Chapter 3

Wenduo Wang

July 20, 2016

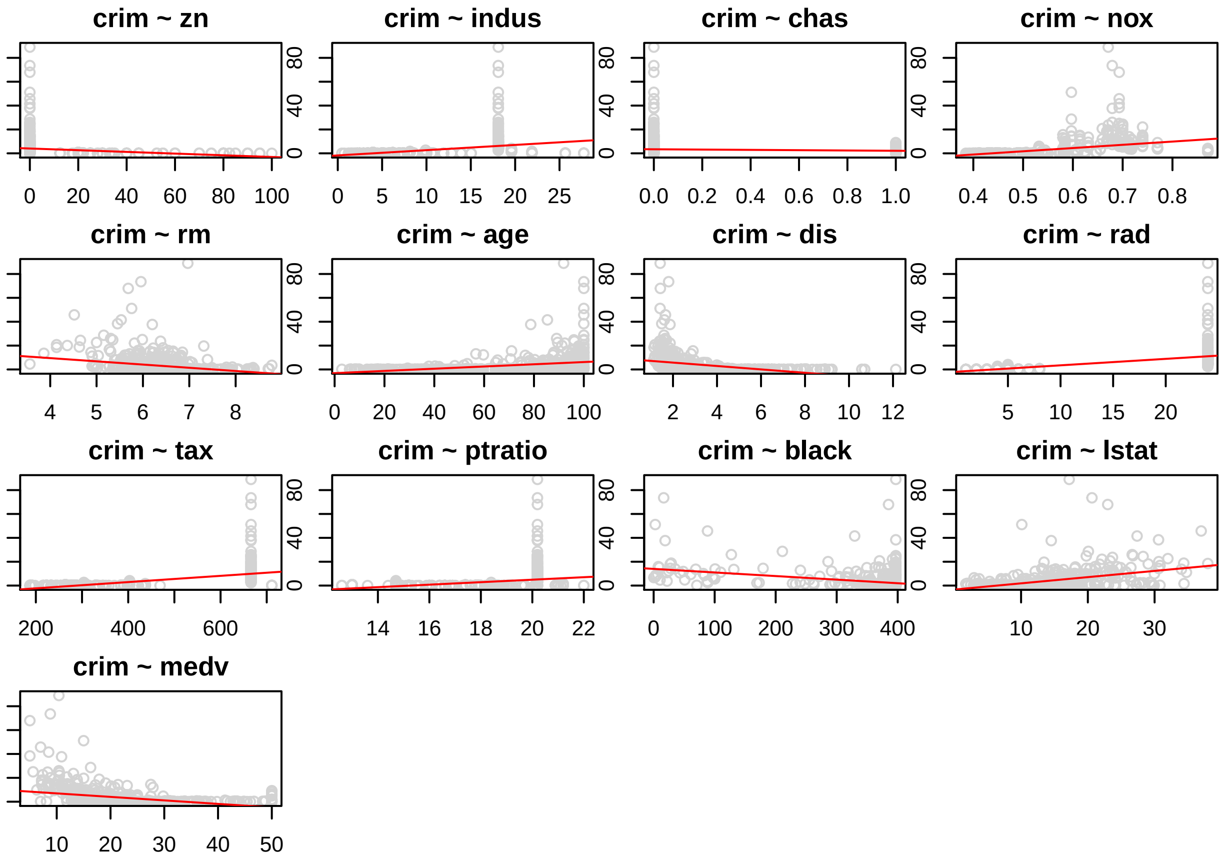
Problem 15

(a) Let's create a simple linear regression model for each variable against `crim`, and plot the fitted line and actual data points.

```
library(MASS)
library(dplyr)
```

```
>>>
>>> Attaching package: 'dplyr'
>>> The following object is masked from 'package:MASS':
>>>
>>>     select
>>> The following objects are masked from 'package:stats':
>>>
>>>     filter, lag
>>> The following objects are masked from 'package:base':
>>>
>>>     intersect, setdiff, setequal, union
```

```
summary(Boston)
row_no <- nrow(Boston)
training_rows <- sample(1:row_no, round(row_no*0.8))
training_set <- Boston[training_rows,]
test_set <- setdiff(Boston, training_set)
confint_df <- data.frame()
par(mfrow=c(4, 4), mar=c(2, 1, 2, 1))
for (i in c(1:ncol(Boston))){
    if (colnames(Boston)[i]=="crim"){
        next
    }
    formula <- as.formula(paste("crim~", colnames(Boston)[i]))
    lm_model <- lm(formula, data=training_set)
    plot(y=Boston$crim, x=Boston[,i], main=paste("crim ~", colnames(Boston)[i]), pch=1, col="lightgray")
    abline(lm_model$coef[1], lm_model$coef[2], col="red")
    confint_df <- rbind(confint_df, confint(lm_model)[2,])
}
print(dim(confint_df))
colnames(confint_df) <- c("2.5%", "97.5%")
row.names(confint_df) <- colnames(Boston)[colnames(Boston)!="crim"]
```




```
>>>       crim                zn              indus            chas
>>>  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
>>>  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
>>>  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
>>>  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
>>>  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
>>>  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
>>>       nox               rm             age              dis
>>>  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
>>>  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
>>>  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
>>>  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
>>>  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
>>>  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
>>>       rad              tax           ptratio          black
>>>  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
>>>  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
>>>  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
>>>  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
>>>  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
>>>  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
>>>      lstat            medv
>>>  Min.   : 1.73   Min.   : 5.00
>>>  1st Qu.: 6.95   1st Qu.:17.02
>>>  Median :11.36   Median :21.20
>>>  Mean   :12.65   Mean   :22.53
>>>  3rd Qu.:16.95   3rd Qu.:25.00
```

```
>>>  Max.   :37.97   Max.   :50.00
>>> [1] 13  2
```

The linear model does not fit the data very well according to the plots. To further assess the correlation, let's have a closer look at the coefficients of these models

```r
print(confint_df)
```

```
>>>                2.5%       97.5%
>>> zn      -0.09935118 -0.04268520
>>> indus    0.35808317  0.51961541
>>> chas    -3.82768332  1.29807214
>>> nox     22.51577304 31.79801225
>>> rm      -3.55908946 -1.84409106
>>> age      0.07281828  0.11311928
>>> dis     -1.70160646 -1.14786283
>>> rad      0.49330998  0.58986045
>>> tax      0.02345440  0.02887982
>>> ptratio  0.76702904  1.31840736
>>> black   -0.03613965 -0.02384384
>>> lstat    0.45027390  0.59180300
>>> medv    -0.35537234 -0.23229956
```

The above code lists the 95% confidence intervals of coefficients for all the predictors with respect to each linear model. As seen from the result, the confident interval of the coefficient of `chas` contains zero, which means this predictor is probably not correlated with `crim`. Meanwhile, other predictors mostly have very small coefficients close to 0, with the exception of `nox`. So up to now, it appears `nox` is most likely to be a true predictor of `crim`.

(b, c) Now let's create a linear model of `crim` on all predictors, and see their coefficients' confident intervals.

```r
lm_model_multiple <- lm(crim~., data=training_set)
print(confint(lm_model_multiple))
```

```
>>>                     2.5 %        97.5 %
>>> (Intercept)   0.275792710 20.3372816122
>>> zn            0.005906254  0.0602465169
>>> indus        -0.185803063  0.0406832888
>>> chas         -2.181862616  1.1271352400
>>> nox         -12.797970913  1.3004008691
>>> rm           -1.181755678  0.5438208326
>>> age          -0.032856938  0.0153680742
>>> dis          -1.123715298 -0.3230844705
>>> rad           0.363724585  0.5982704137
>>> tax          -0.008207642  0.0053972468
>>> ptratio      -0.417741157  0.1023245419
>>> black        -0.009084191  0.0009468215
>>> lstat         0.142689480  0.3494199391
>>> medv         -0.118350664  0.0423924819
```

When all the predictors are simultaneously fitted against `crim`, the result turns out very different from the previous single linear regression models. First, the previously strong predictor `nox` is now largely irrelevant. Actually in this case only `zn`, `dis` and `rad` show a significant correlation with `crim`, which is equivalent to rejecting the null hypothesis that H_0: $\beta_j=0$.

(d) To inspect if there is a correlation in the form of

$$crim = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

we create a linear regression model for each predictor against `crim`, in the format of `lm(crim ~ x + x^2 + x^3, data=training_set)`, and then inspect the model's coefficient confidence interval.

```
for (i in c(1:ncol(Boston))){
    if (colnames(Boston)[i]=="crim"){
        next
    }
    variable <- colnames(Boston)[i]
    formula <- as.formula(paste("crim ~ ", variable, " + I(", variable, "^2) + I(", variable, "^3)", sep
    cat("For predictor:", colnames(Boston)[i], "\n")
    pn_model_3 <- lm(formula, data=training_set)
    print(confint(pn_model_3))
    cat("\n")
}
```

```
>>> For predictor: zn
>>>                     2.5 %        97.5 %
>>> (Intercept)  3.658861e+00  5.051352e+00
>>> zn          -4.914741e-01 -1.112041e-01
>>> I(zn^2)     -8.441882e-04  1.290484e-02
>>> I(zn^3)     -9.308368e-05  2.095429e-05
>>>
>>> For predictor: indus
>>>                    2.5 %       97.5 %
>>> (Intercept)  0.631334004  5.778530428
>>> indus       -2.454347666 -0.920315743
>>> I(indus^2)   0.155025514  0.277047179
>>> I(indus^3)  -0.007414645 -0.004517687
>>>
>>> For predictor: chas
>>>                 2.5 %   97.5 %
>>> (Intercept)  2.746257 4.044981
>>> chas        -3.827683 1.298072
>>> I(chas^2)          NA       NA
>>> I(chas^3)          NA       NA
>>>
>>> For predictor: nox
>>>                  2.5 %    97.5 %
>>> (Intercept)   160.8801  261.0944
>>> nox         -1405.4935 -899.6081
>>> I(nox^2)     1602.2452 2429.8508
>>> I(nox^3)    -1331.4259 -892.1624
>>>
>>> For predictor: rm
>>>                   2.5 %      97.5 %
>>> (Intercept)  219.783988 479.3125698
>>> rm          -209.263732 -86.4206344
>>> I(rm^2)       11.328428  30.4951872
>>> I(rm^3)       -1.475121  -0.4907945
>>>
>>> For predictor: age
>>>                     2.5 %       97.5 %
>>> (Intercept) -5.796322e+00 2.561622e+00
>>> age         -1.057828e-01 4.693557e-01
>>> I(age^2)    -1.055695e-02 7.467448e-04
```

```
>>> I(age^3)     8.131208e-06 7.384325e-05
>>>
>>> For predictor: dis
>>>                   2.5 %     97.5 %
>>> (Intercept)  21.1065970 28.5750522
>>> dis         -15.2181882 -9.9027494
>>> I(dis^2)      1.4101874  2.4722936
>>> I(dis^3)     -0.1235113 -0.0605432
>>>
>>> For predictor: rad
>>>                    2.5 %      97.5 %
>>> (Intercept) -3.402618045 2.171492473
>>> rad         -0.904057033 1.945109126
>>> I(rad^2)    -0.277490344 0.128332516
>>> I(rad^3)    -0.003174294 0.009294897
>>>
>>> For predictor: tax
>>>                     2.5 %        97.5 %
>>> (Intercept)  4.962689e+00  3.887594e+01
>>> tax         -3.153024e-01 -4.145752e-02
>>> I(tax^2)     8.760718e-05  7.792764e-04
>>> I(tax^3)    -5.558744e-07 -1.872422e-08
>>>
>>> For predictor: ptratio
>>>                     2.5 %       97.5 %
>>> (Intercept)    62.4440773 585.82089808
>>> ptratio      -100.9841111  -8.49334118
>>> I(ptratio^2)    0.3043651   5.70013095
>>> I(ptratio^3)   -0.1050957  -0.00112827
>>>
>>> For predictor: black
>>>                     2.5 %       97.5 %
>>> (Intercept)  7.433054e+00 1.465026e+01
>>> black       -3.446801e-02 1.453479e-01
>>> I(black^2)  -8.729851e-04 7.868724e-05
>>> I(black^3)  -1.886857e-07 1.198319e-06
>>>
>>> For predictor: lstat
>>>                    2.5 %      97.5 %
>>> (Intercept) -2.368286644 3.524044980
>>> lstat       -0.851193726 0.485856214
>>> I(lstat^2)  -0.015414196 0.070686286
>>> I(lstat^3)  -0.001003005 0.000605381
>>>
>>> For predictor: medv
>>>                    2.5 %        97.5 %
>>> (Intercept) 38.027613110 48.4434072905
>>> medv        -4.702793923 -3.3587273182
>>> I(medv^2)    0.093422906  0.1465412678
>>> I(medv^3)   -0.001433273 -0.0008062374
```

As seen from the output confidence intervals, it is first noticed that `chas^2` and `chas^3` do not have coefficients. That is due to that `chas` is a binary variable, whose square or cube is essentially itself, so in this case `chas`, `chas^2` and `chas^3` are linearly related, and therefore the latter two polynomial terms are not fitted in the model. `chas` put aside, several polynomial terms exhibit correlation with `crim`, whose 95% coefficient confidence intervals exclude zero. For example, `nox^2` and `nox^3`.