HANDOUT Stat 230 - Introduction to Multiple Linear Regression (MLR - C3.1 & 3.2)

P.B. Matheson adapted from A.S. Wagaman

MLR: Intro to Multiple Linear Regression (3.1)

For this example, we consider a data set about 48 painted turtles, where we know their sex (defined in a binary way here instead of along a continuum) and their height/length/width measurements.

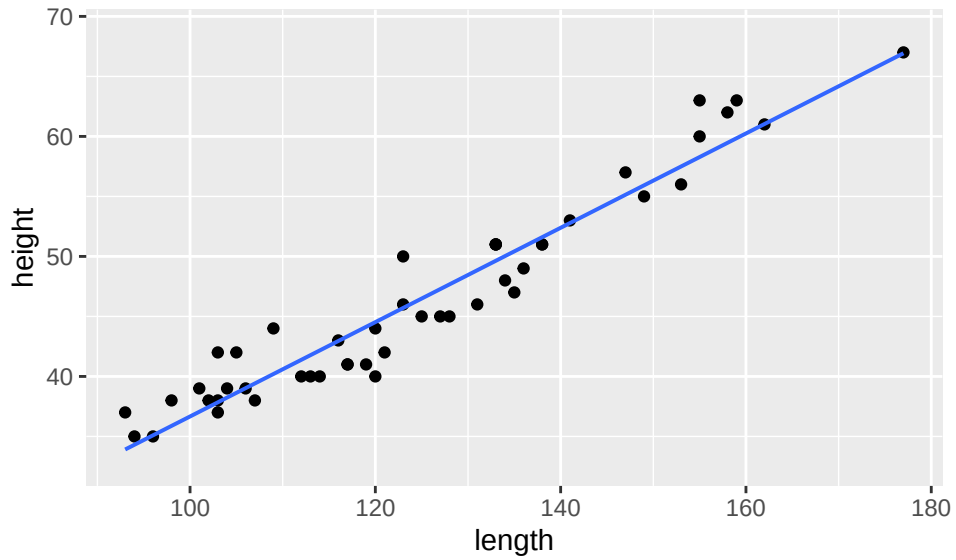
```
turtle <- read.table("https://pmatheson.people.amherst.edu/stat230/paintedturtle.txt", header = TRUE)
glimpse(turtle)
```

```
## Rows: 48
## Columns: 4
## $ length <int> 98, 103, 103, 105, 109, 123, 123, 133, 133, 133, 134, 136, 138,~
## $ width  <int> 81, 84, 86, 86, 88, 92, 95, 99, 102, 102, 100, 102, 98, 99, 105~
## $ height <int> 38, 38, 42, 42, 44, 50, 46, 51, 51, 51, 48, 49, 51, 51, 53, 57,~
## $ sex    <chr> "female", "female", "female", "female", "female", "female", "fe~
```

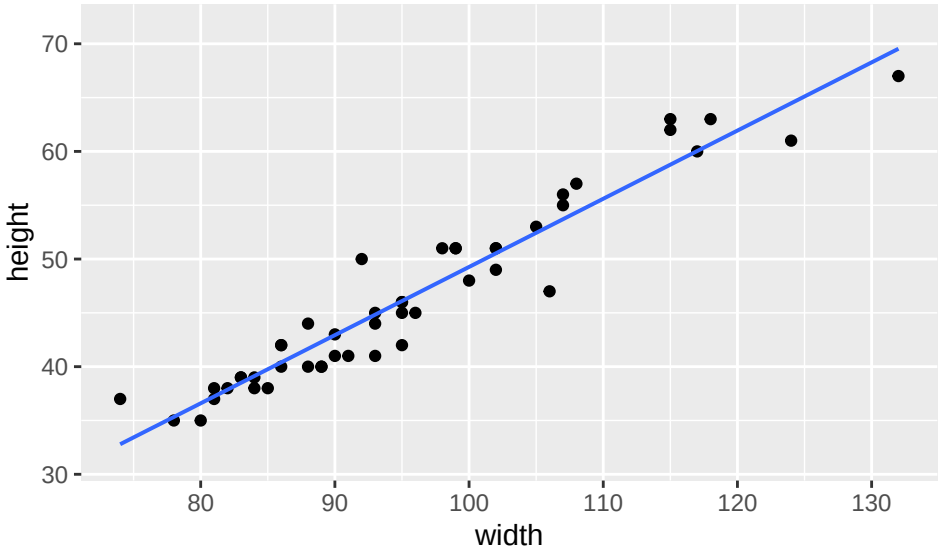
Let's try to predict the height of the turtles, using whatever variables we want. To start us off in MLR, just try using the quantitative variables. There is a qualitative variable (sex of the turtle) in the datafile but we will use it later (Section 3.3) to compare regression lines between groups.

Begin by looking at how height relates to each possible predictor - length and width.

```
gf_point(height ~ length, data = turtle) %>%
  gf_lm()
```



```
gf_point(height ~ width, data = turtle) %>%
  gf_lm()
```




```
cor(height ~ length, data = turtle)
```

```
## [1] 0.9628899
```

```
cor(height ~ width, data = turtle)
```

```
## [1] 0.9599055
```

```
#alternative - correlation matrix
cor(select(turtle, -sex))
```

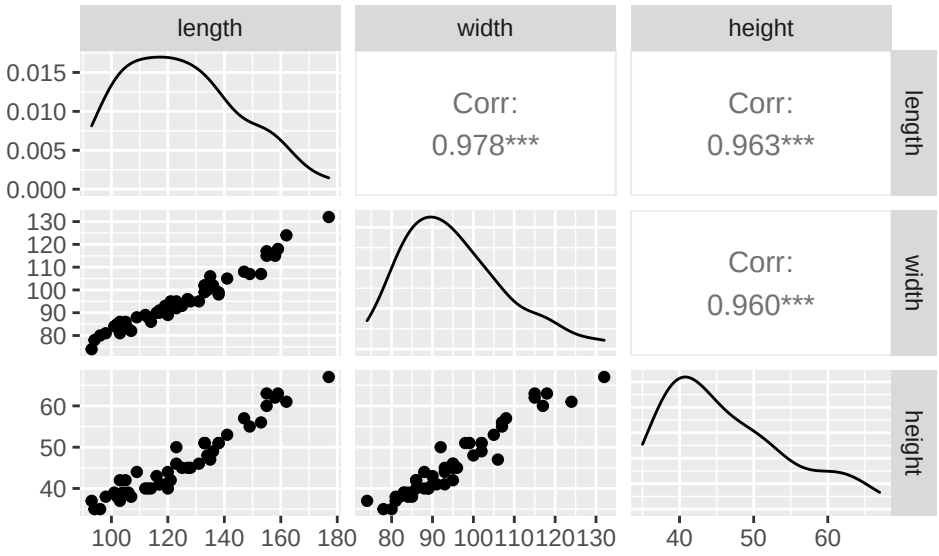
```
##           length     width    height
## length 1.0000000 0.9778869 0.9628899
## width  0.9778869 1.0000000 0.9599055
## height 0.9628899 0.9599055 1.0000000
```

```
#indicates to remove the sex variable from the output because it is categorical
```

In both cases, we see a positive linear relationship with height. We want to see how well we can predict height using both predictors instead of just one though, so we should fit our model.

Rather than making individual scatterplots, a scatterplot matrix can help visualize many relationships at once. This uses a new function *ggpairs* from the GGally package (You will note that the command library(GGally) has been added at the top of this RMD). Can you figure out which plot is which?

```
ggpairs(select(turtle, -sex))
```



You probably noticed that the correlation coefficients with the individual predictors were pretty high. So how can we decide between models - using one predictor or multiple predictors, or how to simplify the model from an even larger set of predictors. Good question! We'll study these problems soon - learning to test between models as well as do variable selection. The t-tests for slope help a little bit. Below, they suggest keeping both variables in the model.

Section 3.1 on Multiple Linear Regression (MLR)

When we have two quantitative predictors in the model, the theoretical model being fit is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon.$$

In R, the order you input the predictors is the order it works with them. So because length is first below, it is our X_1 and width is our X_2. We can flip that if desired - it won't change the equation or anything else.

```
fm <- lm(height ~ length + width, data = turtle)
summary(fm)
```

```
##
## Call:
## lm(formula = height ~ length + width, data = turtle)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6184 -1.6797 -0.0099  0.9893  4.9606
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.16826    3.10312  -2.632  0.01158 *
## length       0.22595    0.07428   3.042  0.00391 **
## width        0.27626    0.12010   2.300  0.02613 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.183 on 45 degrees of freedom
## Multiple R-squared:  0.9348, Adjusted R-squared:  0.9319
## F-statistic: 322.7 on 2 and 45 DF,  p-value: < 2.2e-16
```

From the summary output, we can see that the fitted model is: predicted height = -8.16826 + 0.226(length) + 0.276(width)

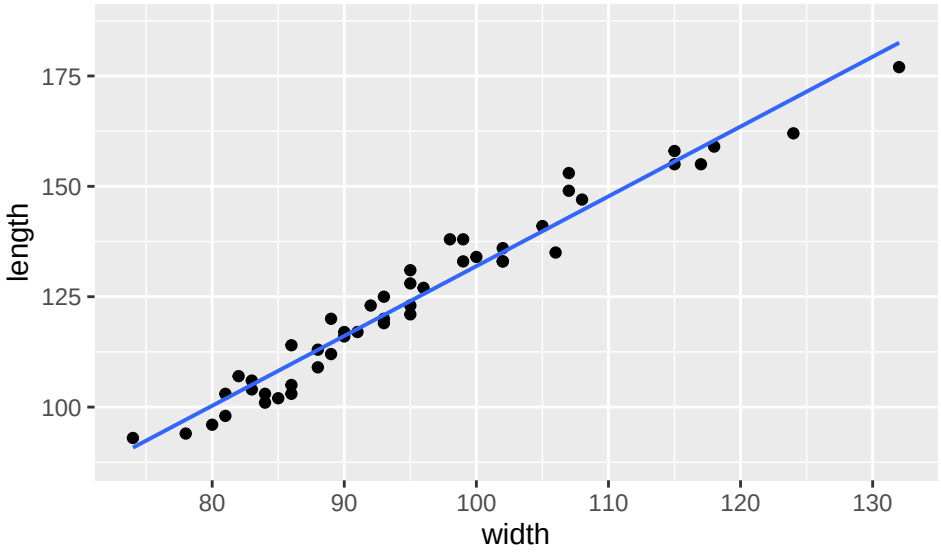
To practice interpreting the coefficients, let's try to interpret the 0.226 slope for length. This means that for each 1 unit increase in length, we expect *average* height to increase by 0.226 units *after* width has been accounted for.

OR

For each 1 unit increase in length, we expect *average* height to increase by 0.226 units while allowing for simultaneous changes in width.

Note: This does NOT say that width needs to remain constant. We can clearly see that length and width are positively correlated (scatterplot below). It doesn't make sense to suggest one might change and the other must stay fixed.

```
gf_point(length ~ width, data = turtle) %>%
  gf_lm()
```



The interpretation of the residual standard error also remains the same as with SLR. Our estimates of height will be about 2.183 units off from the true height based on this regression model. This is the size of a typical residual from this model.

Section 3.2 on Assessing the MLR Model

Conditions

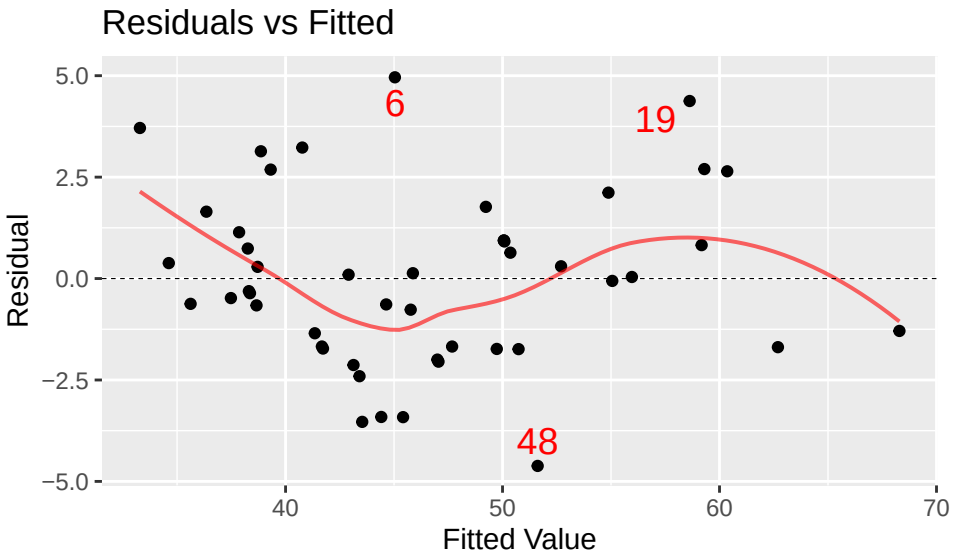
The regression model also still has the fitted values and residuals attached, and checking the conditions proceeds exactly as in SLR. They have the same breakdown, with normality and randomization only being needed if you intend to do inference. Let's check the conditions here.

```
names(fm) #you can see what the fm object contains!
```

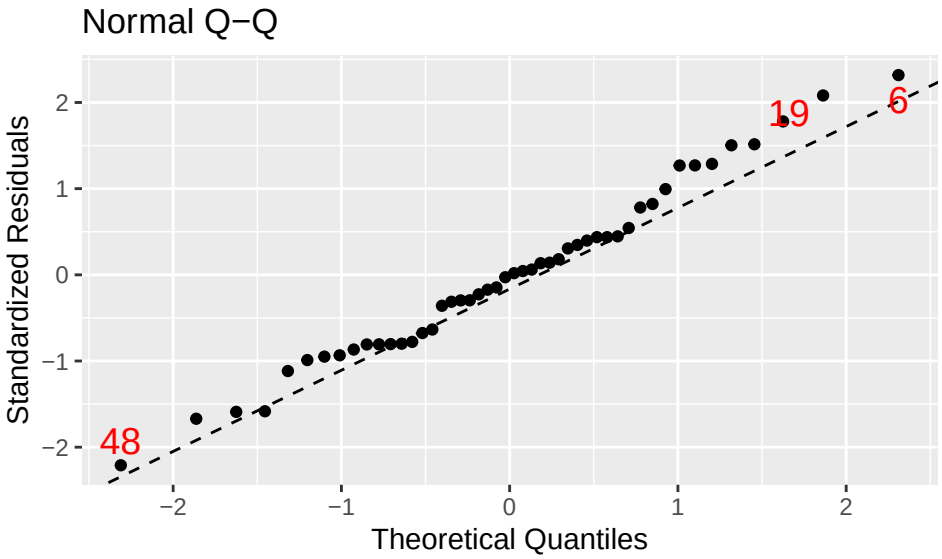
```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
mplot(fm, which = 1) #generates the residuals vs. fitted plot
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
mplot(fm, which = 2) #generates the QQ plot of residuals
```



There don't appear to be any glaring problems with the conditions, though we do see a few unusual points. Bear in mind that R always marks 3 points by default. Those 3 points are the most unusual R is identifying

- but they may not be problematic. Additionally, you could have more than 3 problematic points, but only 3 are marked.

We do have to assume that the turtles were independently selected and the data was collected through a random process (or that the sample of turtles was representative of their population) in order to do inference.

t-tests for Slope

Suppose we want to investigate whether or not individual predictors are significant in the model. Here, that means we are asking if they each contribute to the model beyond what the other variable is able to do on it's own. To do that, we can look at their individual t-test statistics or confidence intervals. For example, a t-test for the slope of length here tells you if length can help predict height above and beyond what width can do on its own.

```
msummary(fm) #you wouldn't normally need to run this a second time, just look at your existing output
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.16826    3.10312  -2.632  0.01158 *
## length       0.22595    0.07428   3.042  0.00391 **
## width        0.27626    0.12010   2.300  0.02613 *
##
## Residual standard error: 2.183 on 45 degrees of freedom
## Multiple R-squared:  0.9348, Adjusted R-squared:  0.9319
## F-statistic: 322.7 on 2 and 45 DF,  p-value: < 2.2e-16
```

```
confint(fm)
```

```
##                    2.5 %     97.5 %
## (Intercept) -14.41825971 -1.9182661
## length        0.07634158  0.3755602
## width         0.03436189  0.5181527
```

In both cases, the p-values are small - 0.00391 for length and 0.02613 for width. The confidence intervals miss 0 (why is 0 the value of interest?). We have evidence that each individual predictor is significant in the model. Bear in mind the *only* change here is the t-distribution used for reference is now a t with n-3 df because we have 2 quantitative predictors instead of just one (k=2; n-k-1 = n-2-1 = n-3). So for this model, or any other model with 2 predictors df = n-3. (This gets more complicated with categorical variables, but the df are always in the output.)

But what if we just want to know if the model (as a whole) is effective at predicting height. In other words, is the model useful? This is where we use the ANOVA ideas and associated F-test. In MLR, the ANOVA provides results about the usefulness of the ENTIRE model. It isn't equivalent to the t-tests anymore.

Overall F-test

The ANOVA F-test null hypothesis here is that the slopes of both predictors are 0. The alternative is that at least one of the slopes is non-zero. (It doesn't tell us which one, or if it's both!) I.E.

$$H_0 : \beta_1 = \beta_2 = 0 \text{ versus } H_A : \text{ At least one } \beta_i \neq 0$$

Obtaining the ANOVA table for regression is still easy. Use the aov command for a partial table or the anova command for the full table, after fitting the model, just like before.

```
aov(fm)
```

```
## Call:
##    aov(formula = fm)
##
## Terms:
##                     length     width Residuals
## Sum of Squares  3049.6512   25.2075  214.3912
## Deg. of Freedom         1         1        45
##
## Residual standard error: 2.182716
## Estimated effects may be unbalanced
```

```
anova(fm) #note the F statistics in here are NOT what you want for the overall F test
```

```
## Analysis of Variance Table
##
## Response: height
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## length     1 3049.65 3049.65 640.112 < 2e-16 ***
## width      1   25.21   25.21   5.291 0.02613 *
## Residuals 45  214.39    4.76
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The main challenge is that the ANOVA table breaks down the variance for EACH predictor, so you should see that the F statistics and p-values here don't match the one at the bottom of the usual regression summary output. In other words, these tables aren't so useful anymore, unless we want to do the computations ourselves. You can use the one reported in the usual model summary. It has the degrees of freedom for the appropriate F distribution to find the p-value provided. NOTE: If you would like to see how you can use the ANOVA table to compute sums of squares in R, an optional section has been provided at the end of the document.

R-squared and Adjusted R-squared

Next up, we can compare the R-squared (noted a muliple R squared in R output) and adjusted R-squared. Here, they are very similar values. R-squared is 0.9348 and adjusted R-squared is 0.9319. The model appears to explain over 93% of the variability in the response. The regular multiple R-squared is what you should use when asked directly what percentage of variability in the response the model explains. However, the adjusted R-squared is better for comparing models.

CIs for mean response and Prediction Intervals

Finally, we can generate confidence intervals for the mean response and prediction intervals for individual responses just like before, though plotting them isn't as easy. Here, I generate both types of intervals for two new turtles with lengths of 100 and 150, and widths of 85 and 110 respectively. First we create the new data set.

```
new.data <- data.frame(length = c(100, 150), width = c(85, 110))
glimpse (new.data)
```

```
## Rows: 2
## Columns: 2
## $ length <dbl> 100, 150
## $ width  <dbl> 85, 110
```

```
# the first "new" turtle has length 100 and width 85
# what are the length and width for the second new turtle?
```

Then we apply the predict function.

```
predict(fm, new.data, int = "confidence", level = 0.90) #for CIs for mean response
```

```
##        fit      lwr      upr
## 1 37.90869 36.67514 39.14225
## 2 56.11267 55.25403 56.97132
```

```
predict(fm, new.data, int = "prediction", level = 0.90) #for prediction intervals
```

```
##        fit      lwr     upr
## 1 37.90869 34.04099 41.7764
## 2 56.11267 52.34774 59.8776
```

You could also just get predicted intervals for your entire data set like this:

```
preddata <- predict(fm, turtle, int = "confidence", level = 0.90)
head(preddata) #shows first 6 observations
```

```
##        fit      lwr      upr
## 1 36.35176 35.41087 37.29266
## 2 38.31029 37.46343 39.15715
## 3 38.86280 37.78799 39.93762
## 4 39.31471 38.42307 40.20634
## 5 40.77102 39.98295 41.55910
## 6 45.03937 44.32027 45.75847
```

Optional - Computing Sums of Squares in R

So, if the one reported in the summary is not the one we want... how do we get it? You need to understand how it is related to the values in the ANOVA tables generated. If you want to check the formulas/math, more info is given below.

Let's take a look at the computations. The SSTotal was $\sum(y - \bar{y})^2$. We can ask R to compute that.

```
# Total sum of squares
SST <- with(turtle, sum((height - mean(height))^2))
```

In order to compute the portion of the variation explained by the model, we need to generate the fitted values (or we could just take them from the regression fitted model) because we need the y-hats. You can also get these from the augmented data set.

```
fitted <- fm$fitted.values #take directly from fm OR...
turtleaugment <- augment(fm)
names(turtleaugment)
```

```
## [1] "height"     "length"     "width"      ".fitted"    ".resid"
## [6] ".hat"       ".sigma"     ".cooksd"    ".std.resid"
```

```
fitted2 <- turtleaugment$.fitted
sum(fitted-fitted2) #check same, 0 because these are equal!
```

```
## [1] -6.32383e-13
```

NOTE: the $ command is a way of using a prefix to indicate what file the variable is coming from. Here the variable "fitted.values" is coming from "$" the dataset "fm".

Now we can compute the other sums of squares:

```
# Residual sum of squares
SSE <- sum((fitted - turtle$height)^2)
# Model/Regression sum of squares
SSModel <- sum((fitted - mean(turtle$height))^2)
```

and check that they add up:

```
SST
```

```
## [1] 3289.25
```

```
SSModel + SSE
```

```
## [1] 3289.25
```

From the ANOVA table breakdown, you can see the 3074.86 from the model IS equal to the 3049.65 + 25.21 reported for the two predictors. To complete the calculation of the test statistic, we need to create the two MS values and then take their ratio:

```
MSModel <- SSModel/2; MSModel
```

```
## [1] 1537.429
```

```
MSE <- SSE/45; MSE
```

```
## [1] 4.76425
```

```
Fstat <- MSModel/MSE; Fstat
```

```
## [1] 322.7013
```

This gives us the 322.7 reported at the bottom of the summary output, and the computer can find the associated p-value. If you want to do this yourself, the command is:

```r
pf(Fstat, 2, 45, lower.tail=FALSE) #we want the probability of being in the upper tail of an F distribu
```

```
## [1] 2.077199e-27
```

Remember you can just read both the F statistic and p-value off from the summary output. The info provided above shows you the underlying computations.

The same interpretations, etc. apply.