MLR Initial Report

Hannah Waddel

Initial Report: Relationship between Birthweight and Several Variables

In this exercise we are going to perform a multiple linear regression to assess whether age, weight, smoking status, or hypertension status were significantly associated with birthweight.

We will begin by reading in the data. I have provided the data in a SAS dataset, "birthweight.sas7bdat".

Importing the Data

The data are stored as an excel file, "Spring 2023 birthweight.xlsx". Before importing the data, make sure that your working directory is set to the folder where you have saved the dataset.

```
setwd("~/Library/CloudStorage/OneDrive-EmoryUniversity/Documents/Work/BIOS 591P 2023/3 MLR/Data/")
```

We need to get the data into a format which R can use. R does not have a built-in capability to read in an excel dataset, but we can install packages which expand R's capabilities.

Packages are easy to install and make R an incredibly flexible programming language. People create R packages to implement new statistical methods or expand R's existing methods.

If a package is of publication quality and makes a non-trivial contribution to R's capabilities, it is stored on the Comprehensive R Archive Network (CRAN). If a package is on CRAN, you can install it using the *install.packages()* function.

The package we are installing today is called "readxl". It allows us to read in Excel datasets, which are stored as ".xlsx" files on your computer. There are multiple packages to do this.

To install the package, we are using the *install.packages()* function, which requires the name of the package we want as a string (text enclosed by " ").

Let's install the package "reacxl" now. Make sure that the package name is in quotation marks.

```
install.packages("readxl")
```

```
## Installing package into '/Users/hbwadde/Library/R/x86_64/4.1/library'
## (as 'lib' is unspecified)

##
## The downloaded binary packages are in
##  /var/folders/4d/hb2p2stj3090dvt4p1f88qrr0000gq/T//RtmpovdkoH/downloaded_packages
```

R will tell you where on your computer it is putting the files for the package (default settings should be fine).

The readxl package gives us an important function *read_xlsx()*. This works like the *read.csv()* function we have used, but for SAS datasets.

However, if we try to use *read_xlsx()* right now, we get an error.

```
bwt <- read_xlsx("Spring 2023 birthweight.xlsx")
```

```
## Error in read_xlsx("Spring 2023 birthweight.xlsx"): could not find function "read_xlsx"
```

R cannot find the function "read_xlsx" because while we have installed the readxl package, we have not loaded it properly for R to use. After we have installed a package, we have to load it into R's library with the *library()* function before R can use the functions inside the package.

```
library(readxl)
```

Notice that we don't need to use quotation marks around the name of the package when we are loading it with *library()*.

Now that we have loaded the sas7bdat function, we can use the *read_xlsx()* function to read in the dataset. We assign the dataset the name "bwt".

```
bwt <- read_xlsx("Spring 2023 birthweight.xlsx")
```

We can check that the dataset read in correctly with the following functions that do similar things as PROC CONTENTS.

```
#Check number of rows and columns (check dimensions)
dim(bwt)
```

```
## [1] 189   6
```

```
#See the first few lines of the dataset
head(bwt)
```

```
## # A tibble: 6 x 6
##      ID   BWT   AGE    WT SMOKE    HT
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     4   709    28   120     1     0
## 2    10  1021    29   130     0     0
## 3    11  1135    34   187     0     1
## 4    13  1330    25   105     0     1
## 5    15  1474    25    85     0     0
## 6    16  1588    27   150     0     0
```

```
#See that the columns have the right name
names(bwt)
```

```
## [1] "ID"    "BWT"   "AGE"   "WT"    "SMOKE" "HT"
```

Descriptive Statistics

We use the *mean()*, *sd()* and *quantile()* functions to find descriptive statistics for the continuous/numeric variables.

```
mean(bwt$BWT)
```

```
## [1] 2944.714
```

```
sd(bwt$BWT)
```

```
## [1] 729.187
```

```
quantile(bwt$BWT)
```

```
##   0%  25%  50%  75% 100%
##  709 2414 2977 3475 5001
```

```
mean(bwt$AGE)
```

```
## [1] 23.2381
```

```
sd(bwt$AGE)
```

```
## [1] 5.298678
```

```
quantile(bwt$BWT)
```

```
##   0%  25%  50%  75% 100%
##  709 2414 2977 3475 5001
```

```
mean(bwt$WT)
```

```
## [1] 129.8148
```

```
sd(bwt$WT)
```

```
## [1] 30.57938
```

```
quantile(bwt$WT)
```

```
##   0%  25%  50%  75% 100%
##   80  110  121  140  250
```

For the discrete/categorical variables, we use the *table()* function to get the frequency of outcomes.

```
table(bwt$SMOKE)
```

```
##
##   0   1
## 117  72
```

```
table(bwt$HT)
```

```
##
##   0   1
## 177  12
```

To get the percentages, we divide the output from *table()* by the number of observations (rows in the dataset), then multiply by 100.

```
table(bwt$SMOKE)/nrow(bwt) * 100
```

```
##
##        0        1
## 61.90476 38.09524
```

```
table(bwt$HT)/nrow(bwt) * 100
```

```
##
##         0         1
## 93.650794  6.349206
```

Calculating and Testing Correlation

To calculate correlation between all the variables of the birthweight dataset, we can use the *cor()* function.

```
cor(bwt)
```

```
##                ID         BWT         AGE          WT       SMOKE          HT
## ID     1.00000000  0.98117762  0.09504326  0.21565483 -0.16061294 -0.11202120
## BWT    0.98117762  1.00000000  0.09017566  0.18572889 -0.15071982 -0.14606266
## AGE    0.09504326  0.09017566  1.00000000  0.18007315 -0.06008380 -0.01583700
## WT     0.21565483  0.18572889  0.18007315  1.00000000 -0.06132677  0.23636040
## SMOKE -0.16061294 -0.15071982 -0.06008380 -0.06132677  1.00000000 -0.02553215
## HT    -0.11202120 -0.14606266 -0.01583700  0.23636040 -0.02553215  1.00000000
```

This gives us every calculated correlation value between every variable in the dataset. We are not interested in some of these variables, so we can also give the *cor()* function just two variables.

```
cor(bwt$WT,bwt$BWT)
```

```
## [1] 0.1857289
```

```
cor(bwt$AGE,bwt$BWT)
```

```
## [1] 0.09017566
```

To test the correlation values and get confidence intervals for the true correlation, we use the *cor.test()* function.

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

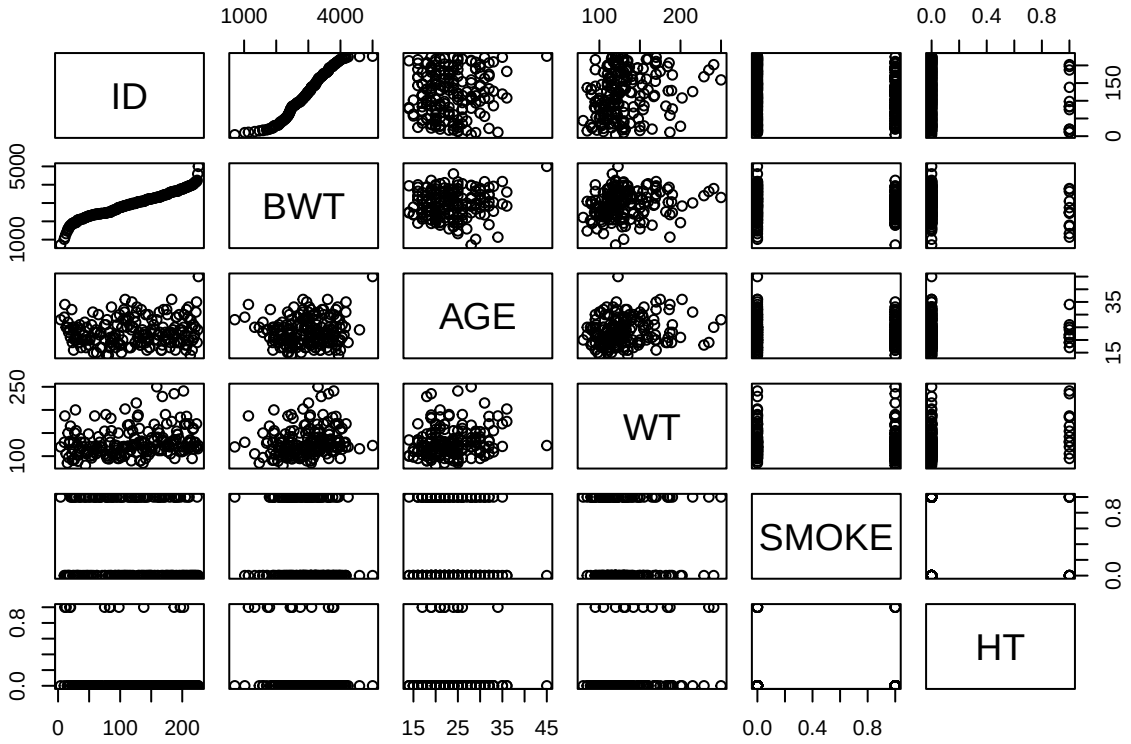
```
##
##  Pearson's product-moment correlation
##
## data:  bwt$WT and bwt$BWT
## t = 2.5848, df = 187, p-value = 0.01051
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.04416951 0.31997686
## sample estimates:
##       cor
## 0.1857289
```

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

```
##
##  Pearson's product-moment correlation
##
## data:  bwt$AGE and bwt$BWT
## t = 1.2382, df = 187, p-value = 0.2172
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.05323986  0.22994635
## sample estimates:
##        cor
## 0.09017566
```

To get pairwise plots between each of the variables, we use the *pairs()* function.

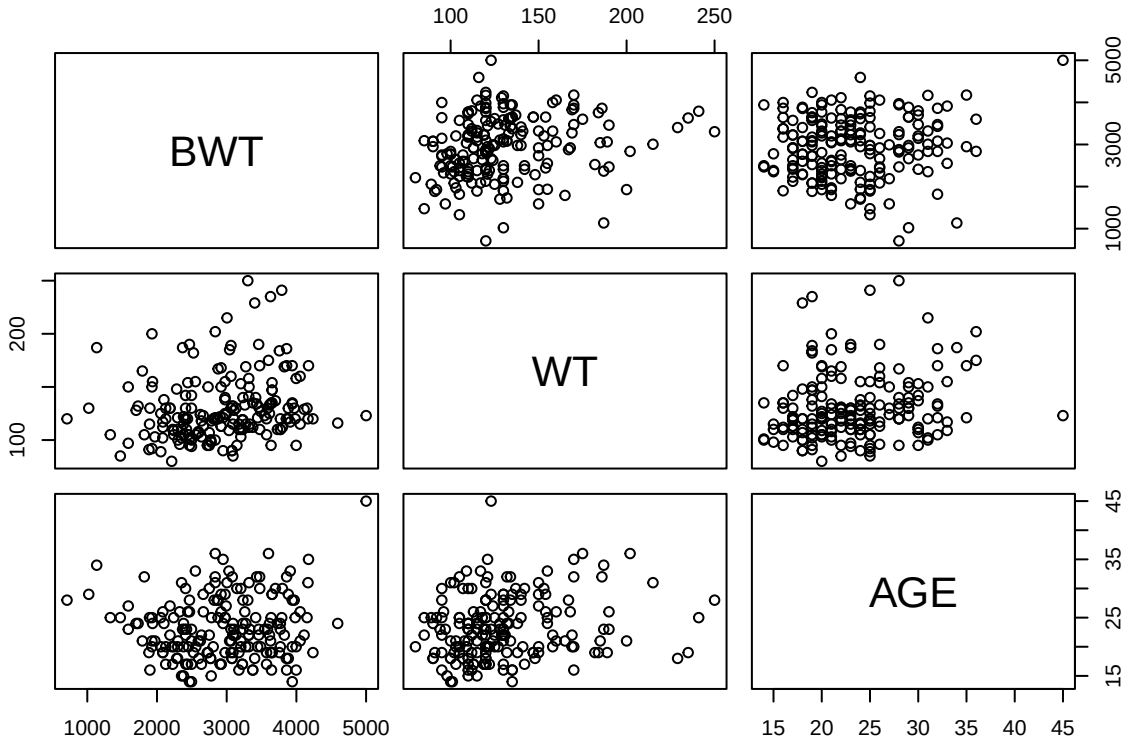
```
pairs(bwt)
```



However, this includes variables we are not interested in, like ID, and categorical variables. We can give *pairs()* the relevant columns of bwt and it will make the pairwise plots between each of them.

Recall the bracket notation *[rows,columns]* that we use to pull the columns of bwt that we want. We give it a list, created using *c()*, of the relevant column names as strings.

```
pairs(bwt[,c("BWT","WT","AGE")])
```



Fitting Multiple Linear Regression Model

Our model is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + E$

Where

Y = birthweight (g)

X_1 = mother's age (years)

X_2 = mother's weight (lbs)

X_3 = mother's smoking status (1 = smoked during pregnancy, 0 = did not smoke)

X_4 = mother's hypertension status (1 = history of hypertension, 0 = no history of hypertension)

E = random error, assumed Normal(0, σ^2)

β_0 = true y-intercept

β_i = true slope associated with X_i, adjusted for the other predictors.

We fit our model with the *lm()* function again. The arguments for the *lm()* function are a formula specifying our model and a dataset.

The formula uses the ~ operator again. The dependent variable (birthweight), goes on the left side of the ~, and the independent variables go on the right side of the ~. We are fitting a linear model where we add the effects of X_1, X_2, X_3, and X_4, so we add the variables age, weight, smoking status, and hypertension status on the right side of the formula. Remember that variable names in R are case-sensitive.

Make sure to save the results from *lm()* as an object.

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

Use the *summary()* function to see the estimates from *lm()*.

```
summary(bwt.lm)
```

```
##
## Call:
## lm(formula = BWT ~ AGE + WT + SMOKE + HT, data = bwt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2118.71  -436.99    -0.28   518.01  1855.62
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2263.532    299.860   7.549 1.97e-12 ***
## AGE            5.427      9.846   0.551  0.58220
## WT             5.184      1.755   2.953  0.00355 **
## SMOKE       -209.863    105.408  -1.991  0.04797 *
## HT          -597.684    215.769  -2.770  0.00618 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 701.4 on 184 degrees of freedom
## Multiple R-squared:  0.09433,    Adjusted R-squared:  0.07464
## F-statistic: 4.791 on 4 and 184 DF,  p-value: 0.001065
```

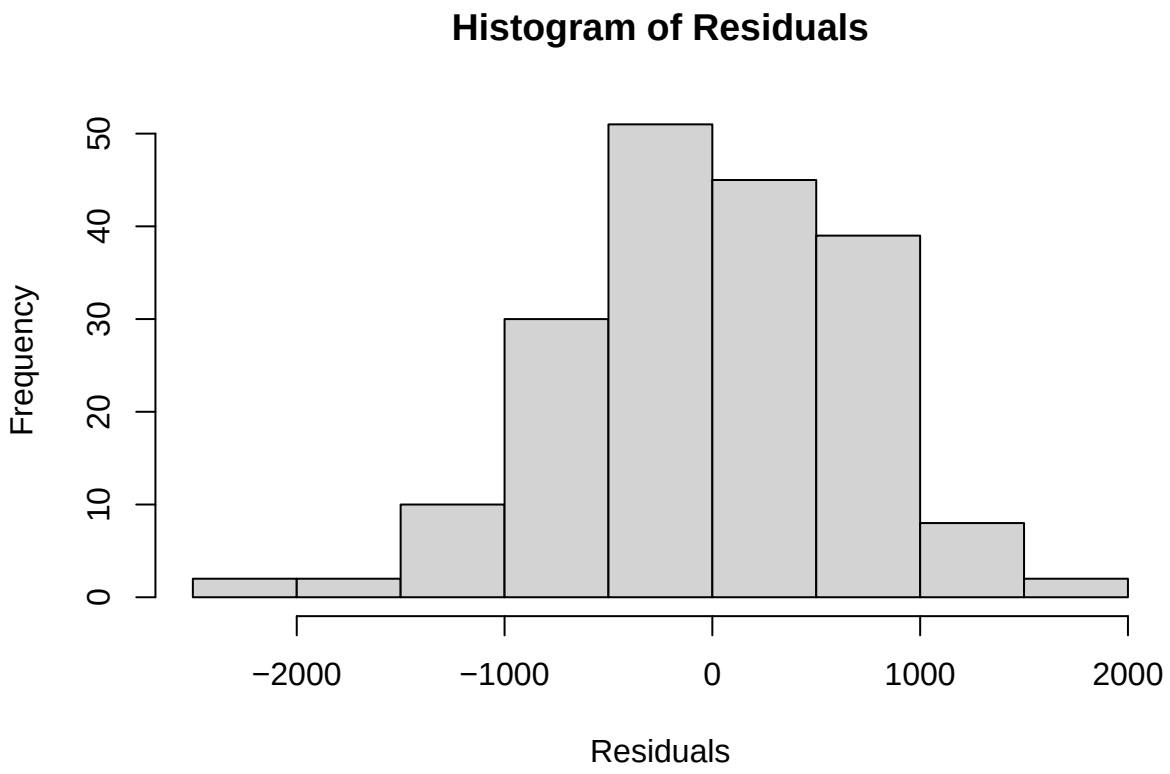
The overall F test is in the bottom row of the summary. The F-test p-value is 0.001065.

The adjusted R-squared value is just above the F statistic, and it is 0.07464.

The "Coefficients:" table has the variable name in the first column, then the estimated $\hat{\beta}_i$, the standard error, the test statistic value for the partial test, and the p-value for the partial tests.

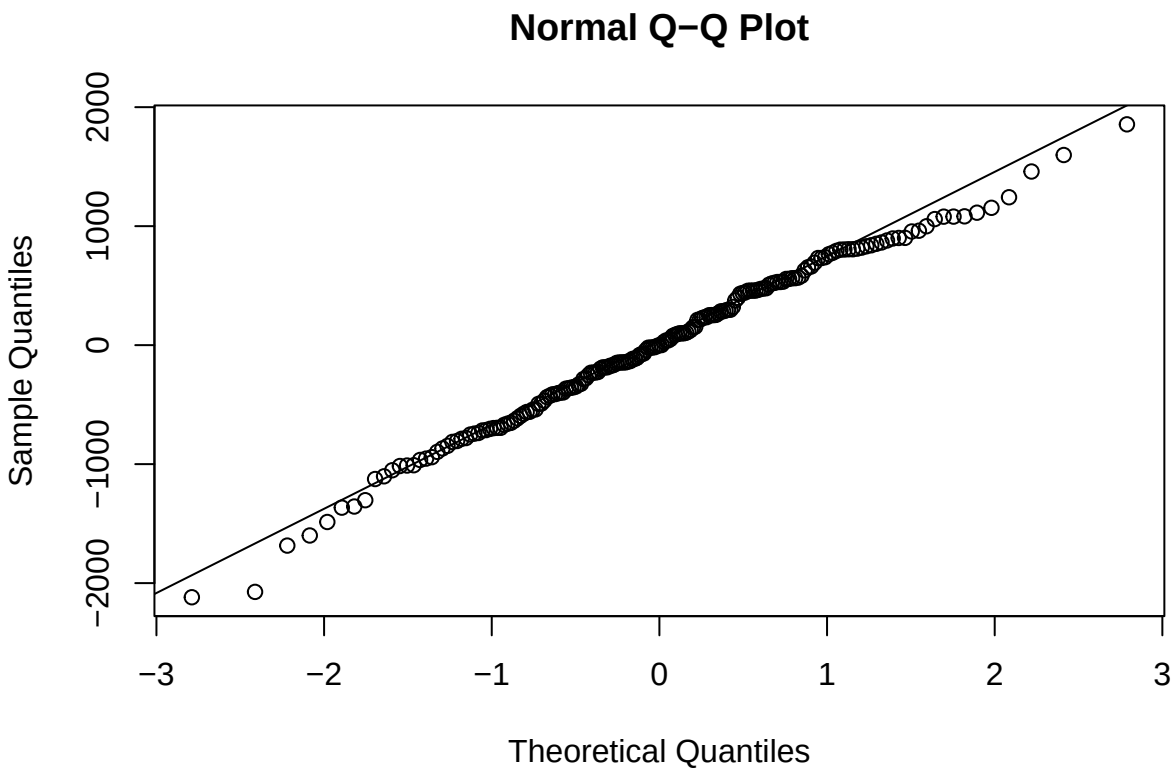
The quantiles for the residuals are also in the summary, but we can get a histogram of the residuals by giving the *hist()* function the residuals of the bwt.lm linear model object, which we get with the *residuals()* function.

```
hist(residuals(bwt.lm), main="Histogram of Residuals", xlab="Residuals")
```



We can also get the normal probability plot with the *qqnorm()* function, then the *qqline()* function to add the normal probability line to the plot.

```
qqnorm(residuals(bwt.lm))
qqline(residuals(bwt.lm))
```



You can export plots in Rstudio to put them in a report by clicking the "export" button just above the plot.