Problem 17 in Chapter 10 asks you to fit a succession of higher degree polynomials to a small data set and to examine what happens to the r^2 and adjusted r^2 statistics. This document will step you through how to do this in R.

r^2 and adjusted r^2

First, lets review the definitions. In simple linear regression, we had a formula for the correlation coefficient, r.

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

The correlation coefficient is always between -1 and 1 with the endpoints of this range corresponding to perfect linear fits with negative or positive slopes respectively. (Recall as well that r is a measure of the adequcy of a linear fit only — you need to look at plots to conclude that a linear fit is adequate.)

The value of r^2 can be interpreted as the proportion of the variability in the response variable y that is explained by the regression — if $r = -1$ or $r = 1$, the regression explains all of the variability. A formula for r^2 that demonstrates this interpretation is here.

$$r^2 = \left(\frac{\text{Total sum of squares} - \text{Residual sum of squares}}{\text{Total sum of squares}}\right)$$

where the total sum of squares is $\sum_{i=1}^{n}(y_i - \bar{y})^2$ and the residual sum of squares is $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. The total sum of squares is the residual sum of squares after fitting a model with an intercept only.

While r^2 has this nice interpretation, its major deficiency is that it will *always* increase as you add additional variables — the residual sum of squares from a small model must be at least as large as that from a larger model of which it is a special case. So, looking at r^2 is not a good strategy for picking out a good model, because you can get increasingly better r^2 values by addiing spurious variables.

One attempt to correct for this is to compute the adjusted r^2 statistic.

$$\text{adjusted } r^2 = \left(\frac{\text{Total mean square} - \text{Residual mean square}}{\text{Total mean square}}\right)$$

where the total mean square is the total sum of squares divided by $n-1$ and the residual mean square is the residual sum of squares divided by $n-p$ where there are p parameters (including the intercept). You can show algebraically that

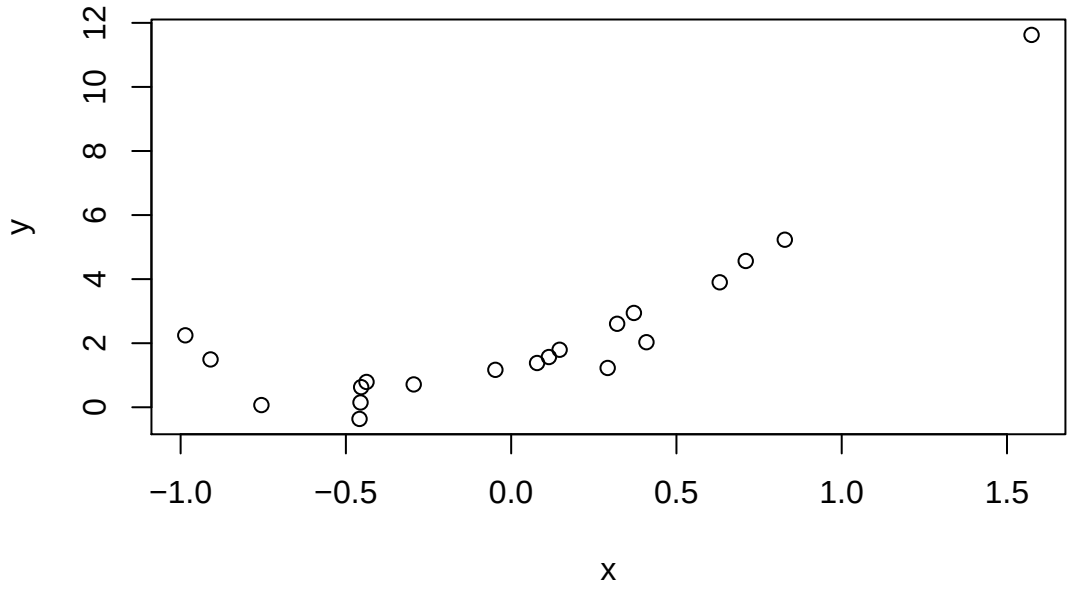
$$\text{adjusted } r^2 = 1 - (1 - r^2)\left(\frac{n-1}{n-p}\right)$$

Fictitious Data Example

Consider now a fictitious data example where the true relationship is quadratic and fits the model $\mu\{y|x\} = 1 + 2x + 3x^2$, but there is random measurement error that affects y — the observed values are the means plus independent, constant variance, mean zero, normal error. I will generate twenty data points with this model where the x values are independent standard normals (and thus quite likely to be between -3 and 3) and then fit a series of polynomial regressions.

Here is the code that generates the data and plots it. I picked the random error to have $\sigma = 0.5$.

```
> x <- rnorm(20, mean = 0, sd = 1)
> y <- 1 + 2 * x + 3 * x^2 + rnorm(20, sd = 0.5)
> plot(x, y)
```



Here is code that fits a series of progressively higher degree polynomials. Notice that the correct degree 2 model fits the data quite well — the estimates of β_0, β_1 and β_2 are close to the true values of 1, 2, and 3. Also, the estimate of σ is not too far from the true value of 0.5. You can see this in the summary of `fit2` below.

But as we add more terms, the estimates of the true nonzero parameters become less accurate and more variable — the estimates are farther away from the true values and the standard errors increase.

Also notice that the r^2 values get progressively larger, but that the adjusted r^2 values can get smaller as the size of the model increases.

Lots of R output

```
> fit0 <- lm(y ~ 1)
> fit1 <- lm(y ~ x)
> fit2 <- lm(y ~ x + I(x^2))
> fit3 <- lm(y ~ x + I(x^2) + I(x^3))
> fit4 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4))
> fit5 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
> summary(fit0)

Call:
lm(formula = y ~ 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6522 -1.5180 -0.7580  0.4023  9.3361

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2891     0.5912   3.872  0.00103 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 2.644 on 19 degrees of freedom

> summary(fit1)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9177 -0.8970 -0.4740  0.1253  4.2279

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1773     0.3599   6.049 1.02e-05 ***
x             3.3157     0.5736   5.780 1.77e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.607 on 18 degrees of freedom
Multiple R-Squared: 0.6499,        Adjusted R-squared: 0.6304
F-statistic: 33.41 on 1 and 18 DF,  p-value: 1.773e-05

> summary(fit2)

Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-0.9377 -0.3095  0.1488  0.2794  0.8186

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.1015     0.1407   7.831 4.87e-07 ***
x             2.4365     0.1927  12.642 4.52e-10 ***
I(x^2)        2.8076     0.2183  12.860 3.46e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.505 on 17 degrees of freedom
Multiple R-Squared: 0.9674,        Adjusted R-squared: 0.9635
F-statistic:   252 on 2 and 17 DF,  p-value: 2.324e-13

> summary(fit3)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3))

Residuals:
    Min      1Q  Median      3Q     Max
-0.8666 -0.1254  0.1128  0.3202  0.4637

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)   0.9806     0.1487   6.593 6.18e-06 ***
x             2.9086     0.3208   9.067 1.05e-07 ***
I(x^2)        3.2565     0.3248  10.025 2.65e-08 ***
I(x^3)       -0.5270     0.2953  -1.785   0.0933 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4753 on 16 degrees of freedom
Multiple R-Squared: 0.9728,        Adjusted R-squared: 0.9677
F-statistic: 190.6 on 3 and 16 DF,  p-value: 9.929e-13
```

```
> summary(fit4)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3) + I(x^4))

Residuals:
     Min       1Q   Median       3Q      Max
-0.97523 -0.05199  0.12856  0.27773  0.52266

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.0843     0.1724   6.289 1.45e-05 ***
x             3.2000     0.4055   7.892 1.02e-06 ***
I(x^2)        2.4362     0.7796   3.125  0.00696 **
I(x^3)       -1.1208     0.5914  -1.895  0.07750 .
I(x^4)        0.6298     0.5453   1.155  0.26618
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4704 on 15 degrees of freedom
Multiple R-Squared: 0.975,        Adjusted R-squared: 0.9683
F-statistic: 146.3 on 4 and 15 DF,  p-value: 8.008e-12
```

```
> summary(fit5)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))

Residuals:
      Min        1Q    Median        3Q       Max
-0.830211 -0.138879  0.001260  0.188362  0.602996

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3125     0.1860   7.058  5.7e-06 ***
x             2.2026     0.5819   3.785  0.00201 **
I(x^2)        0.1290     1.2628   0.102  0.92007
I(x^3)        2.8810     1.9020   1.515  0.15209
I(x^4)        3.4799     1.3895   2.504  0.02525 *
I(x^5)       -2.6985     1.2321  -2.190  0.04594 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4203 on 14 degrees of freedom
```

```
Multiple R-Squared: 0.9814,        Adjusted R-squared: 0.9747
F-statistic: 147.6 on 5 and 14 DF,  p-value: 1.346e-11
```

An R shortcut for polynomial models

Specifying polynomial regression models is tedious — you need to use the `I` function to separate out each term individually and the models get long. R does have a short cut that uses the function `poly` which can be useful if we want to fit a polynomial regression but don't care much about the values of the specific parameters. For example, consider the model

$$\mu(y|x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

and compare it to the model

$$\mu(y|x) = (\gamma_0 - a) + \gamma_1(bx - c) + \gamma_2(dx - e)^2$$

where a, b, c, d, and e are fixed known constants. In both cases, the fitted values would be identical (as long as b and d were each not 0) and it would be possible to find the βs from the γs and vice versa. (You could expand all of the powers and rearrange terms in the second model to put all of the constant terms together, all of the multiples of x together and all of the multiples of x^2 together.) In R, the function `poly` creates variables in which the fitted values and residuals are identical to those resulting from a polynomial model specified the long way, but the coefficients are different because the variables are complicated. (There is a reason for this, that we will not go into.)

The advantage is that we can specify polynomial models more efficiently. For example,

```
> new2 <- lm(y ~ poly(x, degree = 2))
> summary(new2)

Call:
lm(formula = y ~ poly(x, degree = 2))

Residuals:
    Min      1Q  Median      3Q     Max
-0.9377 -0.3095  0.1488  0.2794  0.8186

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            2.2891     0.1129   20.27 2.40e-13 ***
poly(x, degree = 2)1   9.2909     0.5050   18.40 1.16e-12 ***
poly(x, degree = 2)2   6.4938     0.5050   12.86 3.46e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.505 on 17 degrees of freedom
Multiple R-Squared: 0.9674,        Adjusted R-squared: 0.9635
F-statistic:   252 on 2 and 17 DF,  p-value: 2.324e-13
```

Compare this to what we had done earlier.

```
> fit2 <- lm(y ~ x + I(x^2))
> summary(fit2)

Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-0.9377 -0.3095  0.1488  0.2794  0.8186

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.1015     0.1407   7.831 4.87e-07 ***
x             2.4365     0.1927  12.642 4.52e-10 ***
I(x^2)        2.8076     0.2183  12.860 3.46e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.505 on 17 degrees of freedom
Multiple R-Squared: 0.9674,        Adjusted R-squared: 0.9635
F-statistic:   252 on 2 and 17 DF,  p-value: 2.324e-13
```

We can look at the range of the differences of the fitted values to double check that the fits are truly identical. Here, because of slight numerical differences in the two different fits, the numerical differences may be stored as real numbers very close to 0 instead of precisely 0 itself.

```
> range(fitted(new2) - fitted(fit2))

[1] -8.881784e-16  1.332268e-15
```

Getting rid of lots of output

Now, the output of each summary takes a lot of space. For both models, we can record the r^2 and adjusted r^2 statistics from the summary without printing it out.

Here is an example.

```
> (summary(fit2))$r.squared

[1] 0.9673657

> (summary(fit2))$adj.r.squared

[1] 0.9635264

> (summary(new2))$r.squared

[1] 0.9673657

> (summary(new2))$adj.r.squared

[1] 0.9635264
```

Here is a slick way to write a loop that will fit each polynomial model in turn and then store and print the r^2 and adjusted r^2 values.

These commands create arrays to store the answers by repeating the number 0 six times.

```
> r <- rep(0, 6)
> ar <- rep(0, 6)
```

Here is a loop to do all the fits, save, and print what we want.

```
> for (i in 1:6) {
+     fit <- lm(y ~ poly(x, degree = i))
+     r[i] <- (summary(fit))$r.squared
+     ar[i] <- (summary(fit))$adj.r.squared
+ }
> r

[1] 0.6498838 0.9673657 0.9727834 0.9750060 0.9813845 0.9817752

> ar

[1] 0.6304329 0.9635264 0.9676802 0.9683409 0.9747361 0.9733638
```

Homework

For the homework, you can do something similar for the Galileo data. This problem does demonstrate how the r^2 increases as you add variables but that the adjusted r^2 can go up and down. However, as you do this problem and read the case study, recognize that the authors have made a mistake in their understanding of physics. The good fit of a cubic model over a quadratic model has little to do with resistance. Even though each single ball would follow a parabolic arc, the horizontal distance that *different* balls would travel would not have a quadratic relationship with height unless there incline plane finished a curve that shots the balls out horizontally. Here is a description of the physics. The horizontal distance the balls travel is the product of the time they are in the air and their horizontal velocity at the time the balls leave the table (modified by resistance). If the balls flew off the table horizontally, the distance would be a quadratic function of height because the time to drop to the floor would not depend on the initial height. In this case, the parabola would be curved *up* — There would be no (theoretical) limit to the distance the ball could tavel. But in the actual experiment, the balls come off the table *in the direction of the inclined plane*. Thus, they have a downward initial vertical velocity in addition to the horizontal velocity. At high velocity, the ball would travel nearly straight along the initial direction and there would be a maximum distance that would be approached asymptotically as the height (and thus the velocity at the bottom of the inlined plane) increased. The function of distance versus height would be curved down (like in the actual data) but could not be fit perfectly by any polynomial curve. The actual function (ignoring resistance and friction) is of the form

$$distance = \left(\frac{\sqrt{\sin^2(\theta) b^2 (height)^4 + c} - \sin(\theta) b (height)^2}{a} \right) \times \cos(\theta) b (height)^2$$

which can be approximated by a polynomial in the range of the data, but is not a polynomial function. In this formula, θ is the angle of the inclined plane above horizontal. Notice that if $\theta = 0$, then $\sin(\theta) = 0$ and $\cos(\theta) = 1$ and we would expect a quadratic relationship. It would be an interesting exercise to use nonlinear regression to fit the parameters of the model based on elementary physics.