sum_out_subset_hitters$cp), ]
```

```
## (Intercept)      at_bat        hits      hm_run        runs         rbi       walks
##        TRUE        TRUE        TRUE       FALSE       FALSE       FALSE        TRUE
##       years    c_at_bat      c_hits    c_hm_run      c_runs        crbi     c_walks
##       FALSE       FALSE       FALSE        TRUE        TRUE       FALSE        TRUE
##     leagueN   divisionW    put_outs     assists      errors new_leagueN
##       FALSE        TRUE        TRUE       FALSE       FALSE       FALSE
```

Trong kết quả này, `TRUE` có nghĩa là được chọn, `FALSE` có nghĩa là không được chọn.

Kết quả ước lượng hệ số của mô hình tương ứng với tập này bởi đoạn lệnh:

```
coef(out_subset_hitters, which.min(sum_out_subset_hitters$cp))
```

```
##  (Intercept)       at_bat         hits        walks     c_hm_run       c_runs
##  130.9691577   -2.1731903    7.3582935    6.0037597    1.2339718    0.9651349
##      c_walks    divisionW     put_outs
##   -0.8323788 -117.9657795    0.2908431
```

Thực hành 6: Hãy thử với các phương pháp `"backward"` và `"forward"`, cùng với chỉ số Mallows' Cp, cho biết kết quả thu được.

Thực hành 7: Hàm `summary()` cũng trả về kết quả `bic` (chỉ số BIC) và `adjr2` (Adjusted R^2). Hãy sử dụng các chỉ số này để tìm ra tập con tốt nhất cho mô hình.

Thực hành 8: Thay đổi đối số `nvmax` để có tập con chứa số lựa biến lớn hơn.

2.2 Hồi quy từng bước và cross-validation

Trong phần này, ta sử dụng cross-validation để lựa chọn mô hình con tốt nhất trong bộ $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_p$. Do

```
summary(out_subset_hitters)
```

không trả về kết quả của cross-validated error cho mỗi mô hình $\mathcal{M}_j$, nên ta cần viết một đoạn lệnh (hàm) để tính thông số này. Ta có định nghĩa k-fold cross-validated error cho mỗi mô hình $\mathcal{M}_j$:

$$\begin{aligned} \text{CV}_{(k)}^{(j)} &= \frac{1}{k}\sum_{r=1}^{k}\text{RMSE}_r^{(j)} \\ &= \frac{1}{k}\sum_{r=1}^{k}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_{i,r} - \widehat{Y}_{i,r}^{(j)}\right)^2} \end{aligned}$$

trong đó,

- $Y_{i,r}$ là các quan sát của Y trong fold thứ r;
- $\widehat{Y}_{i,r}^{(j)}$ là tiên đoán của Y dựa trên mô hình $\mathcal{M}_j$ và dữ liệu trong fold thứ r.

Ta viết hàm `predict.regsubsets()` dưới đây để tính giá trị tiên đoán Y dựa trên mô hình $\mathcal{M}_j$ (kết quả từ `regsubsets()`) và dữ liệu trong fold thứ r.

```
predict.regsubsets <- function(object, newdata, id_model){
  form <- as.formula(object$call[[2]])
  x_mat <- model.matrix(form, newdata)
  coef_est <- coef(object, id = id_model)
  x_vars <- names(coef_est)
  res <- x_mat[, x_vars] %*% coef_est
  return(as.numeric(res))
}
```

Chú ý: cách viết hàm `a.x()` là một cách viết hàm kế thừa trong `R`, với

- phần tên trước dấu `"."` là tên của hàm thực thi tính toán, tức hàm `a()`;
- phần tên sau dấu `"."` là tên của lớp của kết quả được áp dụng hàm `a()`.

Ví dụ, tới hàm `predict.regsubsets()`:

- hàm thực thi tính toán là `predict()`, lớp hàm thực thi việc tiên đoán;
- lớp của kết quả được áp dụng là lớp `"regsubsets"`.

Bây giờ, ta sẽ thực hiện 5-fold cross-valiation để tìm tập con biến tốt nhất cho mô hình.

Bước 1: ta định nghĩa k và vector chỉ số fold cho từng quan sát trong bộ dữ liệu

```
n_hitter <- nrow(Hitters)
k <- 5
```

```r
set.seed(21)
folds <- sample(rep(1:k, length = n_hitter))
```

vector `folds` chứa thông tin về fold của từng quan sát, ví dụ những quan sát ở vị trí sau sẽ thuộc vào fold thứ 1

```r
which(folds == 1)
```

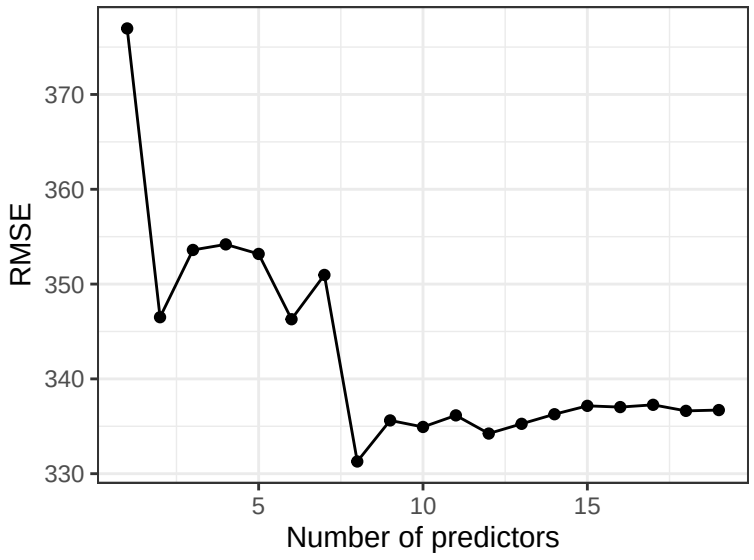
```
##  [1]   1   3   4  10  15  20  21  23  24  32  33  37  39  45  46  49  51  55  56  57
## [21]  62  66  70  72  73  81  82  83  90  97  99 103 109 114 120 121 123 124 131 141
## [41] 147 149 171 181 183 186 192 204 205 231 244 251 254
```

Bước 2: ta tính 5-fold cross-validated error cho 19 mô hình con, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{19}$ (tương ứng tổng số biến chứa trong mô hình)

```r
cv_error_hitters_rj <- matrix(0, nrow = k, ncol = 19)
for(r in 1:k){
  Hitters_train_r <- Hitters[folds != r, ]
  Hitters_test_r <- Hitters[folds == r, ]
  out_subset_hitters_folds <- regsubsets(x = salary ~ ., data = Hitters_train_r,
                                         method = "exhaustive", nvmax = 19)
  for(j in 1:19){
    pred_rj <- predict(out_subset_hitters_folds,
                       newdata = Hitters_test_r, id_model = j)
    cv_error_hitters_rj[r, j] <- sqrt(mean((Hitters_test_r$salary - pred_rj)^2))
  }
}
cv_error_hitters <- colMeans(cv_error_hitters_rj)
```

Ta có thể biểu diễn kết quả 5-fold cross-validated error cho 19 mô hình con

```r
ggplot(data = data.frame(x = c(1:19), y = cv_error_hitters),
       mapping = aes(x = x, y = y)) +
  geom_point() +
  geom_line() +
  labs(x = "Number of predictors", y = "RMSE") +
  theme_bw()
```



Giá trị nhỏ nhất của 5-fold cross-validated error tương ứng với mô hình có số biến là 8

```
which.min(cv_error_hitters)
```

```
## [1] 8
```

Bước 3: ta thực hiện `regsubsets()` và xác định mô hình con với 8 biến hồi quy

```
out_subset_hitters_2 <- regsubsets(x = salary ~ ., data = Hitters,
                                   method = "exhaustive", nvmax = 19)
coef(out_subset_hitters_2, id = which.min(cv_error_hitters))
```

```
##  (Intercept)       at_bat         hits        walks     c_hm_run       c_runs
##  130.9691577   -2.1731903    7.3582935    6.0037597    1.2339718    0.9651349
##      c_walks    divisionW     put_outs
##   -0.8323788 -117.9657795    0.2908431
```

Thực hành 9: Lặp lại nhiều lần ví dụ trên, với `set.seed()`, lưu lại kết quả và nhận xét sự thay đổi. Đưa ra một tập hợp biến chung nhất cho các lần lặp lại.

Thực hành 10: Thực hiện 10-fold cross-validation.

Thực hành 11: Thực hiện cross-validation.

2.3 Phương pháp co hệ số

Ta sẽ sử dụng hàm `glmnet()` trong thư viện `glmnet` để thực hiện ridge regression và lasso regression. Đối với phương pháp SCAD ta có thể sử dụng hàm `ncvreg()` trong thư viện `ncvreg`.

Cụ thể

```
glmnet(x, y, alpha, lambda, family = "gaussian")
```

trong đó

- `x` là ma trận chứa các biến hồi quy có thể trong mô hình;
- `y` là vector chứa giá trị của biến đáp ứng trong mô hình;
- `alpha` là số, có giá trị `"0"` tương ứng với phương pháp ridge regression, có giá trị `"1"` tương ứng với phương pháp lasso regression;
- `lambda` là vector chứa các giá trị của turning parameter λ;
- đối số `family = "gaussian"` tương ứng với mô hình hồi quy tuyến tính.

Ví dụ 3: Ta xét lại mô hình hồi quy tuyến tính dự đoán tiền lương của các cầu thủ bóng rổ chuyên nghiệp tại Mỹ, dựa trên bộ dữ liệu `Hitters` trong thư viện `ISLR2`. Đầu tiên, ta định nghĩa ma trận chứa các biến có thể trong mô hình, và vector của biến đáp ứng.

```
x_hitters <- model.matrix(salary ~ ., data = Hitters)[, -1]
y_hitters <- Hitters$salary
```

Ở đây, ta dùng hàm `model.matrix()` để truy xuất ma trận thiết kế của mô hình.

```
library(glmnet)
```

Ta tạo vector gồm các giá trị cho turning parameter λ.

```
lambda_grid <- 10^seq(from = 10, to = -2, length = 100)
```

bằng cách tạo này, ta có 100 giá trị của λ giảm dần từ 10^{10} tới 10^{-2}.

```
ridge_md <- glmnet(x = x_hitters, y = y_hitters, alpha = 0,
                   lambda = lambda_grid, family = "gaussian")
```

Nếu sử dụng

```
coef(ridge_md)
```

ta sẽ thu được ma trận gồm 20 dòng và 100 cột, tương ứng với ước lượng hệ số của các biến trong mô hình (dòng) theo từng giá trị của turning parameter λ (cột). Ví dụ

```
coef(ridge_md)[, 65]
```

```
##   (Intercept)        at_bat          hits        hm_run          runs           rbi
##  10.368086610   0.003859899   1.090128700  -0.074578917   1.129710071   0.866429174
##         walks         years      c_at_bat        c_hits      c_hm_run       c_runs
##   1.887887126  -0.509625577   0.010812515   0.068237280   0.470146993   0.136628201
##          crbi       c_walks       leagueN     divisionW      put_outs       assists
##   0.146475511   0.015680017  29.543809988 -96.632937860   0.201768977   0.049763941
##        errors   new_leagueN
##  -2.024038348   5.827331162
```

cung cấp ước lượng hệ số của các biến trong mô hình tương ứng với λ thứ 65

```
lambda_grid[65]
```

```
## [1] 174.7528
```

Đặc biệt, ta có thể xác định được ước lượng hệ số tương ứng với một giá trị λ bất kỳ, bằng cách sử dụng hàm

```
predict(object, s, type = "coefficients")
```

Ví dụ, với $\lambda = 20$, ta thu được

```
predict(ridge_md, s = 20, type = "coefficients")
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  9.346990e+01
## at_bat      -8.204203e-01
## hits         3.150678e+00
## hm_run      -1.243330e+00
## runs         8.963224e-01
## rbi          6.508750e-01
## walks        3.654696e+00
## years       -9.803444e+00
## c_at_bat    -6.435125e-03
## c_hits       1.462707e-01
## c_hm_run     7.117256e-01
## c_runs       3.408125e-01
## crbi         2.751881e-01
## c_walks     -3.305205e-01
## leagueN      5.525047e+01
## divisionW   -1.235682e+02
## put_outs     2.674667e-01
## assists      1.889136e-01
## errors      -3.767488e+00
## new_leagueN -2.062228e+01
```

Thực hành 12: Sử dụng hàm `glmnet()` để thực hiện lasso regression.

Bây giờ, ta sẽ tiến hành lựa chọn λ tốt nhất, bằng cách kết hợp validation set và cross-validation. Cụ thể, ta tiến hành các bước như sau:

1. sử dụng hàm `cv.glmnet()` để thực hiện k-fold cross-validation cho lasso regression hoặc ridge regression;
2. xác định giá trị λ sao cho sai số của k-fold cross-validation là nhỏ nhất;
3. xác định ước lượng hệ số tương ứng với giá trị λ tìm được ở bước 2 với `glmnet()`, dựa trên toàn bộ dữ liệu.

Cụ thể, ta thực hiện các bước trên với dữ liệu `Hitter`.

Bước 1, 2: ta thực hiện 5-fold cross-validation cho lasso regression

```
set.seed(24)
out_cv_lasso <- cv.glmnet(x = x_hitters, y = y_hitters, alpha = 1,
                          type.measure = "mse", nfolds = 5,
                          family = "gaussian")
print(out_cv_lasso)
```

```
##
## Call:  cv.glmnet(x = x_hitters, y = y_hitters, type.measure = "mse",      nfolds = 5, alpha = 1, fam
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min   1.68    55  120836 31074      17
## 1se 110.51    10  148287 29614       4
```

Kết quả cho thấy λ tối ưu là 1.68, tương ứng với bình phương sai số (mean squared error) là 120836. Đồng thời, ta thu được 17 hệ số khác 0.

Bước 3: sử dụng λ tối ưu vừa thu được, ta ước lượng các hệ số trong mô hình sử dụng toàn bộ dữ liệu

```
beta_lambda_lasso <- out_cv_lasso$lambda.min
out_lasso_md <- glmnet(x = x_hitters, y = y_hitters, alpha = 1,
                       lambda = lambda_grid, family = "gaussian")
predict(out_lasso_md, s = beta_lambda_lasso, type = "coefficients")
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  144.35148992
## at_bat        -1.81319850
## hits           6.26963170
## hm_run         0.40370469
## runs          -0.14010432
## rbi            .
## walks          5.20920790
## years         -9.73025817
## c_at_bat      -0.01814178
## c_hits         .
## c_hm_run       0.48938908
## c_runs         0.83325390
## crbi           0.43797222
## c_walks       -0.64469631
## leagueN       37.00321617
## divisionW   -118.44129514
## put_outs       0.27954386
## assists        0.23697293
```

```
## errors        -2.52101705
## new_leagueN   -2.29413069
```

Kết quả cho thấy 2 biến hồi quy `rbi` và `c_hits` là có hệ số ước lượng $= 0$. Như vậy, ta có thể sử dụng một mô hình hồi quy với 17 biến, ngoại trừ `rbi` và `c_hits`.

Thực hành 13: Lặp lại nhiều lần ví dụ trên, với `set.seed()`, lưu lại kết quả và nhận xét sự thay đổi. Đưa ra một tập hợp biến chung nhất cho các lần lặp lại.

Thực hành 14: Thực hiện 10-fold cross-validation.

Thực hành 15: Thực hiện 5-fold cross-validation với ridge regression, và so sánh với kết quả của lasso regression (có biến nào có hệ số $= 0$).

3 Chuẩn đoán thặng dư mô hình

Trong phần này, ta thực hiện một số chuẩn đoán cho mô hình:

- chuẩn đoán tính tuyến tính của mô hình;
- tính đồng nhất phương sai của thặng dư;
- tính độc lập của các quan sát;
- các giá trị ngoại lai xuất hiện trong mô hình.

Để xác định thặng dư của mô hình hồi quy tuyến tính, ta sử dụng hàm

```
residuals(object, type = c("working", "partial"))
```

trong đó

- `object` là kết quả của mô hình được ước lượng;
- `type` là dạng của thặng dư muốn xác định, `"working"` xác định bởi $Y_i - \widehat{Y}_i$, `"partial"` tương ứng với thặng dư từng phần.

3.1 Kiểm tra tính tuyến của mô hình

Mô hình hồi quy tuyến tính được phát biểu tổng quát như sau:

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_p x_{pi} + \varepsilon_i.$$

Để kiểm tra sự hợp lý của tính tuyến tính của mô hình, ta sử dụng biểu đồ Residuals vs Fitted, trong đó, thặng dư được xác định ở dạng `"working"`. Biểu đồ được xác định đơn giản thông qua `ggplot`:

```
ggplot(data = object, mapping = aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "Fitted values", y = "Residuals") +
  theme_bw()
```

trong đó

- `object` là kết quả của mô hình được ước lượng;
- `.fitted` là $\widehat{Y}_i$;
- `.resid` là thặng dư dạng `"working"` của mô hình;

- câu lệnh `geom_smooth(method = "loess", se = FALSE)` là để vẽ đường cong theo xu hướng của điểm dữ liệu.

Ví dụ 4: Xét lại bộ dữ liệu `Advertising.csv` chứa dữ liệu về doanh số bán hàng và chi phí dành cho các hình thức quảng cáo: tivi, radio, newspaper. Ta đánh giá mô hình hồi quy:

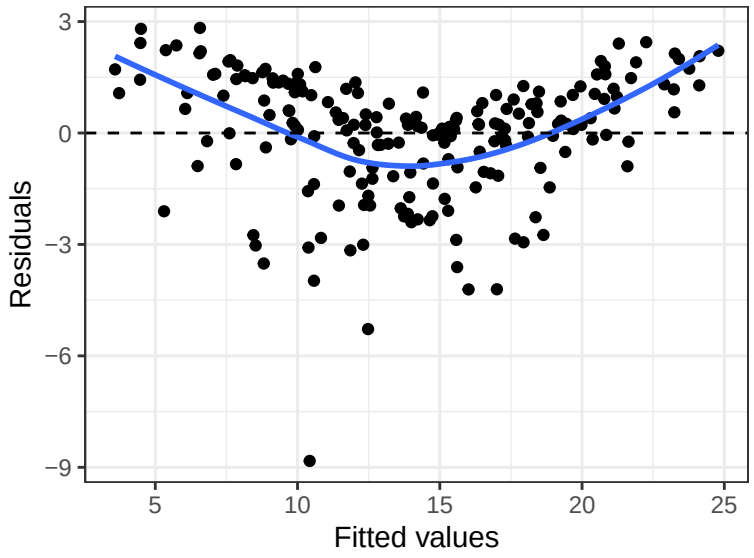
$$\texttt{sales}_i = \beta_0 + \beta_1 \texttt{tv}_i + \beta_2 \texttt{radio}_i + \beta_3 \texttt{newspaper}_i + \varepsilon_i$$

Ước lượng mô hình

```
md_adv <- lm(sales ~ tv + radio + newspaper, data = data_adv)
```

Tiếp theo, ta vẽ biểu đồ thặng dư của mô hình như sau:

```
ggplot(data = md_adv, mapping = aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "Fitted values", y = "Residuals") +
  theme_bw()
```



Thực hành 16: Hãy vẽ biểu đồ Residuals vs Fitted cho mô hình hồi quy có sự tương tác giữa `tv` và `radio`, và cho nhận xét.

Thực hành 17 Hãy vẽ biểu đồ Residuals vs Fitted cho mô hình hồi quy có thể được xây dựng để dựa đoán tiền lương của các cầu thủ bóng rổ chuyên nghiệp, theo dữ liệu `Hitters` trong thư viện `ISLR2`, và cho nhận xét.

3.2 Kiểm tra tính tuyến tính từng phần

Để kiểm tính tuyến tính từng phần, hay mối liên hệ giữa Y và X_j có thực sự tuyến tính, ta sử dụng biểu đồ thặng dư từng phần (partial residual plots).

Ví dụ 5: Xét lại mô hình hồi quy

$$\texttt{sales}_i = \beta_0 + \beta_1 \texttt{tv}_i + \beta_2 \texttt{radio}_i + \beta_3 \texttt{newspaper}_i + \varepsilon_i$$

ta muốn đánh giá tính tuyến tính của thành phần `tv` trong mô hình.

Bước 1 ta cần ước lượng thành phần $\widehat{\beta}_1 \texttt{tv}_i$, bằng hàm

```
predict(object, type = "terms")
```

cụ thể

```
terms_md_adv <- predict(md_adv, type = "terms")
```

kết quả thu được sẽ một bảng kết quả cho từng thành phần tuyến tính, `tv`, `radio`, `newspaper`, có mặt trong mô hình:

```
head(terms_md_adv)
```

```
##           tv     radio   newspaper
## 1  3.8010970  2.740472 -0.04009496
## 2 -4.6928212  3.023267 -0.01509137
## 3 -5.9421960  4.267565 -0.04019871
## 4  0.2039959  3.400327 -0.02899378
## 5  1.5449000 -2.349838 -0.02889003
## 6 -6.3311955  4.833156 -0.04611242
```

Bước 2 tiếp theo, ta xác định thặng dư từng phần

```
partial_resid_md_adv <- residuals(md_adv, type = "partial")
```

kết quả thu được sẽ một bảng kết quả thặng dư cho từng thành phần tuyến tính, `tv`, `radio`, `newspaper`, có mặt trong mô hình:

```
head(partial_resid_md_adv)
```

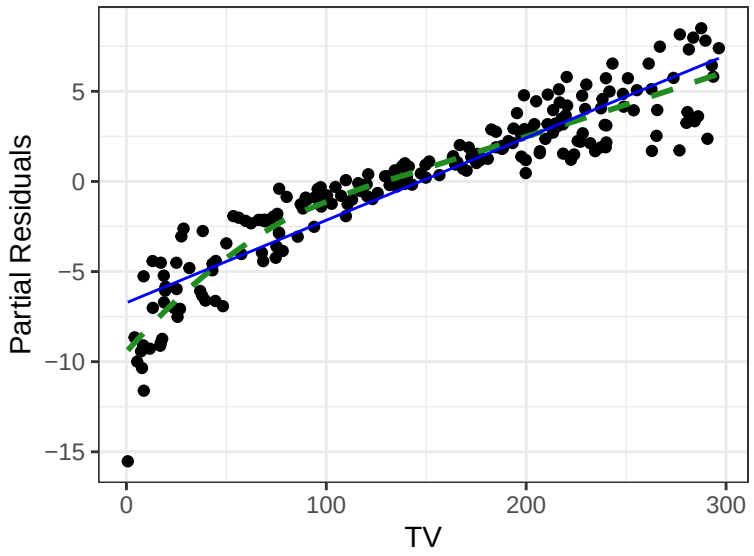
```
##           tv      radio  newspaper
## 1   5.377123  4.3164979  1.5359306
## 2  -6.630676  1.0854125 -1.9529462
## 3  -8.949867  1.2598947 -3.0478695
## 4   1.106166  4.3024979  0.8731767
## 5   1.256228 -2.6385100 -0.3175619
## 6 -11.609543 -0.4451921 -5.3244600
```

Bước 3 để thực hiện vẽ, ta cần tạo một bảng dữ liệu có thông tin cần thiết

```
data_part_resid_tv_md_adv <- tibble(
  tv = data_adv$tv,
  terms_tv = terms_md_adv[, "tv"],
  partial_resid_tv = partial_resid_md_adv[, "tv"]
)
```

Bước 4 ta vẽ biểu đồ như sau:

```
ggplot(data_part_resid_tv_md_adv, mapping = aes(tv, partial_resid_tv)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, linetype = "dashed",
              color = "forestgreen") +
  geom_line(aes(x = tv, y = terms_tv), color = "blue") +
  labs(x = "TV", y = "Partial Residuals") +
  theme_bw()
```



ở đây, ta dùng `geom_line()` để vẽ đường thẳng miêu tả mối quan hệ tuyến tính của thành phần `tv` và `sale`.

Thực hành 18: Vẽ biểu đồ thặng dư từng phần cho hai thành phần `radio` và `newspaper` trong mô hình ở Ví dụ 5, và cho nhận xét.

Thực hành 19: Hãy vẽ biểu đồ thặng dư từng phần cho mô hình hồi quy có sự tương tác giữa `tv` và `radio`, và cho nhận xét.

Thực hành 20: Hãy vẽ biểu đồ thặng dư từng phần cho mô hình hồi quy có thể được xây dựng để dựa đoán tiền lương của các cầu thủ bóng rổ chuyên nghiệp, theo dữ liệu `Hitters` trong thư viện `ISLR2`, và cho nhận xét.

3.3 Kiểm tra tính đồng nhất phương sai

Để kiểm tra tính đồng nhất phương sai của thặng dư trong mô hình hồi quy tuyến tính, ta sử dụng biểu đồ Scale-Location. Biểu đồ này được xác định đơn giản thông qua `ggplot`:

```
ggplot(data = object, mapping = aes(x = .fitted, y = sqrt(abs(.stdresid)))) +
  geom_point(na.rm = TRUE) +
  geom_smooth(method = "loess", se = FALSE, na.rm = TRUE) +
  labs(x = "Fitted Values", y = expression(sqrt("|Standardized residuals|"))) +
  theme_bw()
```

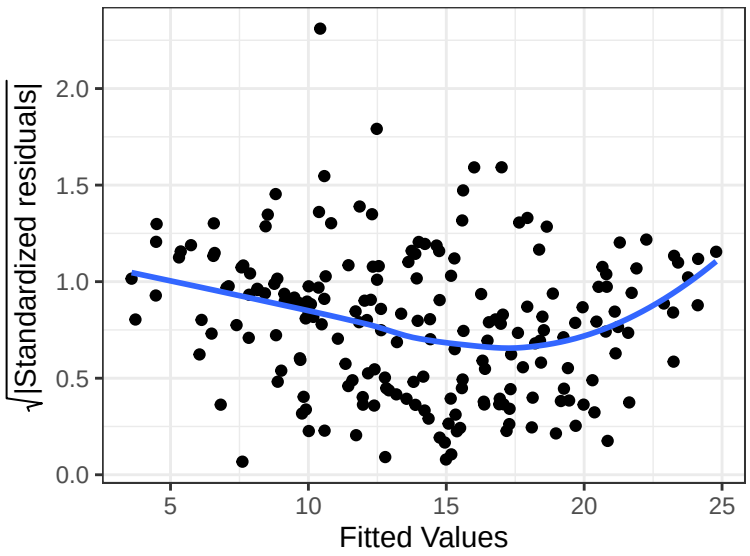
trong đó

- `object` là kết quả của mô hình được ước lượng;
- `.fitted` là $\widehat{Y}_i$;
- `.stdresid` là thặng dư dạng chuẩn hóa của mô hình;
- câu lệnh `geom_smooth(method = "loess", se = FALSE, na.rm = TRUE)` là để vẽ đường cong theo xu hướng của điểm dữ liệu.

Cụ thể, với mô hình hồi quy đang xét, ta có biểu đồ Scale-Location như sau:

```
ggplot(md_adv, aes(.fitted, sqrt(abs(.stdresid)))) +
  geom_point(na.rm = TRUE) +
  geom_smooth(method = "loess", na.rm = TRUE, se = FALSE) +
```

```
  labs(x = "Fitted Values", y = expression(sqrt("|Standardized residuals|"))) +
  theme_bw()
```



Thực hành 21: Thực hiện kiểm tra tính đồng nhất của các mô hình hồi quy tuyến tính đã được đề xuất và ước lượng từ các phần thực hành trên. Và cho nhận xét.

3.4 Kiểm tra điểm ngoại lai trong mô hình

Để kiểm tra điểm ngoại lai có mặt trong mô hình, ta sử dụng biểu đồ Residuals vs Leverage. Biểu đồ này được xác định đơn giản thông qua `ggplot`:

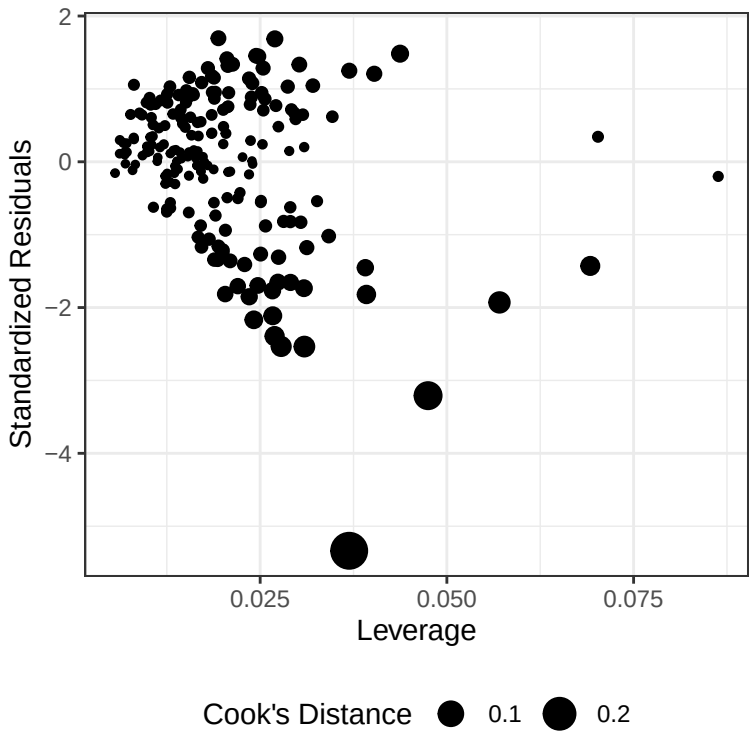
```
ggplot(data = object, aes(x = .hat, y = .stdresid)) +
  geom_point(aes(size = .cooksd)) +
  xlab("Leverage") + ylab("Standardized Residuals") +
  scale_size_continuous("Cook's Distance", range = c(1, 6)) +
  theme_bw() +
  theme(legend.position = "bottom")
```

trong đó

- `object` là kết quả của mô hình được ước lượng;
- `.hat` là giá trị đòn bẩy (leverage);
- `.stdresid` là thặng dư dạng chuẩn hóa của mô hình;
- `.cook` là giá trị khoảng cách Cook (Cook's distance) của các điểm dữ liệu theo mô hình;
- câu lệnh `geom_point(aes(size = .cooksd))` với đối số `size = .cooksd` là để điều chỉnh kích cỡ điểm tương ứng khoảng cách Cook của chúng.

Cụ thể, với mô hình hồi quy đang xét, ta có biểu đồ Residuals vs Leverage như sau:

```
ggplot(md_adv, aes(.hat, .stdresid)) +
  geom_point(aes(size = .cooksd)) +
  xlab("Leverage") + ylab("Standardized Residuals") +
  scale_size_continuous("Cook's Distance", range = c(1, 6)) +
  theme_bw() +
  theme(legend.position = "bottom")
```



Bên cạnh đó, ta có thể tính được các kết quả của

- thặng dư chuẩn hóa bởi `rstandard()`;
- giá trị đòn bẩy của các điểm dữ liệu bởi `hatvalues()`;
- khoảng cách Cook bởi `cook()`.

Cụ thể, ta có

```
std_resid_md_adv <- rstandard(md_adv)
hat_values_md_adv <- hatvalues(md_adv)
cooks_D_md_adv <- cooks.distance(md_adv)
```

Ghép các kết quả này thành một bảng dữ liệu (bằng hàm `tibble()`), ta có thể sắp xếp và tìm ra các điểm là ngoại lai của mô hình.

```
data_cooks_md_adv <- tibble(id_point = 1:nrow(data_adv),
                            rstand = std_resid_md_adv, hats = hat_values_md_adv,
                            cooks = cooks_D_md_adv, sales = data_adv$sales)
data_cooks_md_adv |> arrange(desc(cooks))
```

```
## # A tibble: 200 x 5
##   id_point   rstand      hats     cooks sales
##      <int>    <dbl>     <dbl>     <dbl> <dbl>
## 1      131 -5.33684 0.0369188 0.272956    1.6
## 2        6 -3.20870 0.0474811 0.128306    7.2
## 3       76 -1.92965 0.0570429 0.0563128   8.7
## 4       36 -2.53538 0.0309202 0.0512754  12.8
## 5      179 -2.53384 0.0278142 0.0459215  11.8
## # i 195 more rows
```

Thực hành 22: Thực hiện kiểm tra điểm ngoại lai của các mô hình hồi quy tuyến tính đã được đề xuất và

ước lượng từ các phần thực hành trên. Và cho nhận xét. Nếu có điểm ngoại lai xuất hiện trong biểu đồ, hãy trích xuất các điểm này từ dữ liệu gốc.

4 Mở rộng mô hình hồi quy

Trong phần này, ta cũng tìm hiểu cách sử dụng của các hàm để mở rộng mô hình hồi quy tuyến tính.

4.1 Hồi quy đa thức

Trong `R`, để ước lượng mô hình hồi quy tuyến tính với thành phần đa thức của một (hoặc nhiều) biến hồi quy, ta sử dụng hàm `lm()` kết hợp với `poly()`, cụ thể như sau:

```
lm(formula = y ~ poly(x1, degree) + x2 + ..., data = ..., ...)
```

trong đó,

- `poly(x1, degree)` tương ứng với thành phần đa thức cho biến `x1`;
- đối số `degree` là bậc của đa thức.

Ví dụ 6: ta mở rộng thành phần `tv` thành đa thức bậc 2:

```
md_adv_poly <- lm(sales ~ poly(tv, 2) + radio + newspaper, data = data_adv)
```

Kết quả tổng hợp của mô hình đa thức

```
summary(md_adv_poly)
```

```
##
## Call:
## lm(formula = sales ~ poly(tv, 2) + radio + newspaper, data = data_adv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3583 -0.8701 -0.0484  0.9562  3.5604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.515e+00  2.244e-01  42.401  < 2e-16 ***
## poly(tv, 2)1  5.535e+01  1.525e+00  36.301  < 2e-16 ***
## poly(tv, 2)2 -1.035e+01  1.531e+00  -6.757 1.59e-10 ***
## radio         1.926e-01  7.794e-03  24.706  < 2e-16 ***
## newspaper     8.906e-04  5.306e-03   0.168    0.867
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.521 on 195 degrees of freedom
## Multiple R-squared:  0.9167, Adjusted R-squared:  0.915
## F-statistic: 536.6 on 4 and 195 DF,  p-value: < 2.2e-16
```

Sau khi ước lượng mô hình, các thao tác về tiên đoán, chuẩn đoán mô hình hay hồi quy từng bước được thực hiện tương tự như trường hợp của mô hình cơ bản.

Thực hành 23: Sử dụng hồi quy đa thức, kết hợp với biểu đồ thặng dư từng phần đối với biến `tv`, hãy nâng bậc của đa thức đối với `tv`, cho tới khi xu hướng phi tuyến của thặng dư được mô tả gần đúng bởi thành phần đa thức.

Thực hành 24: Áp dụng mô hình hồi quy đa thức cho mô hình có sự tương tác giữa `tv` và `radio`. Hiệu chỉnh bậc đa thức, cho tới khi biểu đồ Residuals vs Fitted không cho thấy xu hướng phi tuyến.

4.2 Splines

Để áp dụng được hồi quy splines, ta cần thư viện `splines`. Để ước lượng mô hình hồi quy tuyến tính với thành phần B-splines của một (hoặc nhiều) biến hồi quy, ta sử dụng hàm `lm()` kết hợp với `bs()`, cụ thể như sau:

```
lm(formula = y ~ bs(x1, knots, degree) + x2 + ..., data = ..., ...)
```

trong đó,

- `bs(x1, knots, degree)` tương ứng với thành phần B-splines cho biến `x1`;
- đối số `knots` là các điểm nút của B-splines;
- đối số `degree` là bậc của B-splines.

Ví dụ 7: ta mở rộng thành phần `tv` thành B-splines bậc 3, trước tiên ta xác định các điểm nút. Ở đây, ta chọn các điểm nút là các điểm phân vị mẫu của `tv`:

```
knots_tv <- quantile(data_adv$tv, probs = c(0.25, 0.5, 0.75))
knots_tv
```

```
##     25%     50%     75%
##  74.375 149.750 218.825
```

Sử dụng những điểm nút này, ta ước lượng mô hình với thành phần B-splines bậc 3 như sau:

```
md_adv_spline <- lm(sales ~ bs(tv, knots = knots_tv, degree = 3) +
                      radio + newspaper, data = data_adv)
```

Kết quả tổng hợp của mô hình

```
summary(md_adv_spline)
```

```
##
## Call:
## lm(formula = sales ~ bs(tv, knots = knots_tv, degree = 3) + radio +
##     newspaper, data = data_adv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5456 -0.7570  0.0688  0.7463  4.0131
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -1.6068558  0.6640629  -2.420   0.0165 *
## bs(tv, knots = knots_tv, degree = 3)1  6.7637646  1.2283180   5.507 1.17e-07 ***
## bs(tv, knots = knots_tv, degree = 3)2  8.7191364  0.7270780  11.992  < 2e-16 ***
## bs(tv, knots = knots_tv, degree = 3)3 11.8778918  0.9509190  12.491  < 2e-16 ***
## bs(tv, knots = knots_tv, degree = 3)4 14.3786157  0.8262404  17.402  < 2e-16 ***
## bs(tv, knots = knots_tv, degree = 3)5 15.7610667  1.0443366  15.092  < 2e-16 ***
## bs(tv, knots = knots_tv, degree = 3)6 17.2927265  0.8895052  19.441  < 2e-16 ***
## radio                                 0.1959157  0.0072108  27.170  < 2e-16 ***
## newspaper                            -0.0006698  0.0049062  -0.137   0.8916
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.399 on 191 degrees of freedom
## Multiple R-squared:  0.931,  Adjusted R-squared:  0.9281
## F-statistic: 322.2 on 8 and 191 DF,  p-value: < 2.2e-16
```

Sau khi ước lượng mô hình, các thao tác về tiên đoán, chuẩn đoán mô hình hay hồi quy từng bước được thực hiện tương tự như trường hợp của mô hình cơ bản.

Thực hành 25: Vẽ biểu đồ thặng dư từng phần đối với biến `tv` trong mô hình B-splines bậc 3, nhận xét.

Thực hành 26: Áp dụng mô hình hồi B-splines bậc 3 cho biến là nguồn tạo ra sự phi tuyến trong mô hình có sự tương tác giữa `tv` và `radio`. Vẽ biểu đồ thặng dư và thặng dư từng phần và nhận xét.

4.3 GAM

Để áp dụng được hồi quy GAM, ta cần thư viện `mgcv`. Để ước lượng mô hình hồi quy GAM của một (hoặc nhiều) biến hồi quy, ta sử dụng hàm `gam()` như sau:

```
gam(formula = y ~ s(x1) + x2 + ..., data = ..., ...)
```

trong đó,

- `s(x1)` tương ứng với một hàm trơn cho `x1`;
- `x2` có nghĩa là ta đang áp dụng quan hệ tuyến tính giữa `y` và `x2`;
- thành phần tương tác giữa hai biến có thể được thêm theo dạng cơ bản, tức là `x1:x2` hoặc là dạng một hàm trơn của hai biến `s(x1,x2)`.

Ví dụ 8: ta mở rộng thành phần `tv` thành một hàm trơn:

```
md_adv_gam <- gam(sales ~ s(tv) + radio + newspaper, data = data_adv)
```

Kết quả tổng hợp của mô hình

```
summary(md_adv_gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sales ~ s(tv) + radio + newspaper
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.4754609  0.2096422  45.198   <2e-16 ***
## radio        0.1958493  0.0073130  26.781   <2e-16 ***
## newspaper   -0.0003011  0.0049641  -0.061    0.952
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(tv) 6.347  7.499 214.2  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.926   Deviance explained = 92.9%
```

```
## GCV = 2.1041  Scale est. = 2.0057    n = 200
```

Sau khi ước lượng mô hình, các thao tác về tiên đoán, chuẩn đoán mô hình hay hồi quy từng bước được thực hiện tương tự như trường hợp của mô hình cơ bản.

Để xác định thặng dư từng phần trong GAM, ta cần ước tính các thành phần trong công thức hồi quy của mô hình:

```
terms_md_adv_gam <- predict(md_adv_gam, type = "terms")
head(terms_md_adv_gam)
```

```
##      radio   newspaper      s(tv)
## 1 7.403103 -0.02083311  3.4969655
## 2 7.696877 -0.01357765 -4.2548919
## 3 8.989482 -0.02086321 -7.4552700
## 4 8.088575 -0.01761180  0.6890684
## 5 2.115172 -0.01758170  1.8342227
## 6 9.577030 -0.02257924 -8.7365037
```

sau đó, tính thặng dư của mô hình và áp dụng công thức tính thặng dư từng phần (xem trong slide lý thuyết):

```
partial_resid_md_adv_gam <- residuals(md_adv_gam, type = "working") +
  terms_md_adv_gam
head(partial_resid_md_adv_gam)
```

```
##      radio  newspaper       s(tv)
## 1 9.148407  1.7244710   5.2422695
## 2 5.193009 -2.5174456  -6.7587598
## 3 7.300672 -1.7096727  -9.1440795
## 4 8.353082  0.2468955   0.9535758
## 5 1.607898 -0.5248558   1.3269486
## 6 6.483622 -3.1159869 -11.8299113
```

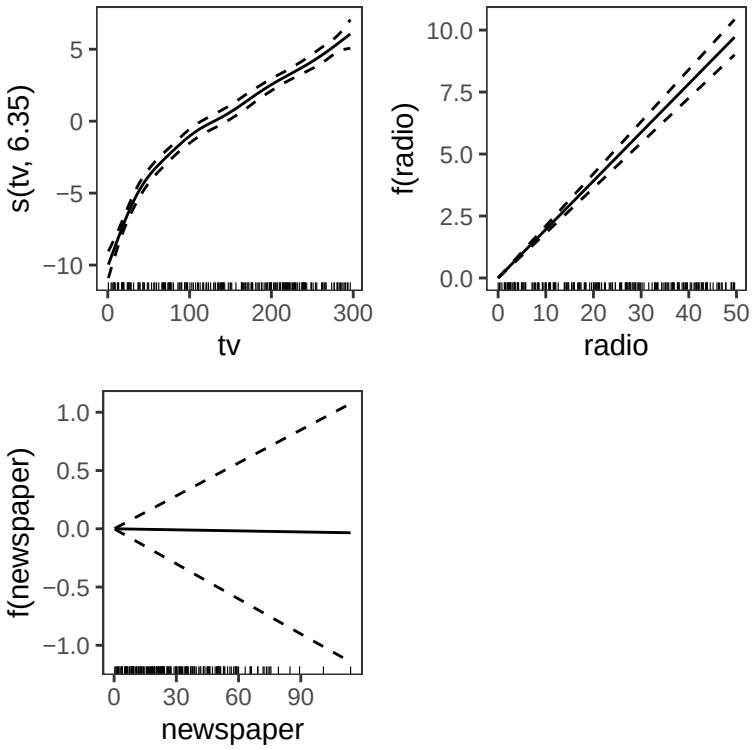
Từng cột tương ứng với kết quả của từng thành phần trong mô hình hồi quy, với tên cột tương ứng tên biến hoặc thành phần.

Thực hành 27: Vẽ biểu đồ thặng dư từng phần đối với biến `tv` trong mô hình GAM, nhận xét.

Thực hành 28: Áp dụng mô hình GAM cho biến là nguồn tạo ra sự phi tuyến trong mô hình có sự tương tác giữa `tv` và `radio`. Vẽ biểu đồ thặng dư và thặng dư từng phần và nhận xét.

Ngoài ra, với hàm `getViz()` trong thư viện `mgcViz` ta có thể minh họa dạng hàm trơn được ước lượng trong mô hình:

```
viz_md_adv <- getViz(md_adv_gam)
print(plot(viz_md_adv, allTerms = T), pages = 1)
```


s(tv, 6.35)
5
0
−5
−10
0
100
200
300
tv
f(radio)
10.0
7.5
5.0
2.5
0.0
0
10
20
30
40
50
radio
f(newspaper)
1.0
0.5
0.0
−0.5
−1.0
0
30
60
90
newspaper