MLR: Regression Diagnostics

Hannah Waddel

3/16/2021

Regression Diagnostics in R

After we fit a multiple linear regression model, we must evaluate the assumptions we have made to fit the model. We will evaluate linearity, outliers, and multicollinearity.

In this exercise we will continue the previous exercise with the birthweight data. Begin by reading in the birthweight data and fit the multiple linear regression model.

```r
## Load the sas7bdat package to read in a SAS dataset
library(sas7bdat)

## Set working directory
setwd("~/OneDrive - Emory University/Documents/Work/Bios 591P 2022/R Materials/3 MLR/Data")
## Read in the data
bwt <- read.sas7bdat("birthweight.sas7bdat")

## Fit MLR model
bwt.lm <- lm(BWT ~ AGE + WT + SMOKE + HT,data=bwt)
```

We are going to load two packages which contain useful functions for regression diagnostics: car and MASS. These packages are automatically installed when you install R, but you must load them with the *library()* function before using them.

```r
## Load packages for regression diagnostics
library(car)
```

```
## Loading required package: carData
```

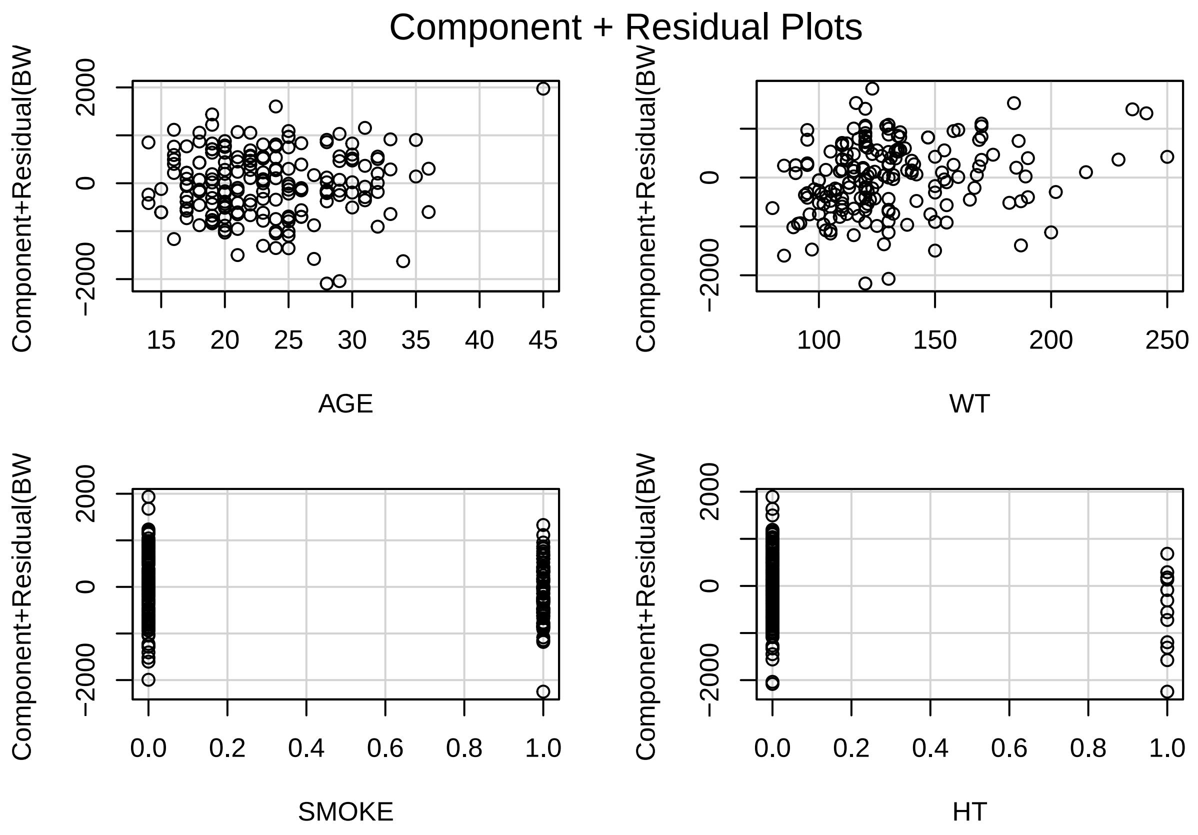
```r
library(MASS)
```

Linearity

Partial Plots

We use the *crPlots()* function, from the car package, in order to fit partial plots. We give two additional arguments, line=FALSE and smooth=FALSE, to tell *crPlots()* not to fit lines over the residuals or add smoothing lines (see what happens when you remove line=FALSE and smooth=FALSE)

```r
## Partial plots: crPlots()
crPlots(bwt.lm,line=FALSE,smooth=FALSE)
```



Influential Points and Outliers

Leverage

We use the function *hatvalues()* in order to calculate the leverage values in R. It is called "hatvalues" because the leverage values are calculated from the hat matrix (used to fit the linear model).

The *hatvalues()* function takes our linear model object as its argument. Calculate and save the leverage values.

```r
## Leverage: hatvalues()
lev.vals <- hatvalues(bwt.lm)
```

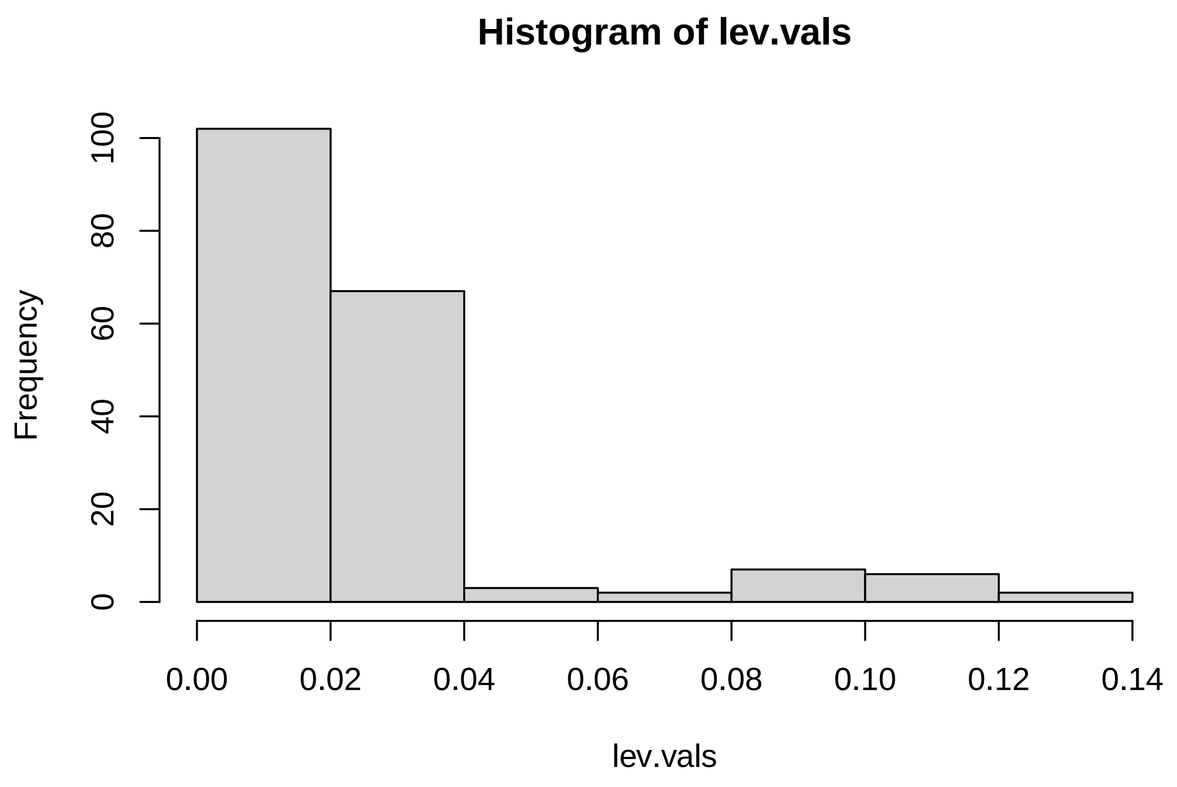
The threshold for concerning leverage values is $\frac{2(k+1)}{n}$, where we have k explanatory variables. In this case, we have 4 explanatory variables. Calculate and save the leverage threshold.

We calculate n using the *length()* function, which will tell us how many leverage values we have, and thus how many subjects we have in our model.

```r
lev.threshold <- 2*(4+1)/length(lev.vals)
```

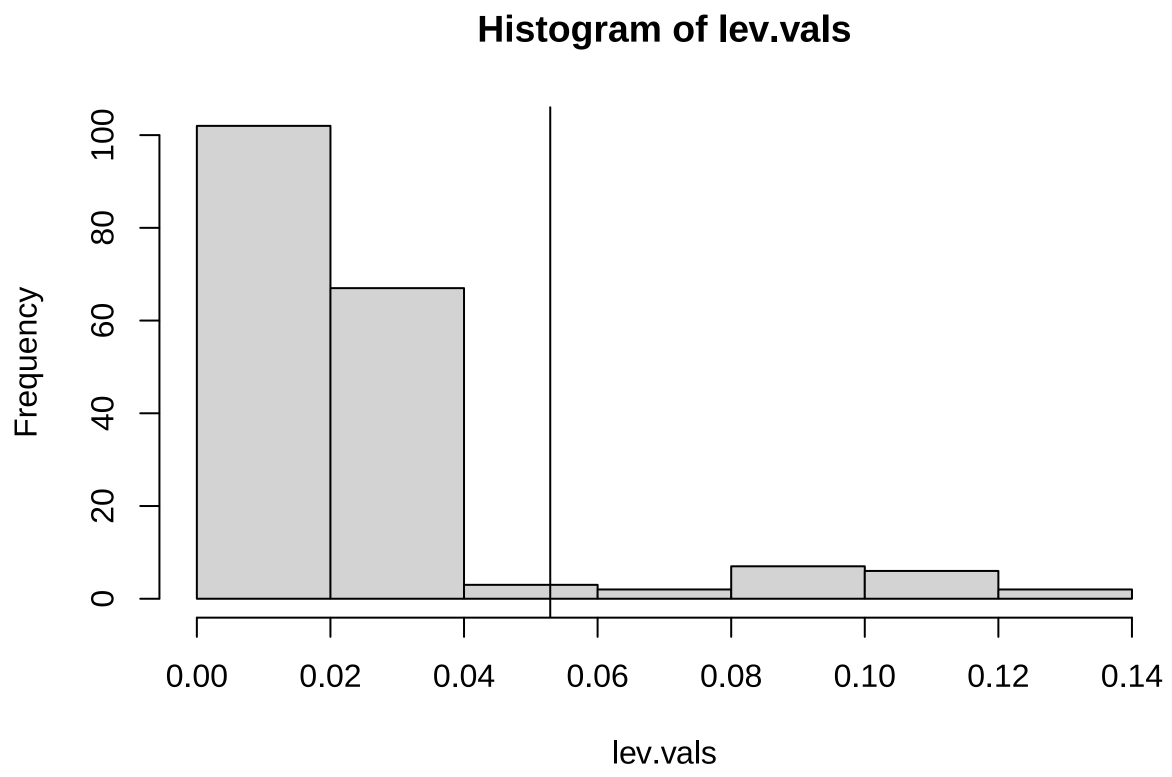
We can see a histogram of the leverage values with the *hist()* function.

```
hist(lev.vals)
```



To more easily see the values of concern, we add a vertical line to our plot with the *abline()* function. This is a very flexible function which can add lines to a plot. We'll see it again, but for now we add a vertical line with the "v=" argument in abline, and we draw the line at the leverage threshold.

```
hist(lev.vals)
abline(v=lev.threshold)
```



To see which values are above the threshold, we give R a logical (true/false) statement with the $>$ operator.

```
lev.vals > lev.threshold
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##    66    67    68    69    70    71    72    73    74    75    76    77    78
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##    79    80    81    82    83    84    85    86    87    88    89    90    91
## FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   144   145   146   147   148   149   150   151   152   153   154   155   156
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##   157   158   159   160   161   162   163   164   165   166   167   168   169
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##   170   171   172   173   174   175   176   177   178   179   180   181   182
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   183   184   185   186   187   188   189
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

This just gives us a list of true/false values telling us which observations have a leverage value higher than the threshold.

The list of true/false values is actually stored as 1s and 0s, so if we take the sum of the true and false values, we will see how many leverage values are higher than the threshold.

```
sum(lev.vals > lev.threshold)
```

```
## [1] 17
```

There are 17 observations with a leverage value higher than the threshold.

In order to see which leverage values are high, we use the *which()* function to tell us the location where "lev.vals > lev.threshold" is true.

```
which(lev.vals > lev.threshold)
```

```
##   3   4   9  10  51  58  59  72  82  98 110 127 135 152 161 165 189
##   3   4   9  10  51  58  59  72  82  98 110 127 135 152 161 165 189
```

Now, if we want to see the data for the observations with a high leverage value, we use the brackets *[,]* to select the specific rows of the birthweight dataset.

```
bwt[which(lev.vals > lev.threshold),]
```

```
##      ID  BWT AGE  WT SMOKE HT RACE
## 3    11 1135  34 187     0  1    1
## 4    13 1330  25 105     0  1    2
## 9    19 1729  24 132     0  1    1
## 10   20 1790  21 165     1  1    1
## 51   75 2442  26 154     0  1    2
## 58   83 2495  17 142     0  1    2
## 59   84 2495  21 130     1  1    1
## 72   98 2750  22  95     0  1    2
## 82  108 2836  36 202     0  0    3
## 98  126 3005  31 215     1  0    1
## 110 138 3100  22 120     0  1    1
## 127 159 3303  28 250     1  0    2
## 135 168 3402  18 229     0  0    1
## 152 187 3629  19 235     1  1    1
## 161 197 3756  19 184     1  1    2
## 165 202 3790  25 241     0  1    3
## 189 226 5001  45 123     0  0    1
```

Cook's Distance

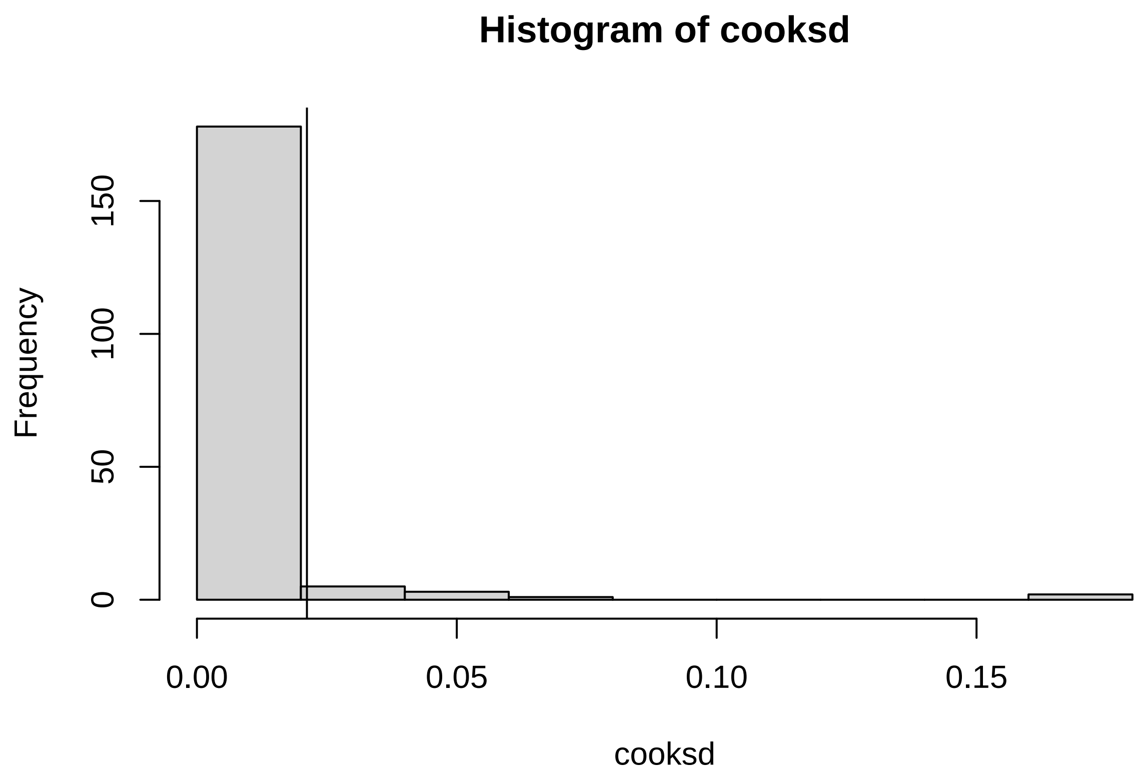
We follow a similar process with Cook's distance, but the threshold of concern is now $\frac{4}{n}$. We calculate Cook's distance with the *cooks.distance()* function, which takes our linear model object as its argument.

```
## Cook's distance: cooks.distance()
```

```r
cooksd <- cooks.distance(bwt.lm)

# Threshold: 4/n
cooks.threshold <- 4/length(cooksd)

hist(cooksd)
abline(v=cooks.threshold)
```



```r
cooksd > cooks.threshold
```

```r
which(cooksd > cooks.threshold)
```

```
##   1   2   3   4   9  17 110 152 161 165 189
##   1   2   3   4   9  17 110 152 161 165 189
```

```r
bwt[which(cooksd > cooks.threshold),]
```

```
##      ID  BWT AGE  WT SMOKE HT RACE
## 1     4  709  28 120     1  0    2
## 2    10 1021  29 130     0  0    2
## 3    11 1135  34 187     0  1    1
## 4    13 1330  25 105     0  1    2
## 9    19 1729  24 132     0  1    1
## 17   28 1928  21 200     0  0    2
## 110 138 3100  22 120     0  1    1
## 152 187 3629  19 235     1  1    1
## 161 197 3756  19 184     1  1    2
## 165 202 3790  25 241     0  1    3
## 189 226 5001  45 123     0  0     1
```

```
sum(cooksd > cooks.threshold)
```

```
## [1] 11
```

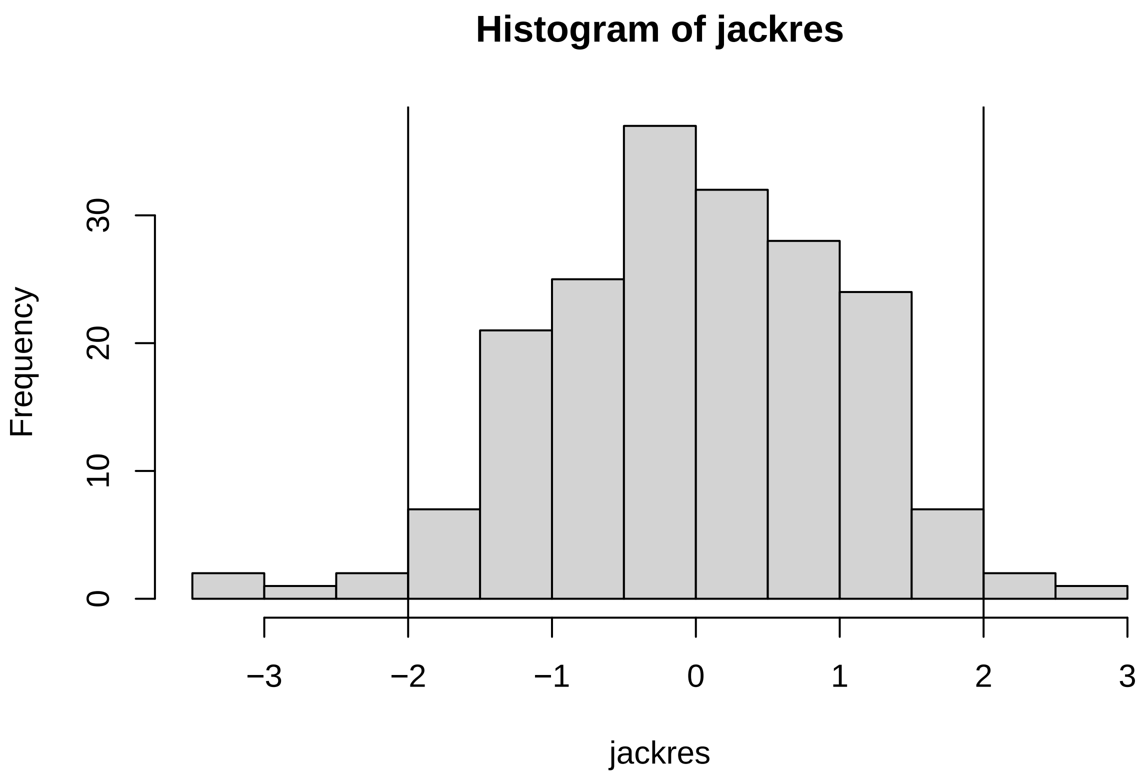
Jackknife Residuals

The jackknife residuals are calculated with the *studres()* function from the MASS package, because another name for the jackknife residuals is “studentized residuals”.

```
## Jackknife Residuals: studres()
jackres <- studres(bwt.lm)
```

The threshold for the jackknife residuals is 2, but remember that the residuals can be negative–so we are looking for jackknife residuals with an absolute value greater than 2.

```
# Threshold: absolute value > 2
hist(jackres)
abline(v=2)
abline(v=-2)
```



```
abs(jackres) > 2
```

```
which(abs(jackres) > 2)
```

```
##   1   2   3   6  17 187 188 189
##   1   2   3   6  17 187 188 189
```

```
bwt[which(abs(jackres) > 2),]
```

```
##      ID  BWT AGE  WT SMOKE HT RACE
## 1     4  709  28 120     1  0    2
## 2    10 1021  29 130     0  0    2
## 3    11 1135  34 187     0  1    1
## 6    16 1588  27 150     0  0    2
## 17   28 1928  21 200     0  0    2
## 187 224 4238  19 120     1  0    1
## 188 225 4593  24 116     0  0    1
## 189 226 5001  45 123     0  0    1
```

Multicollinearity

Correlation

We test the correlation between explanatory variables with the *cor.test()* function. Here, we calculate the correlation between age and weight, which are our two continuous explanatory variables.

```
cor.test(bwt$AGE,bwt$WT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  bwt$AGE and bwt$WT
## t = 2.5034, df = 187, p-value = 0.01316
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.03832798 0.31471467
## sample estimates:
##       cor
## 0.1800732
```

Variance Inflation Factors

We can also evaluate multicollinearity by calculating the Variance Inflation Factors. The *vif()* function from the car package will do this for us.

```
vif(bwt.lm)
```

```
##      AGE       WT    SMOKE       HT
## 1.040064 1.100864 1.006497 1.063364
```

Recall that the threshold for concerning VIFs is around 10, so we do not see any VIFs of concern.