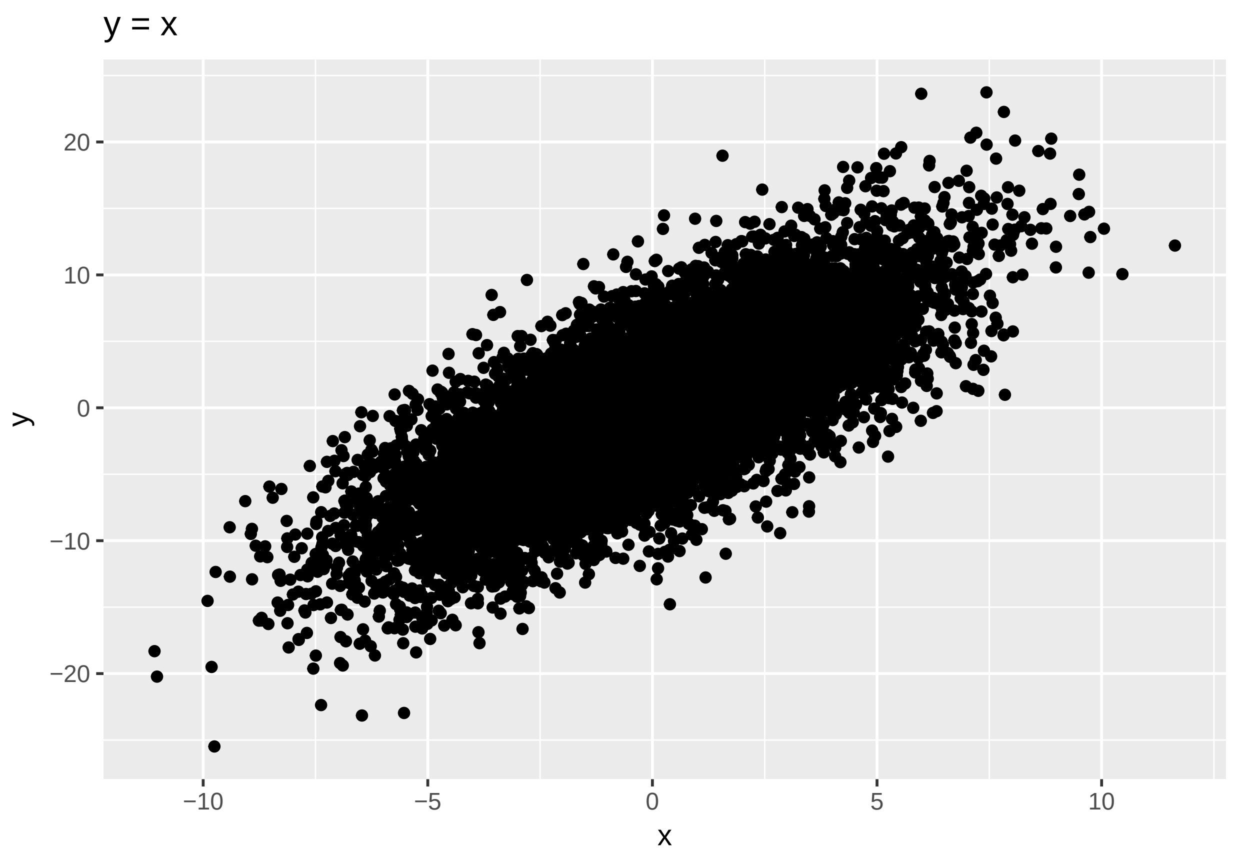
HW6_B66084

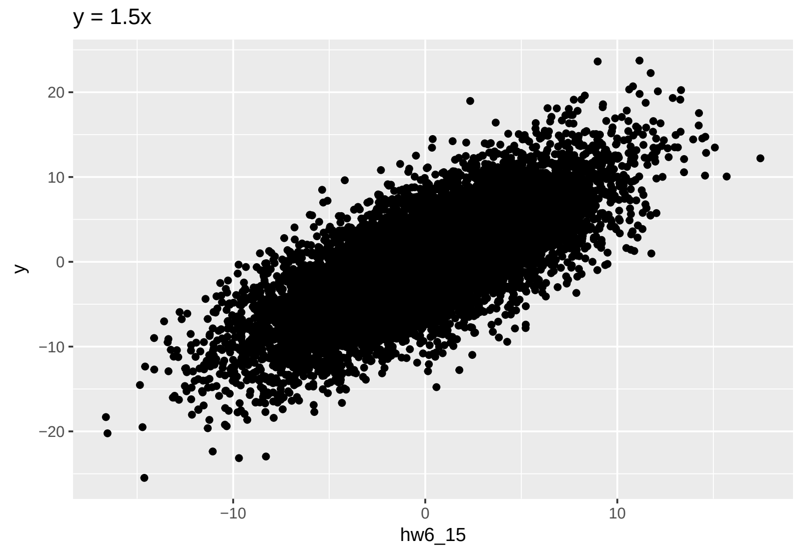
Luis Gerson Noboa Martillo

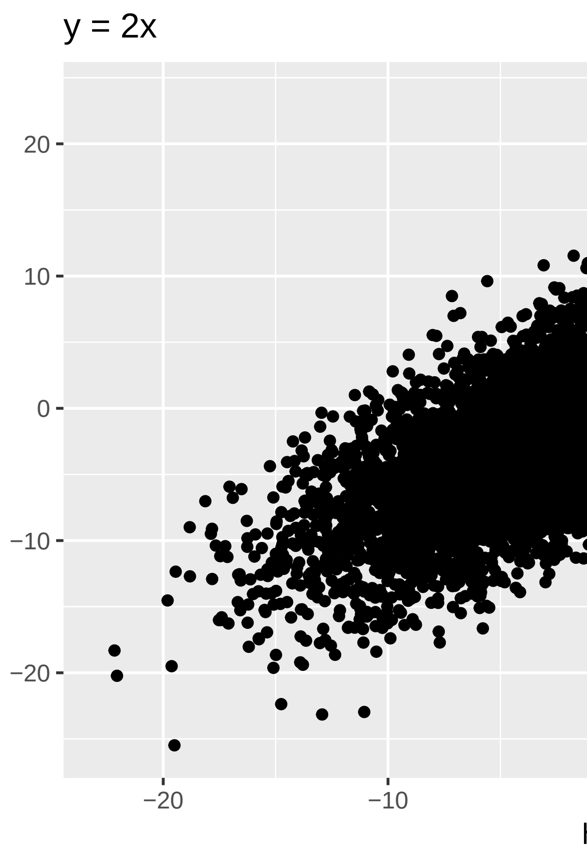
1 de abril de 2017

Exercise 1

```
ggplot(hw6, aes(x = x, y = y)) + geom_point() + ggtitle("y = x")

hw6_15 = 1.5 * (hw6$x)
ggplot(hw6, aes(x = hw6_15, y = y)) + geom_point() + ggtitle("y = 1.5x")

hw6_2 = 2 * hw6$x
ggplot(hw6, aes(x = hw6_2, y = y)) + geom_point() + ggtitle("y = 2x")
```







In order to calculate RMSE, the following function was defined:

```
## help from
## https://heuristically.wordpress.com/2013/07/12/calculate-rmse-and-mae-in-r-and-sas/

rmse <- function(e) {
    return(sqrt(mean(e^2)))
}
```

Then, errors were calculated in order to get the RMSE:

```
error1 = hw6$x - hw6$y
paste("Error for y=X:", rmse(error1))
```

```
## [1] "Error for y=X: 4.27442339031979"
```

```
error15 = hw6_15 - hw6$y
paste("Error for y=1.5X:", rmse(error15))
```

```
## [1] "Error for y=1.5X: 4.01796461964308"
```

```
error2 = hw6_2 - hw6$y
paste("Error for y=2X:", rmse(error2))
```

```
## [1] "Error for y=2X: 4.3005013241168"
```

As explained by the Wikipedia page we were linked for in the homework website, the root mean square deviation is used to measure the difference between the actual values versus the predicted. Root mean square error gives a lot of weight to values that are very wrong. There's another measure, the mean absolute error, that gives equal weight to all errors, so by analyzing RMSE, we can see how deviated are the predicted values from the actual values, and how off are the biggest values of the vector.

Obviously, a big value means that the predictions are completely off, and that big errors are very common. A small value means that errors are not that small, so the predicted data is a good indicator of the real life scenario.

In our datasets, we can see that the smallest RMSE comes when the X value is multiplied by 1.5. It displays a value of approximately 4. Other values display higuer values, reaching 4.27 and 4.30. This means that errors for these sets are bigger than for the 1.5 set.

Exercise 2

```
reg1 = lm(formula = quality ~ alcohol, data = wine)
summary(reg1)
```

```
##
## Call:
## lm(formula = quality ~ alcohol, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.06114 -0.51660 -0.02887  0.50220  2.94153
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.85178    0.18874   9.811   <2e-16 ***
## alcohol      0.36267    0.01801  20.132   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7674 on 1597 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.2019
## F-statistic: 405.3 on 1 and 1597 DF,  p-value: < 2.2e-16
```

For the first relationship, there were four values that had the lowest amount of p-value: volatile acidity, citric acid, sulphates, and alcohol. This was done by testing manually, no other fancy methods. The value for them were all displayed like $< 2.2e-16$ by R. All of the other variables had a higher p-value.

Important to note is that residual sugar's p-value is 0.4082, making it the biggest p-value out of all the combinations, so that one will be very hard to use on the subsequent models. Also, free sulfur dioxide displays a value of 0.0513, so it will probably not be useful for the model either.

At first, my first reaction was to choose alcohol because I think it is a better, more understandable comparison to make initially than volatile acidity, for example. It is easier to explain that quality has a close relation to alcohol, since that's something everyone knows, rather than using volatile acidity or sulphates. Fortunately, alcohol is the one of the four that displays the least standard error and the highest t-value, making it the ideal variable to start the analysis.

```
summary(lm(formula = quality ~ alcohol + residual.sugar, data = wine))
```

```
##
## Call:
## lm(formula = quality ~ alcohol + residual.sugar, data = wine)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0620 -0.5159 -0.0286  0.5024  2.9424
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.849670   0.190678   9.700   <2e-16 ***
## alcohol        0.362608   0.018036  20.105   <2e-16 ***
## residual.sugar 0.001078   0.013632   0.079    0.937
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7676 on 1596 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.2014
## F-statistic: 202.5 on 2 and 1596 DF,  p-value: < 2.2e-16
```

Just as an example, since the p-value for alcohol is very low, it makes the overall p-value to be low too, but the p-value for residual sugar individually is very, very high. As predicted, we won't be able to use it for the model.

```r
reg2 = lm(formula = quality ~ alcohol + volatile.acidity, data = wine)
summary(reg2)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.64793 -0.48784 -0.03594  0.47353  2.63799
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.11357    0.20042   15.54   <2e-16 ***
## alcohol           0.31405    0.01739   18.06   <2e-16 ***
## volatile.acidity -1.43045    0.10349  -13.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7255 on 1596 degrees of freedom
## Multiple R-squared:  0.2877, Adjusted R-squared:  0.2868
## F-statistic: 322.3 on 2 and 1596 DF,  p-value: < 2.2e-16
```

We can see very low values of error, t and p values if we mix alcohol and volatile acidity, indicating that they are closely related to the quality of the wine. Thus the next examinations are made with these two values. Volatile acidity was chosen instead of, for example, sulphates or citric acid because it had the highest absolute t-value and the second highest std error.

```r
reg3 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates,
    data = wine)
summary(reg3)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates,
##     data = wine)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -2.71811 -0.48270 -0.01813  0.47834  2.38775 
## 
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)       2.66419    0.21340  12.485  < 2e-16 ***
## alcohol           0.30979    0.01723  17.975  < 2e-16 ***
## volatile.acidity -1.28002    0.10579 -12.100  < 2e-16 ***
## sulphates         0.62963    0.10992   5.728 1.21e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7183 on 1595 degrees of freedom
## Multiple R-squared:  0.302,  Adjusted R-squared:  0.3007 
## F-statistic: 230.1 on 3 and 1595 DF,  p-value: < 2.2e-16
```

Out of all the remaining values, sulphates had the least p-value of them, so it was chosen for the next variable. A thing to not was that p-values start to get higher and higher, so there's less values to choose from because they exceed the 0.05 value threshold. We also see t values dropping, although std error keeps the same. R-squared values are also increasing.

A curious thing was that citric acid, which had a small p-value on its own, now displays a 0.478 value, making it impossible to use now.

```r
reg4 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    fixed.acidity, data = wine)
summary(reg4)
```

```
## 
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates + 
##     fixed.acidity, data = wine)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -2.97371 -0.46646 -0.02419  0.46861  2.46179 
## 
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)       2.31148    0.24292   9.515  < 2e-16 ***
## alcohol           0.31639    0.01733  18.257  < 2e-16 ***
## volatile.acidity -1.20164    0.10868 -11.057  < 2e-16 ***
## sulphates         0.58607    0.11059   5.299 1.32e-07 ***
## fixed.acidity     0.03260    0.01082   3.013  0.00263 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7165 on 1594 degrees of freedom
## Multiple R-squared:  0.306,  Adjusted R-squared:  0.3043 
## F-statistic: 175.7 on 4 and 1594 DF,  p-value: < 2.2e-16
```

At this level, only fixed acidity and pH are elegible for choosing, because all of the other variables have high p-values. However, we start seeing p-values closer to 0.05, unlike the previous steps in which p-values reached 10 to the power of -16 and similar. I have no idea why the std. error is so low, when it would probably have to get higher with each step. T-value is indeed getting lower, as we can see from the increase in p-value. This is the first observation that doesn't have three stars all over the variables, with fixed acidity having two only.

```r
reg5 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    fixed.acidity + chlorides, data = wine)
summary(reg5)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     fixed.acidity + chlorides, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.92408 -0.47928 -0.01835  0.46414  2.34538
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.45406    0.24504  10.015  < 2e-16 ***
## alcohol           0.30023    0.01781  16.858  < 2e-16 ***
## volatile.acidity -1.14601    0.10930 -10.485  < 2e-16 ***
## sulphates         0.77102    0.12105   6.369 2.48e-10 ***
## fixed.acidity     0.03417    0.01079   3.168 0.001564 **
## chlorides        -1.58172    0.42928  -3.685 0.000237 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7137 on 1593 degrees of freedom
## Multiple R-squared:  0.3119, Adjusted R-squared:  0.3097
## F-statistic: 144.4 on 5 and 1593 DF,  p-value: < 2.2e-16
```

By adding chlorides, it somehow decreased the p-value for sulphates and and fixed acidity. It also maintained kind of the samet-value, but standard error is now increasing significantly. R square values are also increasing.

```r
reg6 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    fixed.acidity + chlorides + total.sulfur.dioxide, data = wine)
summary(reg6)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     fixed.acidity + chlorides + total.sulfur.dioxide, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.76867 -0.46662 -0.01522  0.46267  2.31033
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.7247009  0.2540497  10.725  < 2e-16 ***
## alcohol               0.2853059  0.0181581  15.712  < 2e-16 ***
## volatile.acidity     -1.1359581  0.1088659 -10.434  < 2e-16 ***
## sulphates             0.8156525  0.1211025   6.735 2.28e-11 ***
## fixed.acidity         0.0286172  0.0108380   2.640 0.008360 **
## chlorides            -1.6283155  0.4276264  -3.808 0.000146 ***
## total.sulfur.dioxide -0.0021406  0.0005595  -3.826 0.000135 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.7107 on 1592 degrees of freedom
## Multiple R-squared:  0.3181, Adjusted R-squared:  0.3156
## F-statistic: 123.8 on 6 and 1592 DF,  p-value: < 2.2e-16
```

Total sulfure dioxide was successfully added to the model without disrupting other values that much or by increased t-value. Standard error is surprisingly small for this value, even smaller than alcohol, which is the variable that had the best numbers in relation to quality.

```
reg7 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    fixed.acidity + chlorides + total.sulfur.dioxide + density,
    data = wine)
summary(reg7)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     fixed.acidity + chlorides + total.sulfur.dioxide + density,
##     data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.80790 -0.46445 -0.01019  0.44963  2.29619
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.535e+01  1.648e+01   2.145 0.032071 *
## alcohol               2.603e-01  2.209e-02  11.784  < 2e-16 ***
## volatile.acidity     -1.092e+00  1.110e-01  -9.838  < 2e-16 ***
## sulphates             8.523e-01  1.224e-01   6.963 4.84e-12 ***
## fixed.acidity         5.206e-02  1.604e-02   3.245 0.001199 **
## chlorides            -1.633e+00  4.272e-01  -3.823 0.000137 ***
## total.sulfur.dioxide -2.059e-03  5.605e-04  -3.673 0.000248 ***
## density              -3.271e+01  1.652e+01  -1.980 0.047845 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.71 on 1591 degrees of freedom
## Multiple R-squared:  0.3198, Adjusted R-squared:  0.3168
## F-statistic: 106.9 on 7 and 1591 DF,  p-value: < 2.2e-16
```

By adding density, we see that the p-value is barely lower than 0.05, but it also makes the value from the intercept receive. T-value is getting really low now, but standard error decreased for all values.

Final model

```
reg8 = lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
    fixed.acidity + chlorides + total.sulfur.dioxide + density +
    residual.sugar, data = wine)
summary(reg8)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     fixed.acidity + chlorides + total.sulfur.dioxide + density +
##     residual.sugar, data = wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.95168 -0.45752 -0.00107  0.44699  2.29259
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.021e+01  1.936e+01   3.110 0.001904 **
## alcohol               2.347e-01  2.444e-02   9.603  < 2e-16 ***
## volatile.acidity     -1.067e+00  1.113e-01  -9.587  < 2e-16 ***
## sulphates             9.023e-01  1.239e-01   7.281 5.18e-13 ***
## fixed.acidity         6.493e-02  1.687e-02   3.850 0.000123 ***
## chlorides            -1.730e+00  4.284e-01  -4.038 5.64e-05 ***
## total.sulfur.dioxide -2.392e-03  5.762e-04  -4.152 3.47e-05 ***
## density              -5.762e+01  1.941e+01  -2.969 0.003032 **
## residual.sugar        3.755e-02  1.542e-02   2.436 0.014975 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7089 on 1590 degrees of freedom
## Multiple R-squared:  0.3223, Adjusted R-squared:  0.3189
## F-statistic: 94.54 on 8 and 1590 DF,  p-value: < 2.2e-16
```

Adding residual sugar somehow worked here, when at the beginning it didn't. P-value is just above threshold with 0.0149, while t-value keeps betting lower. This is also the final model, because adding pH, citric acid, or free sulfur dioxide ends up in getting values greater than 0.05.

Since adding density, the standard error values are very low, indicating that the values are closer to the regression line, which means that this model behaves pretty accurately. Then t-values have gotten smaller and smaller since the start of the creation of the model, while p-values have gotten higher. Since they're closely related, it makes sense that p-value increases while t-value decreases.

The estimates can be used in the following formula:

quality = 6.021e+01 + (2.347e-01 x alcohol) - (1.067e+00 x volatile.acidity) + (9.023e-01 x sulphates) + (6.493e-02 x fixed.acidity) - (1.730e+00 x chlorides) - (2.392e-03 x total.sulfur.dioxide) - (5.762e+01 x density) + (3.755e-02 x residual.sugar)

Also, as we can see, the R-squared values are pretty small, indicating that the regression is pretty close to the actual values.

```r
rmse(reg8$residuals)
```

```
## [1] 0.7069276
```

```r
help(lm)
```

```
## starting httpd help server ...
```

```
##  done
```

RMSE is also small for the regression, so we can be confident that this model is very accurate compared to the real values that were delivered to us.

Exercise 3

Hypothesis

The T test will confirm that as Iris versicolor petal width increases, Iris setosa petal width also increases.

T-test method

```r
iris = read.table("iris.data", header = FALSE, stringsAsFactors = F,
    sep = ",")

irisSubset = subset(iris, V5 != "Iris-virginica")
autoTest = t.test(irisSubset$V3 ~ irisSubset$V5)
autoTest
```

```
##
##  Welch Two Sample t-test
##
## data:  irisSubset$V3 by irisSubset$V5
## t = -39.469, df = 62.117, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.937604 -2.654396
## sample estimates:
##     mean in group Iris-setosa mean in group Iris-versicolor
##                         1.464                         4.260
```

```r
autoTest$p.value
```

```
## [1] 1.05721e-45
```

```r
# The same as doing this versicolor = subset(iris, V5 ==
# 'Iris-versicolor') setosa = subset(iris, V5 ==
# 'Iris-setosa') t.test(versicolor$V3, setosa$V3)
```

Manual t-test

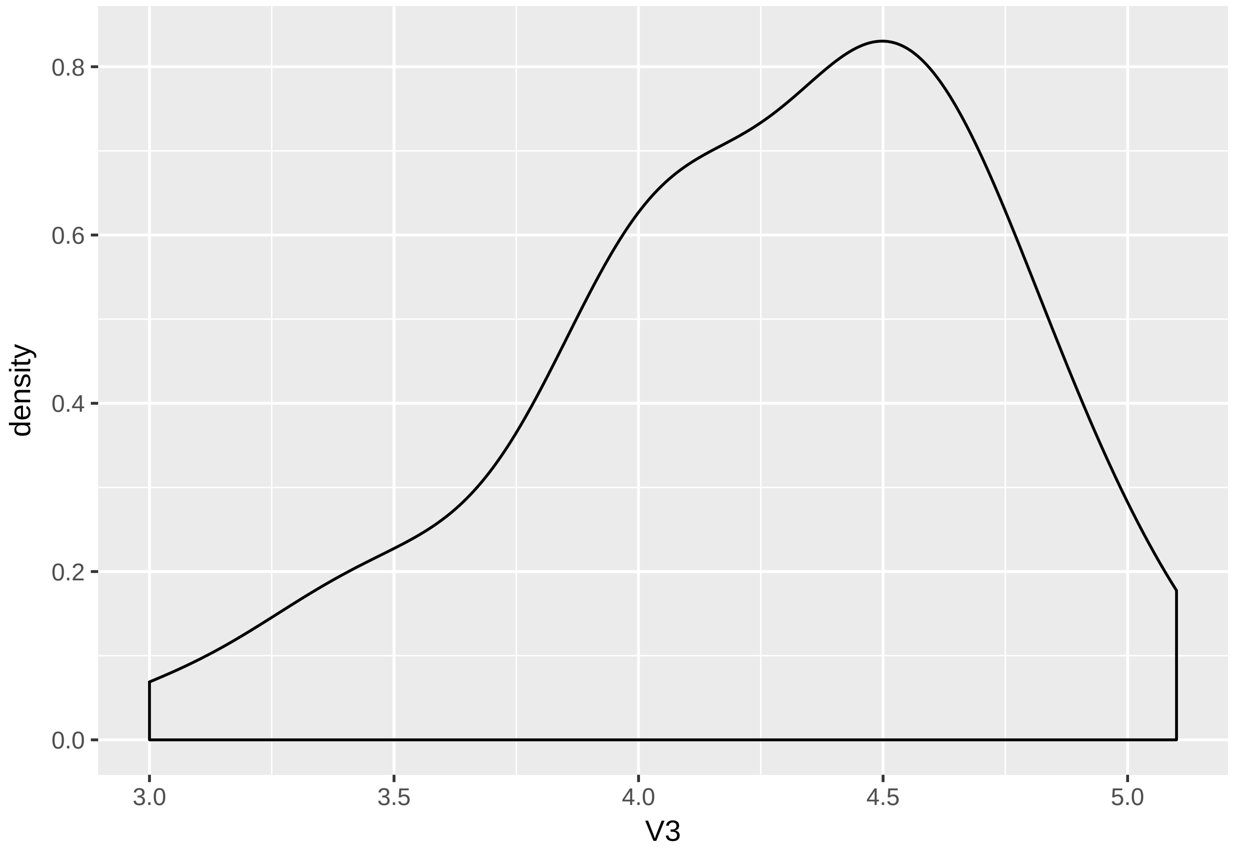
```r
versicolor = subset(iris, V5 == "Iris-versicolor")
setosa = subset(iris, V5 == "Iris-setosa")

# adapted from
# http://stats.stackexchange.com/questions/30394/how-to-perform-two-sample-t-tests-in-r-by-inputting-sa
mean1 = mean(versicolor$V3)
mean2 = mean(setosa$V3)
sd1 = sd(versicolor$V3)
sd2 = sd(setosa$V3)
values = 50

se = sqrt((sd1^2/values) + (sd2^2/values))
df = ((sd1^2/values + sd2^2/values)^2)/((sd1^2/values)^2/(values -
    1) + (sd2^2/values)^2/(values - 1))
```

```r
t = 2 * pt(-abs((mean1 - mean2)/se), df)
t
```

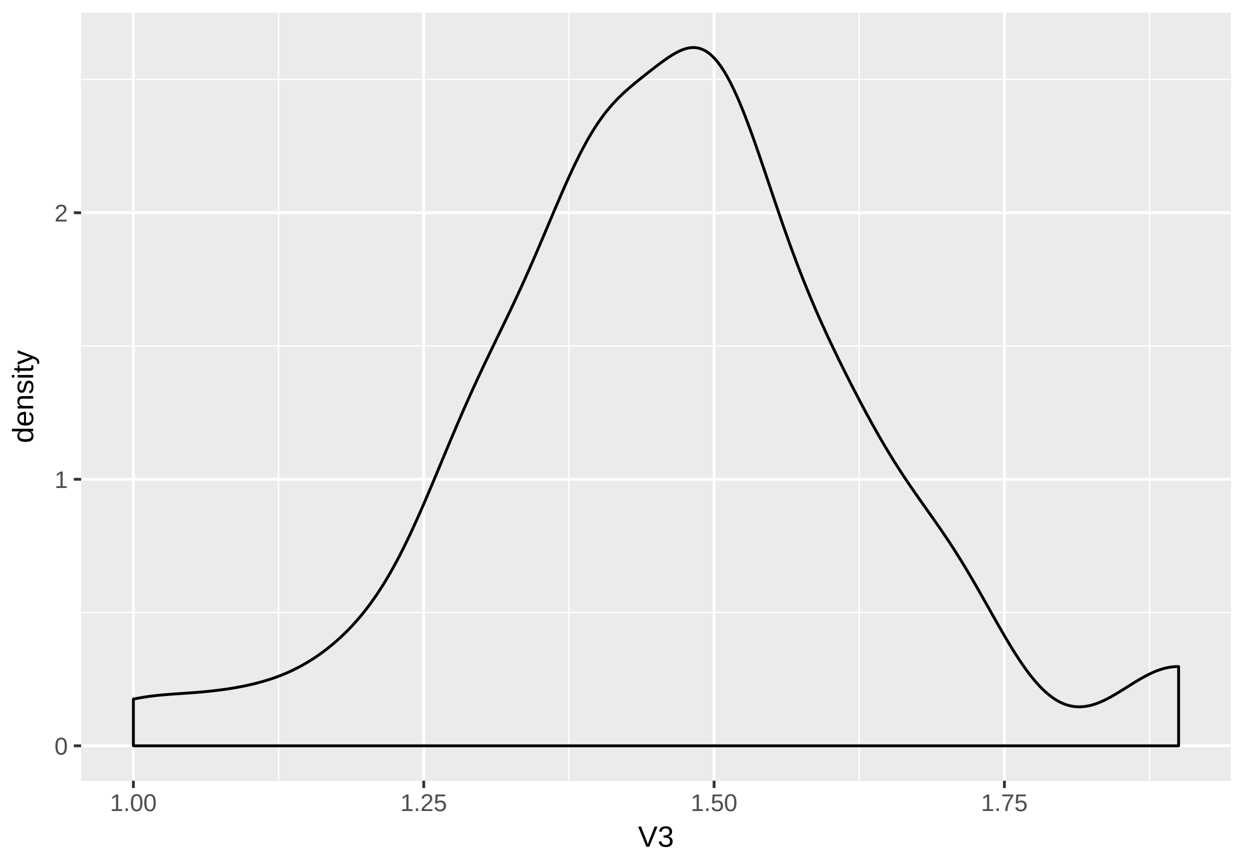
```
## [1] 1.05721e-45
```

We can see that both p-values are the same. This small p-value means that the hypothesis was correct and petal widths for both plants increase at the same ratio.

```r
ggplot(versicolor, aes(x = V3)) + geom_density()
```



```r
ggplot(setosa, aes(x = V3)) + geom_density()
```



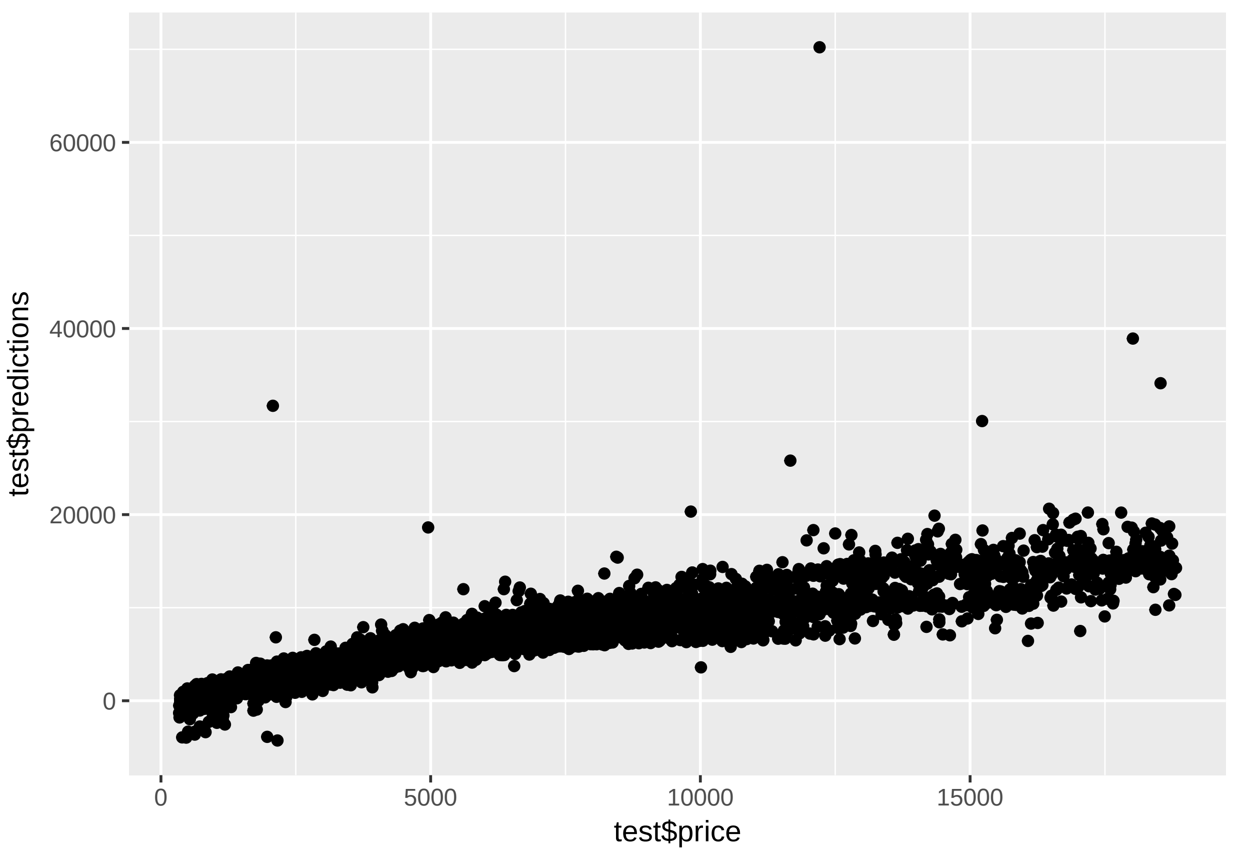
We can see from the desity plots that both graphs are actually not that similar. we see that the graph for versicolor seems more big towards the left, toward higher values. The graph for setosa seems more centered and according to a normal distribution. This is strange to see, since the tests clearly show a close dependency. I don't know how to explain this.

Exercise 4

```r
diamonds = read.table("diamonds.csv", header = TRUE, stringsAsFactors = F,
    sep = ",")
train_idx = sample(nrow(diamonds), nrow(diamonds) * 0.8)
train = diamonds[train_idx, ]
test = diamonds[-train_idx, ]

perform <- function(case) {
    r = getModel(case)

    test$predictions = predict(newdata = test, r)
    MSE = sqrt((1/nrow(test)) * sum((test$price - test$predictions)^2))
    # e = test$price - test$predictions rmse(e)

    return(MSE)
}

getModel <- function(case) {
    r = 0
    if (case == 1) {
        r = lm(formula = price ~ carat + cut + clarity + depth +
            table + x + y + z, data = train)
    } else if (case == 2) {
        r = lm(formula = price ~ carat + cut + clarity + depth +
            table + x + y + z + poly(carat, 2) + poly(depth,
            2), data = train)
    } else if (case == 3) {
        r = lm(formula = price ~ carat + cut + clarity + depth +
            table + x + y + z + carat^3 + carat^2 + depth^3 +
            depth^2, data = train)
    } else {
        r = lm(formula = price ~ carat + cut + clarity + depth +
            table + x + y + z + carat^3 + carat^2 + depth^3 +
            depth^2 + poly(x, 2) + poly(y, 2) + poly(z, 2), data = train)
    }

    return(r)
}
```

```r
rmseValues = c(perform(1), perform(2), perform(3), perform(4))

firstModel = getModel(1)
paste("RMSE for first model:", rmse(firstModel$residuals), ", prediction:",
    rmseValues[1])
## [1] "RMSE for first model: 1258.37097744854 , prediction: 1424.24866300316"

secondModel = getModel(2)
paste("RMSE for second model:", rmse(secondModel$residuals),
    ", prediction:", rmseValues[2])
## [1] "RMSE for second model: 1246.88764259693 , prediction: 1301.28137433535"
thirdModel = getModel(3)
paste("RMSE for third model:", rmse(thirdModel$residuals), ", prediction:",
    rmseValues[3])
## [1] "RMSE for third model: 1258.37097744854 , prediction: 1424.24866300316"

fourthModel = getModel(4)
paste("RMSE for fourth model", rmse(fourthModel$residuals), ", prediction",
    rmseValues[4])
## [1] "RMSE for fourth model 1220.8736234869 , prediction 58284.1769597912"
```

```r
test$predictions = predict(newdata = test, getModel(1))
ggplot(test, aes(x = test$price, y = test$predictions)) + geom_point()
```



I didn't understand the instructions so I have no idea of how to proceed.