Chapter 7

Linear Regression

In God We Trust; All Others Must Bring Data
– *Willian Edwards Deming*

Now we get down to the business of actually establishing relationships between variables. As the chapter name suggests, we will first explore linear relationships between variables.

7.1 Simple Linear Regression

Lets start with a simple case of two variables - the departure delay and arrival delay variables from the NYC flights data we have seen before.

```
nyc <- read.csv("nycflightsjan13.csv")
```

```
attach(nyc)
```

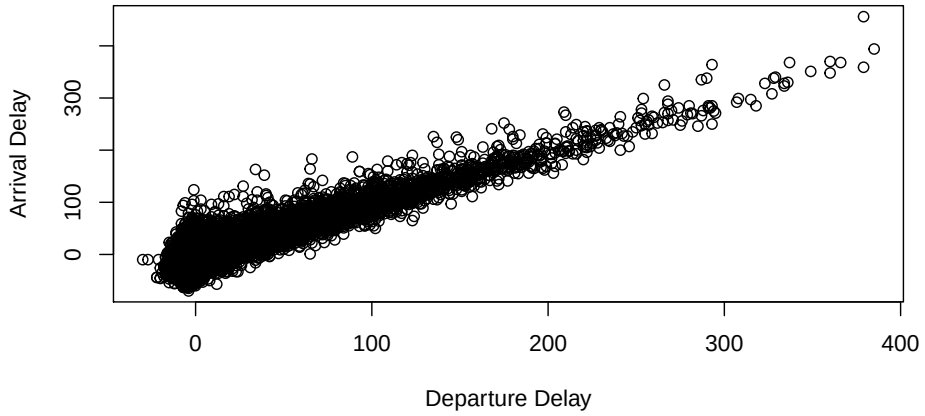
```
cor(arr_delay, dep_delay, use="complete.obs")
## [1] 0.9163321
```

Clearly the two variables are heavily correlated with each other. We can confirm this understanding by taking a quick graphical look at these variables. Figure 7.1 shows the plot of Arrival Delay vs Departure Delay.

What we essentially want to do is to draw a line through the Figure: 7.1 such that the line represents all the scatter points well. There are several ways of drawing this line. The most popular method is called OLS (Ordinary Least Square) that draws a line such that the sum of the square of distances (called **residuals** between the line and all the points is minimized. This method is commonly known as linear regression.

We can now formalize the relationship between the two variables by running a **Linear Regression**. Recall that a linear regression essentially involves modeling the relationship between a predictor variable (x) and a response variable (y) as follows: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$. This is similar to plotting a line with a y intercept of β_0 and the β_1 as the slope of the line. Through OLS Linear Regression we are trying to estimate the values of βs. Further, to ensure that the analysis is robust and applicable, we are also looking to check whether a) the model itself is statistically significant and b) the coefficients

```
plot(x=dep_delay[dep_delay < 400], y=arr_delay[dep_delay < 400],
     xlab="Departure Delay", ylab="Arrival Delay")
```




```
# Focusing only on departure delay < 400 mins for more readable plot
```

Figure 7.1: Scatter plot of Arrival Delay vs Departure Delay

are statistically significant.

The term **Statistically Significant** has a specific meaning here. It means that we can reject the null hypothesis of no effect with more than 95% (or any other level you choose, the default is 95%) confidence. That means that the probability that what we are observing is because of pure chance and randomness is less than 5%. Note that the probability of the analysis being flat out wrong is non-zero. That's why we need to be careful in interpreting the results of a regression analysis.

We can run the linear regression using the `lm()` command.

```
m <- lm(arr_delay ~ dep_delay)
```

We have saved the result of the model in the object `m`. We can now extract relevant information from this object.

```
coef(m) #provides coefficients
## (Intercept)   dep_delay
##   -4.057010    1.020178
confint(m) #provides confidence interval for the coefficients
##                 2.5 %    97.5 %
## (Intercept) -4.259533 -3.854486
## dep_delay    1.014800  1.025556
```

Now that we know the coefficients, we can now draw the fitted line that models the relationship between Arrival Delay and Departure Delay as shown in Figure 7.2. This is the same line we described in our earlier discussion of

```
plot(x=dep_delay[dep_delay < 400], y=arr_delay[dep_delay < 400],
     xlab="Departure Delay", ylab="Arrival Delay")
abline(coef(m), lwd=2, col="red")
```

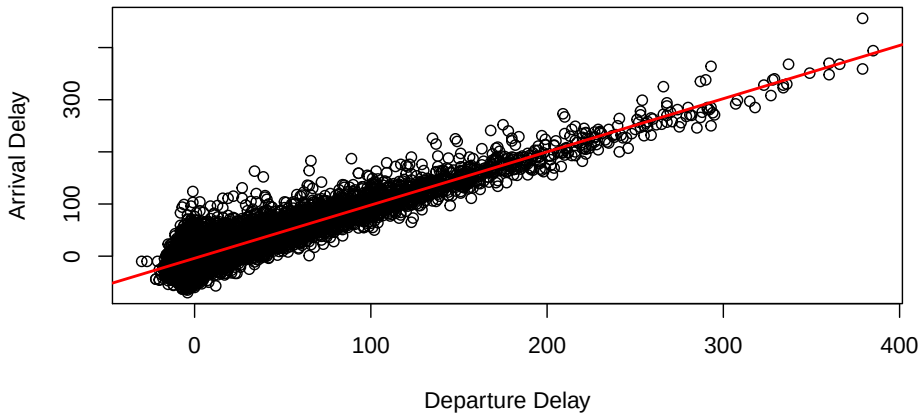


Figure 7.2: Plotting Fitted Values of Regression Model

OLS - this line minimizes the sum of the squares of the distance between all the points and this line.

As you can see that the line is not a perfect fit for all the points. The model provides a handy statistics called the Coefficient of Determination or the R-Square to check just how good of a fit the line actually is. To get R-Square and other useful information, we can use the command (summary) provides a good balanced amount of details.

```
summary(m)
##
## Call:
## lm(formula = arr_delay ~ dep_delay)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.185  -9.903  -1.024   9.057 132.371
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.057010   0.103326  -39.26   <2e-16 ***
## dep_delay    1.020178   0.002744  371.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 16.19 on 26396 degrees of freedom
##   (606 observations deleted due to missingness)
## Multiple R-squared:  0.8397,Adjusted R-squared:  0.8397
## F-statistic: 1.382e+05 on 1 and 26396 DF,  p-value: < 2.2e-16
```

Important things to note here is that the model itself has a **p-value** of close to zero. p-value can be interpreted as the probability of something happening by chance, by randomness. So first thing we see is that the probability of us getting this result by chance is virtually nil. So that's a good start.

Now we see that the p-value for the coefficient of Departure Delay (or β_1) in our model is again close to zero. This means that the probability that the predictor (in this case Departure Delay) has no impact (in other words $\beta_1 = 0$) on the response (in this case Arrival Delay) is close to zero as well. So we can conclude that Departure Delay has a statistically significant relationship with Arrival Delay. As the value of the coefficient is 1.02, we can conclude that, on an average, each minute of Departure Delay increases the Arrival Delay by 1.02 minutes.

Finally, the model has an R-Square of 0.8397. This can be interpreted as a goodness-of-fit indicator. We can say that the model explains 83.97% of the variance in the response variable. The R-Square value is susceptible to number of predictor variables - adding more predictor variables, even if they are not significant, increases the R-Square value. To correct for this over-estimation, the Adjusted R-Square can be used instead. As the model summary shows, the Adjusted R-Square value is same as R-Square value as we have only one predictor variable right now.

Multiple Regression

The basic concept of OLS Linear Regression can be easily extended to more than one predictor variables. For illustration, lets look at the following dataset that has measurement of Systolic Blood Pressure, Age (in years) and Weight (in lbs) for patients.

```
bp <- read.csv("bp.csv")
```

We can consider that Blood Pressure of a patient is a function of Age of patient and Weight of patient. So we can specify the following linear regression model.

```
bpmodel <- lm(bp ~ age + weight, data=bp)
summary(bpmodel)
##
## Call:
## lm(formula = bp ~ age + weight, data = bp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3233  -3.6146   0.1924   3.8740  12.0820
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.27980    4.75515   8.891 2.38e-16 ***
## age          0.94348    0.08792  10.731  < 2e-16 ***
## weight       0.25135    0.04526   5.553 8.16e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.457 on 217 degrees of freedom
## Multiple R-squared:  0.8394,Adjusted R-squared:  0.8379
## F-statistic: 567.1 on 2 and 217 DF,  p-value: < 2.2e-16
```

We can see in the model summary results that the model itself - our argument that a patient's age and weight have a significant relationship with patient's blood pressure - is indeed statistically significant (model p-value $<$ 0.05). The coefficients of our two predictor variables are statistically significant as well with p-value for both coefficients $<$ 0.05. We can interpret that for each year's increase in age, blood pressure rises by 0.94 points. Further, for each lbs increase in weight, blood pressure increases by 0.25. These two predictor variables explain about 84% of the variance in blood pressure as shown by the R-Square value.

Prediction Based on Regression Model

Once we have a model, we can use the model to predict new values. If, for example, a person is of 60 years age and 140 lb weight, we can predict corresponding blood pressure. We can get either a point prediction or an interval prediction corresponding to a confidence interval.

```
newvalue <- data.frame(age=50, weight=150)
#A Point Prediction
predict(bpmodel, newdata=newvalue)
##        1
## 127.1567
#Confidence Interval
predict(bpmodel, newdata=newvalue, interval="confidence")
##        fit      lwr      upr
## 1 127.1567 124.7148 129.5985
#Prediction Interval
predict(bpmodel, newdata=newvalue, interval="prediction")
##        fit      lwr      upr
## 1 127.1567 116.1275 138.1858
```

As the results show, a person with age 50 years and weight 150 years is predicted to have a blood pressure value of 127.16. This is the point estimate. The 95% confidence interval for the mean blood pressure value is 124.71 - 129.60. The 95% prediction interval for the point estimate is 116.13 - 138.19.

7.2 Regression Diagnostics

OLS Regression depends on the underlying data following several assumptions. If these conditions are not met then regression results need to be further investigated as they may not be accurate. Following are the key requirements:

Normality

Like much of Statistics, OLS works best with data that follows Normal Distribution. Specifically, linear regression assumes that the residuals are normally distributed. The residuals are assumed to be independent and with a mean of zero.

We can plot residuals to check their normality. See Figure: 7.3 for four such plots. First plot is a histogram of residuals - looks pretty well distributed for our case. Second plot is a Q-Q Plot of residuals - again we see that residuals stick to the normal line very well. Plot 3 shows residuals against fitted values while Plot 4 shows Cook's Distance. Visual inspection of these charts does not show specific patterns that may violate normality of residuals. We will use Cook's Distance plot in our discussion of outliers in coming sections.

We can also use the Shapiro-Wilk Normality test as shown below:

```
shapiro.test(residuals(bpmodel))
##
##  Shapiro-Wilk normality test
##
## data:  residuals(bpmodel)
## W = 0.9934, p-value = 0.4379
```

We can see that the null hypothesis of normality was not rejected as the p-value > 0.05. So we can conclude that the assumption of normality is satisfied in our model.

Linearity

As the name suggests, we are only exploring linear relationships between predictors and the response variable. A good graphical way to explore linearity is to check the plots created by the `residualPlots()` function in the `car` package. As Figure: 7.4 suggests, weight seems to have a straight linear relationship with. Age, however, shows some curvature that can be further explored.

It is important to note that OLS only needs residuals to be normal. Predictor variables need not be distributed normally. This allows us to do nearly limitless data transformation (taking square, cube, square-root, log etc.) and still meet the linearity assumption. For example, we could very well have run a regression analysis with log transformed values of age and weight as predictor variables.

```
par(mfrow=c(2,2))
hist(x = residuals(bpmodel), xlab = "Value of residual", main="")
plot(x=bpmodel, which=2) #QQ Plot
plot(x=bpmodel, which=1) #Residuals vs Fitted
plot(x=bpmodel, which=4) #Cook's Distance
```

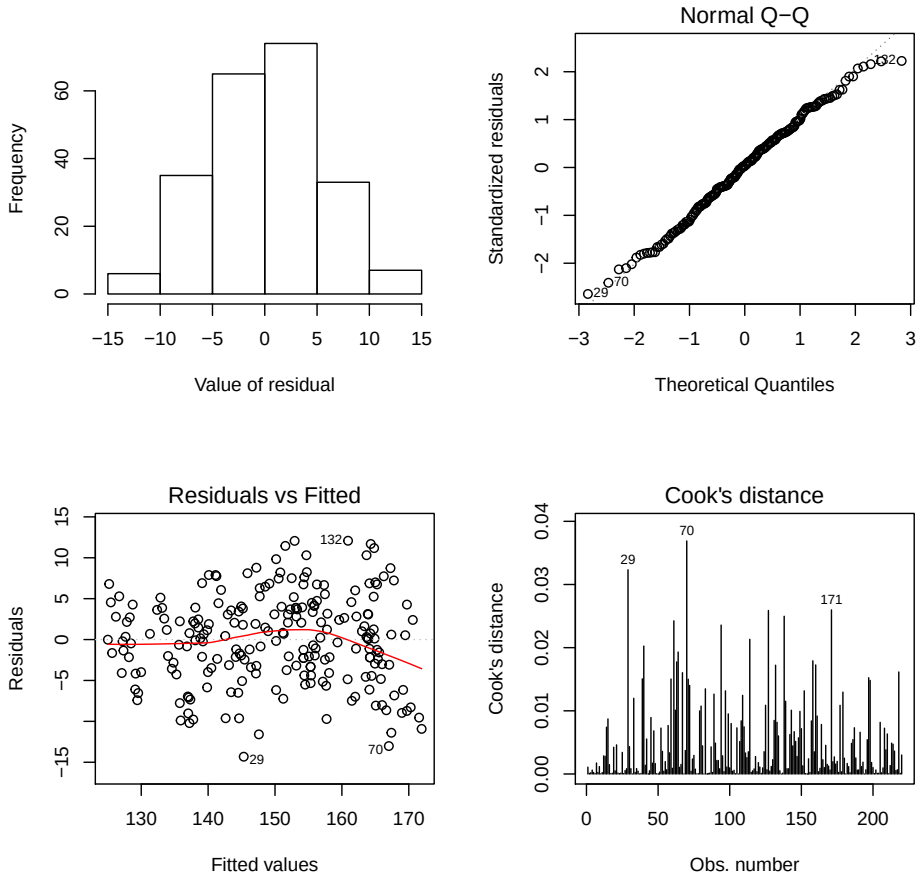
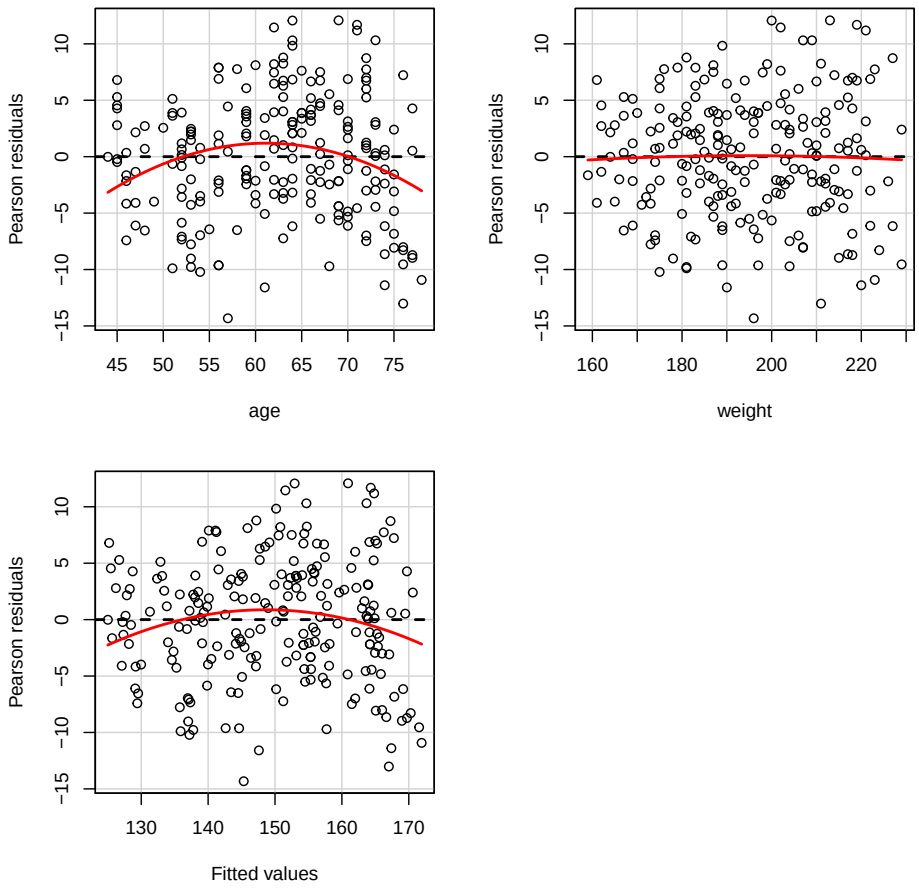


Figure 7.3: Histogram of Residuals

```
residualPlots(bpmodel)
```




```
##             Test stat Pr(>|t|)
## age            -3.251    0.001
## weight         -0.262    0.794
## Tukey test     -2.359    0.018
```

Figure 7.4: Plots to Check Linearity

```
newbp <- lm(bp ~ log(age) + log(weight), data=bp)
summary(newbp)
##
## Call:
## lm(formula = bp ~ log(age) + log(weight), data = bp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.0462  -3.8825   0.4448   3.5703  12.0816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -338.664     28.593 -11.844  < 2e-16 ***
## log(age)      57.062      5.272  10.823  < 2e-16 ***
## log(weight)   48.124      8.727   5.515 9.89e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.393 on 217 degrees of freedom
## Multiple R-squared:  0.8432,Adjusted R-squared:  0.8417
## F-statistic: 583.3 on 2 and 217 DF,  p-value: < 2.2e-16
```

Key here is to build a model that is theoretically consistent. Running a regression analysis and getting a bunch of coefficients is the easy part - the hard part is to design a model in the first place that makes sense. As they say - Correlation is not Causation. Just because a model is statistically significant, it does not mean that the model is also correct.

Homegeniety of Variance

The OLS model assumes that all the residuals have a constant variance. It is conventional to look at the plot in Figure: 7.5 to check for potential heterogeneity in the model, also called heteroscedasticity. As the plot show, there is small trend in the residuals but it may not be significant.

We can further explore the possibility of heteroscedasticity by using the Breuch Pagan test available as the `bptest()` function in the `lmtest` package.

```
bptest(bpmodel)
##
##  studentized Breusch-Pagan test
##
## data:  bpmodel
## BP = 7.83, df = 2, p-value = 0.01994
```

The Breuch Pagan test tests the null hypothesis of no heteroscedasticity. As can be seen from the results, the null hypothesis is rejected at 95% confidence interval and we do **not** have constant variance of residuals in our model.

```
plot(x=bpmodel, which=3)
```

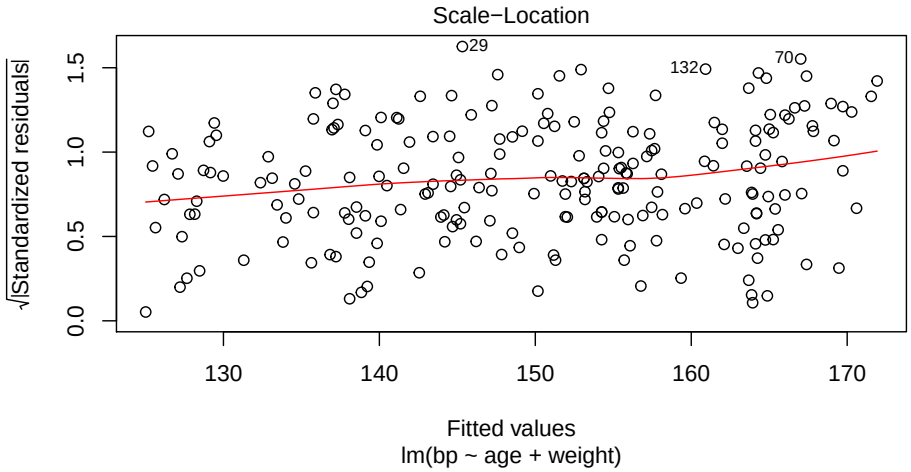


Figure 7.5: Plot to Check for Heteroscedasticity

Uncorrelated Predictors

The OLS model assumes that the predictor variables are independent and not related to each other. Any significant correlation between predictor variables would make the model results less reliable. Correlation between predictor variables, called MultiCollinearity can be estimated using Variance Inflation Factors. Variance Inflation Factors can be calculated using the `vif()` function. A VIF value less than 10 is considered acceptable.

```
vif(bpmodel)
##      age   weight
## 4.51628 4.51628
```

We can see from the output above that correlations between predictor variables is not a concern in our model.

Independent Residuals

We can if residuals are correlated with each other using the Durbin-Watson test. This test is available as function `dwtest` as part of the `lmtest` package. Like the Breuch Pagan test, this test looks at null hypothesis of no autocorrelation and tries to reject it. A p-value > 0.05 shows that the null hypothesis could not be rejected and we do not have statistically significant evidence of autocorrelation.

```
dwtest(bpmodel)
##
##  Durbin-Watson test
##
```

```
## data:  bpmodel
## DW = 2.1197, p-value = 0.8192
## alternative hypothesis: true autocorrelation is greater than 0
```

Outliers

Outlier values are those that differ greatly from other values of predictors or the response variables. Outliers can skew the regression line and can have significant influence on the regression results. Outliers can be visually detected in various diagnostic plots we have done so far. Formal detection of outliers usually considers two approaches: Leverage and Influence. Leverage statistics are designed to identify observations which have predictor values that are far away from the rest of the data. Influence statistics look at which values play too large a role in the regression model.

We can measure the change in the regression results as a result of deleting an observation. This value is called `DFBETAs`. It can be calculated using the `dfbetas` function.

```
dfb <- dfbetas(bpmodel)
head(dfb)
##    (Intercept)           age       weight
## 1 -0.039521410  0.0040605801  0.017320382
## 2 -0.011804765 -0.0053884549  0.009100589
## 3 -0.010800476 -0.0186179048  0.016753416
## 4  0.010021037 -0.0212482412  0.006672112
## 5  0.003553237  0.0065673756 -0.004930005
## 6 -0.008912326  0.0003038909  0.004934012
```

Output above shows that the first data point, if removed, will change the intercept by -0.039. Similarly We can measure the influence that an observation has on its fitted value with the function `dffits`. DFFITS can be interpreted in the same way as DFBETAS.

```
dff <- dffits(bpmodel)
head(dff)
##           1           2           3           4           5           6
## -0.05685922 -0.01827616 -0.02433017 -0.04307776  0.02656855  0.01340052
```

While the DFFITS are good for measuring the influence on a single fitted value, Cook's Distance measures the influence an observation on all of the fitted values simultaneously. We saw a plot of Cook's Distance in 7.3. We can also generate Cook's Distance values using the `cooks.distance` function.

```
cooksd <- cooks.distance(bpmodel)
head(cooksd)
##            1            2            3            4            5
## 1.081293e-03 1.118300e-04 1.981837e-04 6.209621e-04 2.362273e-04
##            6
## 6.013163e-05
```

We can also formally test for outliers using the `outlierTest()` function in the `car` package.

```
outlierTest(bpmodel)
##
## No Studentized residuals with Bonferonni p < 0.05
## Largest |rstudent|:
##    rstudent unadjusted p-value Bonferonni p
## 29 -2.68033          0.0079222           NA
```

This function identifies significant outliers. As we can see, our current dataset does not have any significant outliers. However, there still would be observations that will have more influence on the model than other. These influential observations can be identified using the `influence.measure()` function.

```
influence.measures(bpmodel) #Output removed to save space
```

7.3 Improving Regression Model

In this section we will explore various techniques for building the most appropriate regression model. We will start by checking how we can specify our model to have higher order variables like square of values.

Polynomial Regression

We will consider a new dataset called *trees* for this portion. The dataset has the following columns: $Volume$, $Girth$ and $Height$. Here $Volume$ is the response variable and $Girth$ and $Height$ are the predictor variables. We can quickly explore this data by building a scatterplot shown in Figure: 7.6.

As we can see, the data does not seem linear. Values towards the right are curving upwards. We can try to capture this non-linear effect by specifying a square term in the model as follows: $y = \beta_0 + \beta_1 x_1 + \beta_1 x_1^2 + \epsilon$. However, including a square term usually adds a large dose of multicollinearity (correlation between predictor variables) to the model. To reduce the negative impact of multicollinearity, it is suggested that data should be standardized - rescaled to have a mean = 0. We can do this in R as follows:

```
treemodel <- lm(Volume ~ scale(Girth) + I(scale(Girth)^2))
summary(treemodel)
##
## Call:
## lm(formula = Volume ~ scale(Girth) + I(scale(Girth)^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4889 -2.4293 -0.3718  2.0764  7.6447
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       27.7452     0.8161  33.996  < 2e-16 ***
## scale(Girth)      14.5995     0.6773  21.557  < 2e-16 ***
```

```
attach(trees)
plot(Girth, Volume)
```

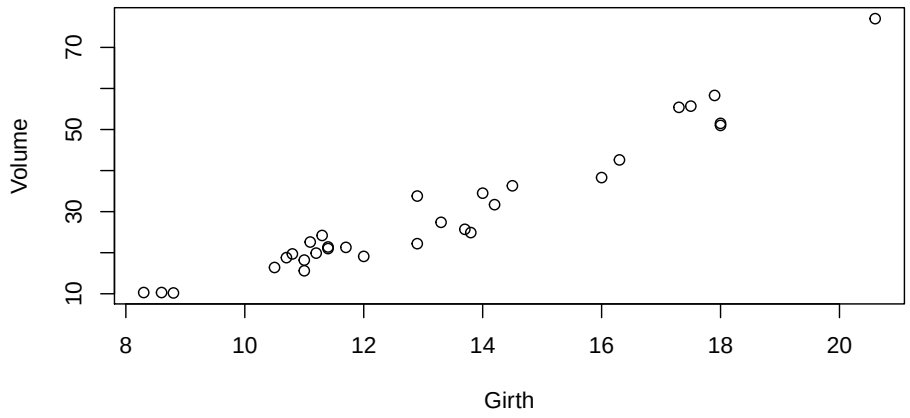


Figure 7.6: Volume vs Girth of Tree Trunks

```
## I(scale(Girth)^2)   2.5067     0.5729   4.376 0.000152 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.335 on 28 degrees of freedom
## Multiple R-squared:  0.9616,Adjusted R-squared:  0.9588
## F-statistic: 350.5 on 2 and 28 DF,  p-value: < 2.2e-16
```

Results show that both *Girth* and $Girth^2$ have significant coefficients. The model is strongly significant as well and has a very high R-Square value. We can graphically see the how model fits in Figure: 7.7. As we can see, the model fits the data very well.

When including higher order variables, we should be mindful of the principle of `parsimony` which states that if a higher order term x^m is part of the model then all lower order terms x, x^2 through x^{m-1} should be part of the model as well.

Interaction Between Predictors

Even though predictor variables are assumed to be independent of each other, they may interact in their influence on the response variable. That is - the impact of one predictor variable in the response variable may depend on the value of a different predictor variable. We can capture such interaction effects by including an additional term in our model. Such interaction terms are usually formed by multiplying one predictor variable by another. This results in model specifications like: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1:2} x_1 x_2 + \epsilon$

```
plot(scale(Girth), Volume)
lines(fitted(treemodel) ~ scale(Girth))
```

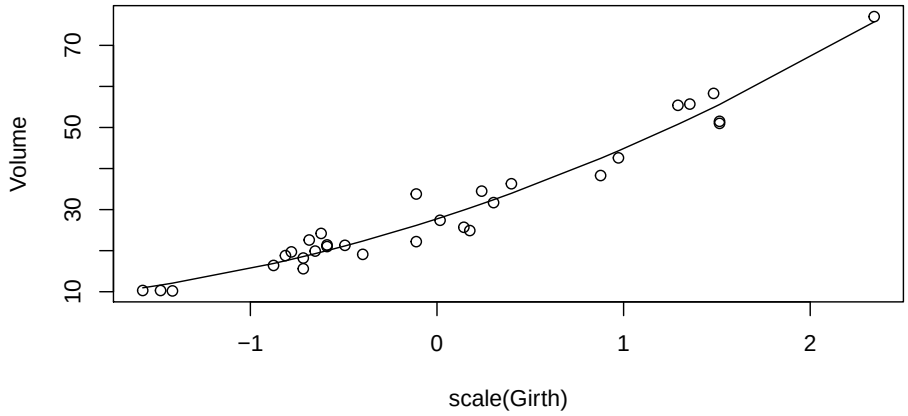


Figure 7.7: Quadratic Model of Volume vs Girth of Tree Trunks

where $\beta_{1:2}$ indicate a coefficient of an interaction term. We can do this in R as follows:

```
treeint <- lm(Volume ~ Girth + Height + Girth:Height)
summary(treeint)
##
## Call:
## lm(formula = Volume ~ Girth + Height + Girth:Height)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5821 -1.0673  0.3026  1.5641  4.6649
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.39632   23.83575   2.911  0.00713 **
## Girth        -5.85585    1.92134  -3.048  0.00511 **
## Height       -1.29708    0.30984  -4.186  0.00027 ***
## Girth:Height  0.13465    0.02438   5.524 7.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.709 on 27 degrees of freedom
## Multiple R-squared:  0.9756,Adjusted R-squared:  0.9728
## F-statistic: 359.3 on 3 and 27 DF,  p-value: < 2.2e-16
```

We can see that the interaction term has a significant and positive coefficients. This means that the impact of *Girth* on *Volume* is higher for higher values of *Height*. Note that we can also interpret the results the other way around - that the impact of *Height* on *Volume* is higher for higher values of *Girth*. Continuing with the principle of parsimony discussed earlier, it is recommended to include all the constituents of the interaction effect (called main effects) be included in the model as well.

Categorical Predictor Variables

So far all our predictor variables have been numeric and continuous. However, often we get data that is Qualitative or Categorical in nature. In R vocabulary - a *factor* with many *levels*. These *levels* may be unordered (nominal values) or ordered (ordinal values).

Our *trees* dataset does not have any categorical variables - but we can create one. We will recode the *Height* variable into a categorical variable *is.tall* which will take the value *yes* for $Height > 80$ and the value *no* for $Height <= 80$.

```
trees$is.tall <- cut(trees$Height, breaks = c(-Inf, 80, Inf),
                     labels = c("no", "yes"))
class(trees$is.tall)
## [1] "factor"
```

Now we want to include *is.tall* variable in our regression model. R handles such categorical variables by converting them to a binary, 0/1 variable called a `dummy variable`. We can think of this as an *indicator* variable - it indicates whether the tree is tall or not. Essentially, we have a model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ where x_2 will take values 0 or 1. Note that when the value is 0, the model reduces to $y = \beta_0 + \beta_1 x_1 + \epsilon$. OTOH, when the value is 1, the model becomes $y = (\beta_0 + \beta_2) + \beta_1 x_1 + \epsilon$. The net effect is of just changing the y-intercept by the amount of β_2. We can run this model in R:

```
treesdummy <- lm(Volume ~ Girth + is.tall, data=trees)
summary(treesdummy)
##
## Call:
## lm(formula = Volume ~ Girth + is.tall, data = trees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8961 -2.3250  0.7692  1.7068  7.1240
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -35.8044     3.0530 -11.728 2.56e-12 ***
## Girth         4.8986     0.2303  21.271  < 2e-16 ***
## is.tallyes    4.7695     1.7003   2.805  0.00904 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
treestall <- split(trees, trees$is.tall)
treestall[["yes"]]$fit <- predict(treesdummy, treestall[["yes"]])
treestall[["no"]]$fit <- predict(treesdummy, treestall[["no"]])
plot(Volume ~ Girth, data = trees, type = "n")
points(Volume ~ Girth, data = treestall[["yes"]], pch = 1)
points(Volume ~ Girth, data = treestall[["no"]], pch = 2)
lines(fit ~ Girth, data = treestall[["yes"]] )
lines(fit ~ Girth, data = treestall[["no"]])
```

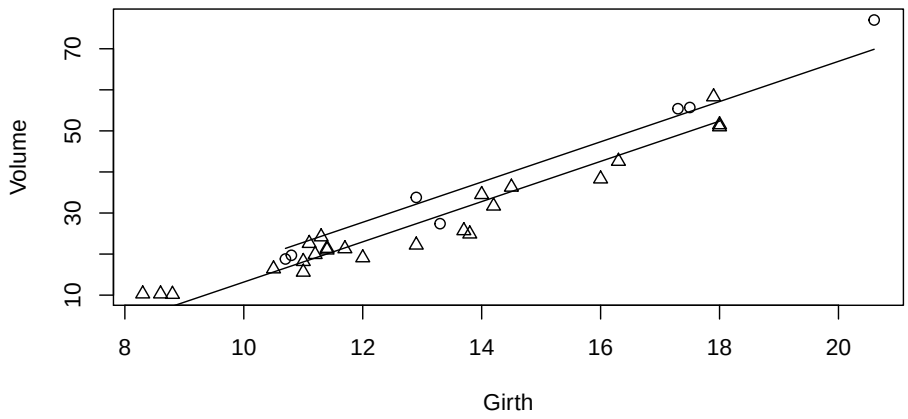


Figure 7.8: Impact of Dummy Variable

```
##
## Residual standard error: 3.823 on 28 degrees of freedom
## Multiple R-squared:  0.9495,Adjusted R-squared:  0.9459
## F-statistic: 263.3 on 2 and 28 DF,  p-value: < 2.2e-16
```

Regression results show that the model is significant and all coefficient estimates are statistically significant as well. We can conclude that the response variable differs for trees with $is.tall$ = yes and trees with $is.tall$ = no. The mean amount of difference is 4.7695. We can explore the result graphically as shown in Figure: 7.8.

Non-Linear Regression

We have really been sidestepping the fact that we really know how tree's trunk volume is related to its girth and height. Volume of a cylinder of radius r and height h is simply $\pi r^2 h$. So ideally we would want a regression model where $Girth$ and $Height$ are multiplied together - $y = \beta_0 x_1^{\beta_1} x_2^{\beta_2}$. This form of course does not meet the linearity assumption. However, we can transform the model into a linear one by taking log of both sides. This will give us the following model: $logy = Ln\beta_0 + \beta_1 logx_1 + \beta_2 logx_2$. Lets see how

this looks in R.

```
treeslog <- lm(log(Volume) ~ log(Girth) + log(Height), data = trees)
summary(treeslog)
##
## Call:
## lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.168561 -0.048488  0.002431  0.063637  0.129223
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
## log(Girth)   1.98265    0.07501  26.432  < 2e-16 ***
## log(Height)  1.11712    0.20444   5.464 7.81e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08139 on 28 degrees of freedom
## Multiple R-squared:  0.9777,Adjusted R-squared:  0.9761
## F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

As we can see, this is our best model yet!

7.4 Model Selection

When we have several predictor variables available, the important question arises of which of those predictors to include in the regression model. Ideally, variable section should be done based on theoretical criteria and not analytical convenience. However, some times it may be necessary to choose variables to build the best model possible in absence of theoretical considerations. In this section we will look at stepwise regression using the command `step()` as a method for selecting predictor variables. Then we will consider how to compare two different models using the function `anova()`.

step: stepwise method for variable selection
anova: used for comparing two models

The mathematical criterion for deciding whether to include or remove predictor variables is called `AIC` - Akaike Information Criterion - which is defined for a linear regression with K predictor variables as: $AIC = (SS_{res}/\sigma^2)+2K$. For best model performance, we want the smallest AIC value. As number of predictors grow, without a corresponding significant decline in magnitude of residuals, the AIC goes up. Hence, minimizing AIC ensures that only predictors with significant impact on residuals are included in the model.

Stepwise Regression

We will explore our last model of trees' volume using Stepwise Regression (saved as model *treeslog*. The default stepwise mode is `backward elimination` where we start with a full model (all predictors included) and

test removing a predictor variable in each step. The combination with lowest AIC is selected as the preferred model.

```
step(object = treeslog, direction = "backward")
## Start:  AIC=-152.69
## log(Volume) ~ log(Girth) + log(Height)
##
##               Df Sum of Sq    RSS      AIC
## <none>                      0.1855 -152.685
## - log(Height)  1    0.1978 0.3832 -132.185
## - log(Girth)   1    4.6275 4.8130  -53.743
##
## Call:
## lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)
##
## Coefficients:
## (Intercept)   log(Girth)  log(Height)
##      -6.632        1.983        1.117
```

We can see in the output that when all predictors are included, the model has an AIC of -152.69. The output table shows that if any of the predictors are removed, the AIC value increases. Hence, we conclude that the full model is the best mode and we do not need to remove any predictors from our model.

We can also run the stepwise regression using `forward selection`. In this case we start with an empty model (only the intercept) and then additional predictors are added in each step until minimum AIC value is reached.

```
emptymodel <- lm(log(Volume) ~ 1, data=trees) #Start with an empty model
step(object = emptymodel, direction = "forward",
     scope = log(Volume) ~ log(Girth) + log(Height))
## Start:  AIC=-38.82
## log(Volume) ~ 1
##
##               Df Sum of Sq    RSS      AIC
## + log(Girth)   1    7.9254 0.3832 -132.185
## + log(Height)  1    3.4957 4.8130  -53.743
## <none>                      8.3087  -38.817
##
## Step:  AIC=-132.19
## log(Volume) ~ log(Girth)
##
##               Df Sum of Sq     RSS     AIC
## + log(Height)  1   0.19778 0.18546 -152.69
## <none>                     0.38324 -132.19
##
## Step:  AIC=-152.69
## log(Volume) ~ log(Girth) + log(Height)
##
```

```
## Call:
## lm(formula = log(Volume) ~ log(Girth) + log(Height), data = trees)
##
## Coefficients:
## (Intercept)   log(Girth)  log(Height)
##      -6.632        1.983        1.117
```

Output shows that the empty model (only an intercept) has an AIC of -38.82. In the first step one predictor is added and AIC reaches -132.19; then in the second step second predictor is added and AIC reaches the minimum value of -152.69. We reach the same optimal solution as in the backward elimination process before.

Model Comparison

We can directly compare two models as well. We will create a new model first for comparison purposes.

```
trees$random <- rnorm(nrow(trees)) #Random variable
model.A <- lm(log(Volume) ~ log(Girth) + log(Height),
              data = trees)
model.B <- lm(log(Volume) ~ log(Girth) + log(Height)
              + trees$random, data = trees)
```

Clearly model.B should not be better than model.A as the only difference is a random variable. However, how do we test the relative effectiveness of the two models. First, we can just calculate AIC values using the function `AIC()`. We can see in the output below that model.A has lower AIC values than model.B.

```
AIC(model.A); AIC(model.B)
## [1] -62.71125
## [1] -66.69933
```

We can also consider the subset model (model.A) as the null hypothesis, the superset model (model.B) as the alternate hypothesis and then use the function `anova()` to check if we can reject the null hypothesis in the favor of the alternate hypothesis.

```
#Null hypothesis first, then alternate hypothesis
anova(model.A, model.B)
## Analysis of Variance Table
##
## Model 1: log(Volume) ~ log(Girth) + log(Height)
## Model 2: log(Volume) ~ log(Girth) + log(Height) + trees$random
##   Res.Df     RSS Df Sum of Sq      F  Pr(>F)
## 1     28 0.18546
## 2     27 0.15289  1  0.032577 5.7532 0.02363 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the output shows, with a p-value of higher than 0.05, we can not reject

the null hypothesis. Thus the analysis shows that model.A is the preferred model.

7.5 Conclusion and Word of Caution

In this chapter, we introduced the idea of linear regression using OLS. We saw how to run linear regression model in R including extracting valuable information from the model. We then looked at various assumptions inherent in OLS and how to test whether those assumptions hold in our model. Finally, we looked at model selection issues including stepwise regression and comparing different models.

Linear Regression is NOT a Silver Bullet

Linear regression is extremely popular. Modern statistical software has made building a linear regression model so easy that we are tempted to use it everywhere. Rarely do regression diagnostics - including testing of linear regression model assumptions, are carried out in a robust and reliable matter. Further, incorrect interpretation of regression model results is endemic.

Linear regression is a specific tool valid only for specific contexts that meet its requirement. The regression analysis is going to be only as useful as the model it tries to estimate. Hence, utmost case must be taken in first designing a theoretically consistent model before the model is evaluated using regression analysis.