Initial Trials with Regression Analysis in R

James D. Wilson, Assistant Professor of Statistics, University of San Franscisco

September 5th, 2017

Components of Regression

We will first *briefly* review the basic components of a regression. The goal of a **regression model** is to quantify the relationship between two variables Y and $\mathbf{X}$ where:

- Y is the **response variable**, or **dependent variable**. This is typically treated as a vector of n observations where entry Y_i is the i^{th} observation.
- $\mathbf{X}$ is the design matrix of **explanatory variables**, or **independent variables**. This is typically an n x p matrix where p = number of explanatory variables. In this way, the i^{th} column of $\mathbf{X}$ represents the n observations of the i^{th} variable.

A **regression model** is a formal means of expressing the relationship between $\mathbf{X}$ and Y. In particular, a regression model postulates that:

1) There is a probability distribution of Y for each level of $\mathbf{X}$
2) The means (expected value) of these probability distributions vary in some systematic fashion with $\mathbf{X}$

Important Note: The existence of a statistical relationship between the response variable Y and the explanatory variable $\mathbf{X}$ does **not** imply that Y depends *causally* on $\mathbf{X}$! There is a large branch of statistics devoted to understanding how to determine causality; however, we will not talk about that here.

Example: The *cars* data set preloaded in R

R comes equipped with a large collection of data sets that are very simple to access. These data serve as a great resource for teaching and simple exploratory analyses. Let's look at what datasets are available.

```
data()
```

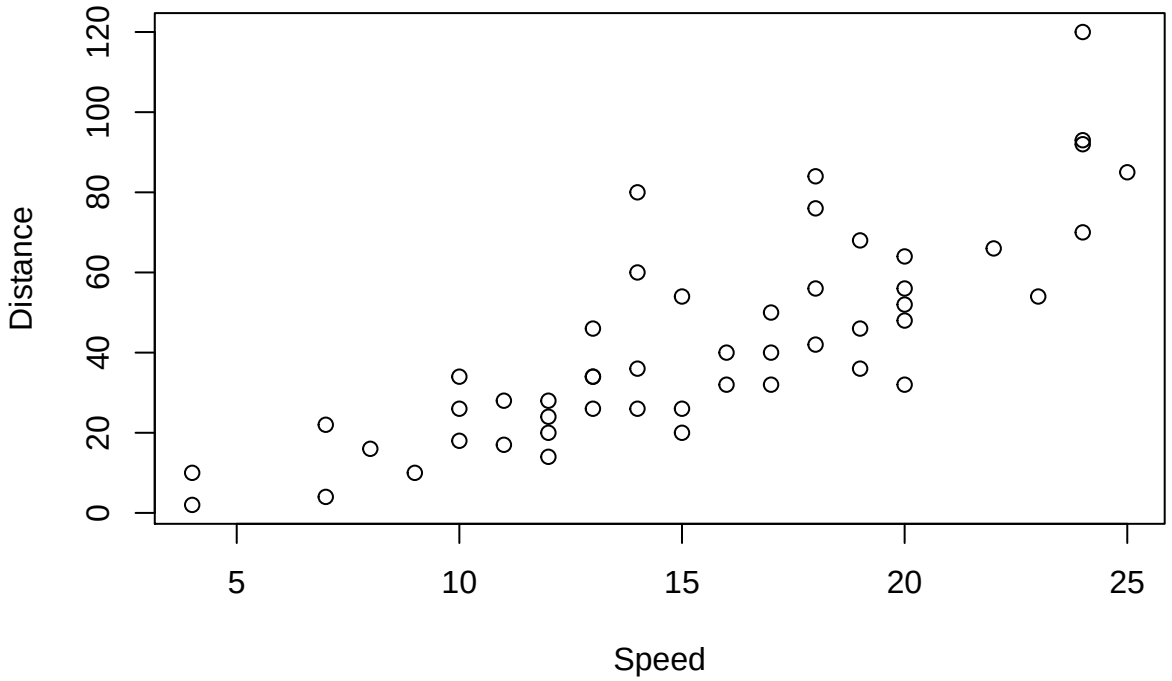
We will look at the *cars* data set. Let's load and look at a summary of this data.

```
data(cars)
?cars
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Now let's look at a scatter plot of *dist* against *speed*.

```
plot(cars$dist ~ cars$speed, xlab = "Speed", ylab = "Distance")
```



From the above plot, it certainly looks like there is a relationship between the variables *dist* and *speed*. But the big question is, what kind of regression model should we fit? We turn now to two popular types of regression: linear regression and logistic regression.

Linear Regression

Given a response variable Y and the design matrix $\mathbf{X}$, the **linear regression model** of Y on $\mathbf{X}$ is given by:

$$Y = \mathbf{X}\beta + \epsilon \qquad (1)$$

Here,

- β is a vector of p parameters (typically unknown)
- ϵ is a vector of *errors*.

If $\mathbf{X}$ only contains one explanatory variable, the linear regression model in (1) reduces to **simple linear regression** as follows:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \qquad (2)$$

Note: We typically include a column of 1s as the first column in a design matrix $\mathbf{X}$ to account for the intercept term β_0.

For statistical inference purposes (i.e. hypothesis testing, prediction, and confidence intervals), we make *one* of the following *two* assumptions on the errors ϵ:

1) $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$
2) $\mathbb{E}[\epsilon_i] = 0$, $Var(\epsilon_i) = \sigma^2$, and $Cov(\epsilon_i, \epsilon_j) = 0$

As commonly done in practice, let's assume that 1) above is true. Then if we would like to fit model (1), we estimate (via a least-squares approach that turns out to be the normal equations from linear algebra) unknown parameters β using $\hat{\beta}$ where:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$

Then our best estimate for Y is given by $\hat{Y} = \mathbf{X}\hat{\beta}$. The estimate $\hat{Y}$ is called the **least-squares** estimate for Y and one can easily draw the least-squares line using these estimates.

Using assumption 1), we are able to then derive a distribution for our estimate $\hat{\beta}$. In particular, if we know the variance of ϵ then

$$\hat{\beta} \sim N(\beta, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}) \qquad (3)$$

Often it is the case that we do **not** know the true variance of ϵ, so we can estimate it by calculating the sample variance of the residuals $e = Y - \hat{Y}$. Supposing that

$$s^2 = \sum_{i=1}^{n}((e_i - \bar{e})^2)/(n-1)$$

Now let $V = (\mathbf{X}^T\mathbf{X})^{-1}$ be the p x p matrix representing the covariance of $\hat{\beta}$ from (3). Then if we estimate σ^2 with s^2, we have that for each β_i,

$$\frac{\hat{\beta}_i - \beta_i}{s * \sqrt{V_{i,i}}} \sim T_{n-1} \qquad (4)$$

Importantly, (4) gives us a way now to *test* whether or not our estimates in our linear regression model are statistically significant. Indeed, for the test

$$H_o : \beta_i = 0 \qquad vs. \qquad H_1 : \beta_i > 0$$

we can now use a p-value under the assumption that $\hat{\beta}$ is truly a centered T_{n-1} random variable. This is in fact exactly what R tells us when we fit a linear regression.

Example: A Simple Linear Regression for the cars data

The function *lm()* is the primary function for use here. (lm = linear model). Let's first look at an overview of the *lm()* function.

```
?lm
```

Now we will fit a linear regression model of the variable *dist* on *speed*. Then we will analyze the fit using the summary function.

```
reg1 <- lm(dist ~ speed, data = cars)
names(reg1)
```

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

```
summary(reg1)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

The above summary includes a lot of important information for the fit regression model. For now, let's focus on the *Coefficients* output. The table here reports 4 outcomes for each parameter in the model:

- **Estimate**: the least squares estimate of the parameter
- **Std. Error**: the standard error of the estimate. From equation (4), the standard error of $\hat{\beta}_i$ is $s * \sqrt{V_{i,i}})$.
- **t-value**: the observed t-statistic as calculated using equation (4) when assuming the true $\beta_i = 0$
- **Pr(**$> |t|$**)**: the two sided p-value associated with the hypothesis test of whether or not $\beta_i = 0$

According to our output, both the intercept and the parameter associated with *speed* are statistically significant at a 0.05 level. So, if we let $Y_i = i$th observation of *dist*, and let $x_i = i$th observation of *speed*. Our least squares regression model is

$$\hat{Y}_i = -17.5791 + 3.9324 * x_i$$

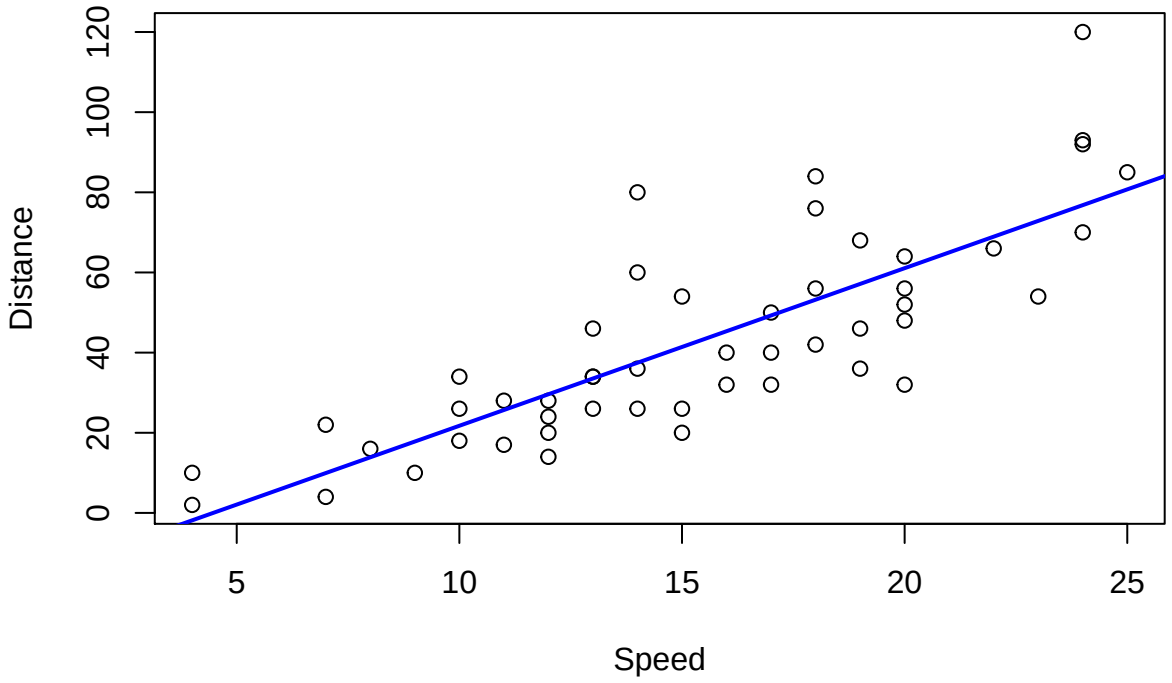
Another important value given in the above output is the **Multiple R-squared**. This is defined as follows:

- **Multiple R-squared**: a value between 0 and 1 that represents the proportion of variation in Y that is explained by the covariates in the linear model.

In general, we want these values to be as close to 1 as possible where values equal to 1 indicate that the covariates perfectly explain the variance of Y. In this case, Y would share an exact linear relationship with the covariates.

Let's now look at the regression line on the original scatterplot.

```
plot(cars$dist ~ cars$speed, xlab = "Speed", ylab = "Distance")
abline(reg1, col = "blue", lwd = 2)
```



Testing the Normal Assumptions of a Linear Regression

An important step in fitting a linear regression model is to ensure that a linear model explains the relationship between the two variables, and more importantly, if the normality assumptions on the errors are met.

Basic exploratory analysis and correlation coefficients can be used to evaluate the "linearity" of a fit. To test the normality assumptions, we must rely on the residuals of the fit $e = \hat{Y} - Y$. Under assumption 1), the residuals *should* be normally distributed. It follows that the standardized residuals, r, have a standard normal distribution. That is,

$$r = \frac{e - \mathbb{E}[e]}{\sigma_e} \sim N(0, 1)$$

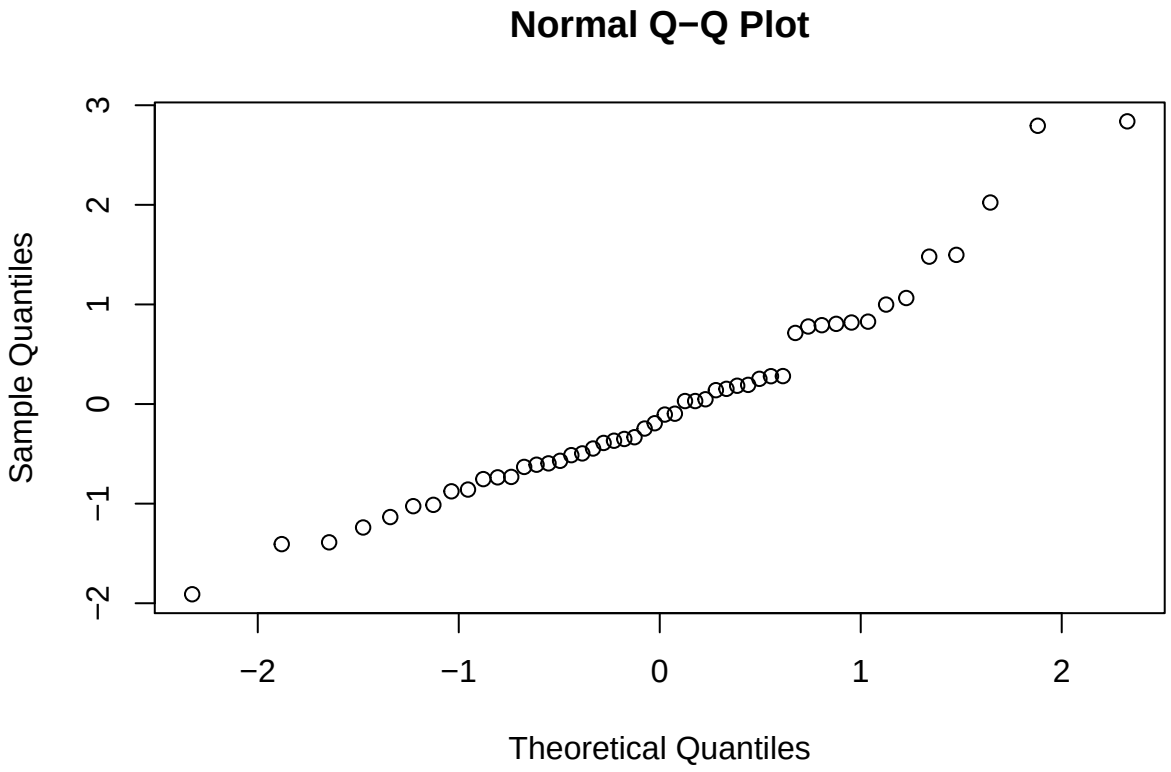
Of course, we don't know the true mean or variance of e we estimate them and rely on the central limit theorem to give us normality. So checking our normality assumptions comes down to the following test:

$$H_o : r \sim N(0, 1) \qquad vs. \qquad H_1 : r \nsim N(0, 1)$$

From earlier discussion, we know that we can use the Kolmogorov-Smirnov test of normality to test this. Let's look at the residuals from the *cars* data.

```
# calculate the standardized residuals
e <- reg1$residuals
std.residuals <- (e - mean(e)) / sd(e)

# plot the qqplot of the residuals
qqnorm(std.residuals)
```



Now let's use the Kolmogorov-Smirnov test to test for normality of the residuals.

```
ks.test(std.residuals, rnorm(1000))
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  std.residuals and rnorm(1000)
## D = 0.153, p-value = 0.2149
## alternative hypothesis: two-sided
```

The normality assumptions for the *cars* data are satisfied as we fail to reject the null hypothesis that the residuals are normally distributed.

Confidence Intervals and Prediction using the *predict()* function in R

Once we have fit a linear regression model, we'd often next like to predict future observations given new data, or provide confidence intervals on estimates calculated from functions of our estimated parameters. The *predict()* function in R can handle each of these situations. Let's first look at the *predict()* function.

```
?predict.lm
```

Note that I looked up *predict.lm* rather than *predict*. This is because *predict()* is a base function in R that will have different interpretations depending upon the class of the input object. We can check the class of our variable *reg1* by typing:

```
class(reg1)
```

```
## [1] "lm"
```

1) **Predictions for new data**

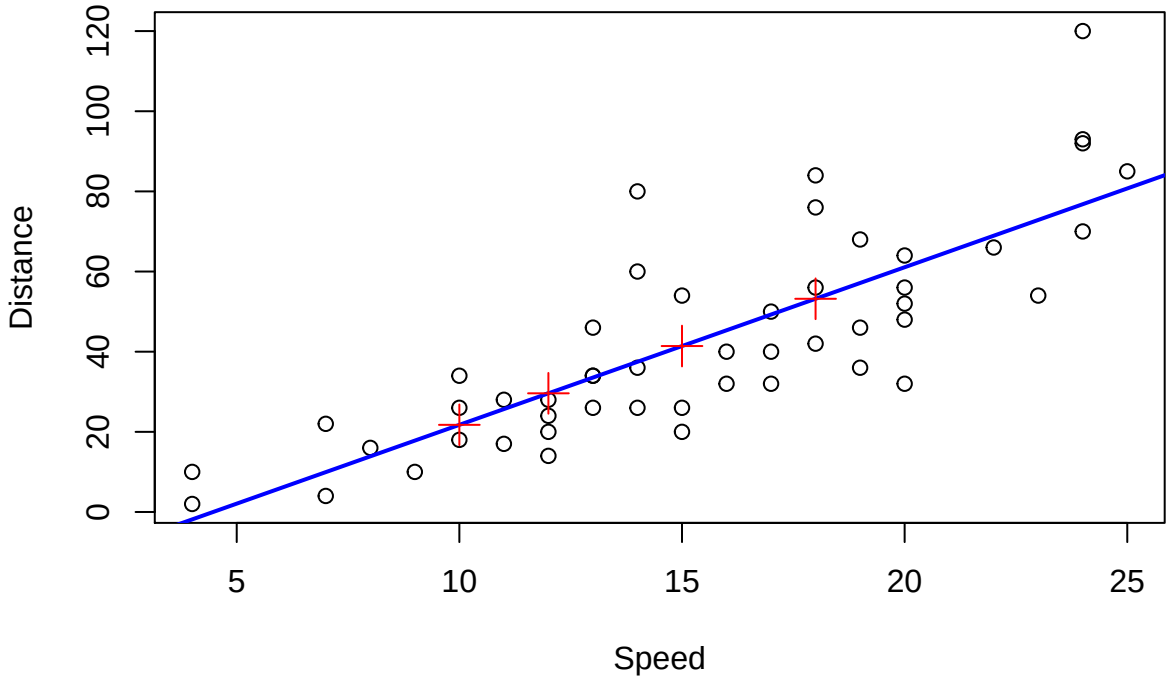
```r
#create new speed data where all values are within our original data
new.speeds <- c(10, 12, 15, 18)

#predict confidence intervals for each of the data points
predictions <- predict(reg1, newdata = data.frame(speed = new.speeds), type = "response",
                       interval = 'none', level = 0.95)

#Look at the new values
predictions
```

```
##        1        2        3        4
## 21.74499 29.60981 41.40704 53.20426
```

```r
#Plot them on the original plot
plot(cars$dist ~ cars$speed, xlab = "Speed", ylab = "Distance")
abline(reg1, col = "blue", lwd = 2)
points(x = new.speeds, y = predictions, col = "red", pty = 5, pch = 3, cex = 2)
```



2) **Confidence Intervals for observed data**

```r
#create new speed data where all values are within our original data
new.speed <- seq(min(cars$speed), max(cars$speed), length.out=100)

#predict confidence intervals for each of the data points
predictions <- predict(reg1, newdata = data.frame(speed = new.speed),
                       interval = 'confidence', level = 0.95)

#look at the first 6 predictions
head(predictions)
```

```
##          fit        lwr       upr
## 1 -1.8494599 -12.329543  8.630624
## 2 -1.0153125 -11.334611  9.303986
## 3 -0.1811652 -10.340226  9.977896
```

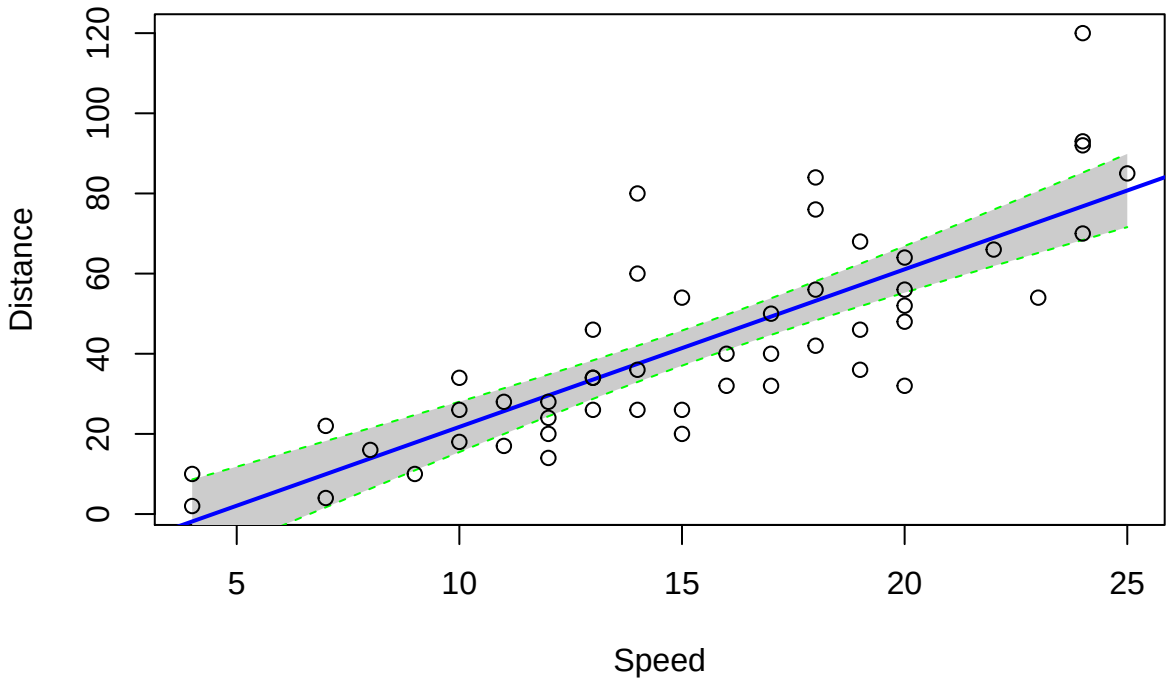
```
## 4  0.6529821  -9.346414 10.652378
## 5  1.4871294  -8.353203 11.327462
## 6  2.3212767  -7.360622 12.003175
```

Let's now plot confidence intervals for each point that we interpolated.

```
# plot the original scatterplot with least squares line
plot(cars$dist ~ cars$speed, xlab = "Speed", ylab = "Distance", type = "n")

# add fill for the confidence regions
polygon(c(rev(new.speed), new.speed), c(rev(predictions[ , 3]), predictions[ , 2]),
        col = 'grey80', border = NA)

# Add lines for the confidence intervals
lines(new.speed, predictions[ , 3], lty = 'dashed', col = 'green')
lines(new.speed, predictions[ , 2], lty = 'dashed', col = 'green')
abline(reg1, col = "blue", lwd = 2)
points(cars$dist ~ cars$speed)
```



Case Study: Multiple Regression with the iris data

We are now ready to try a more complicated example. Below I include a chunk of code that can be used to investigate the relationship between the sepal length of flowers and their species, sepal width, petal length, and petal width. First, let's load the data and summarize each variable.

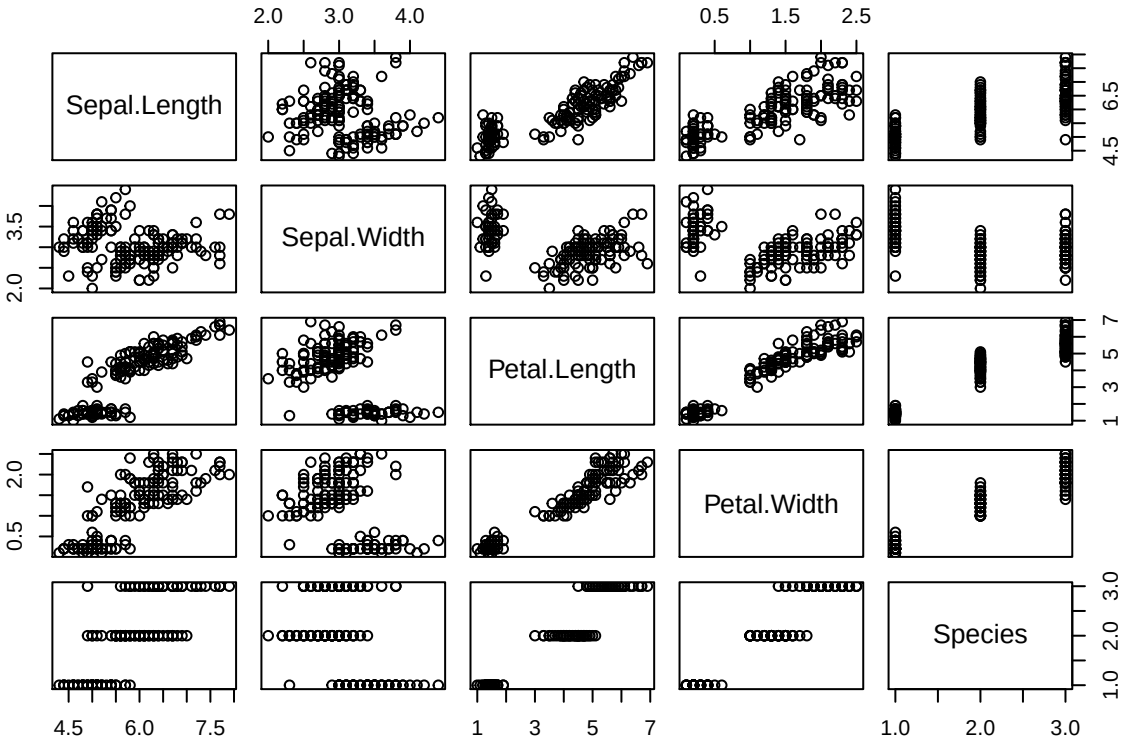
```
data(iris)
?iris
summary(iris)
```

```
##    Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
```

```
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Now, we can look at pairwise scatterplots of these variables using the *pairs()* function.

```
pairs(iris)
```



Let's now run a linear regression of *sepal.length* on the remaining variables, look at the output, and then check the normal assumptions on this regression. This is all done in the following code.

```
#run a linear regression model
reg.iris <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species, data = iris)
summary(reg.iris)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width +
##     Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79424 -0.21874  0.00899  0.20255  0.73103
##
```

```
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.17127    0.27979   7.760 1.43e-12 ***
## Sepal.Width         0.49589    0.08607   5.761 4.87e-08 ***
## Petal.Length        0.82924    0.06853  12.101  < 2e-16 ***
## Petal.Width        -0.31516    0.15120  -2.084  0.03889 *
## Speciesversicolor -0.72356    0.24017  -3.013  0.00306 **
## Speciesvirginica  -1.02350    0.33373  -3.067  0.00258 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3068 on 144 degrees of freedom
## Multiple R-squared:  0.8673, Adjusted R-squared:  0.8627
## F-statistic: 188.3 on 5 and 144 DF,  p-value: < 2.2e-16
```

Now let's define the variables used in this regression in the following way:

- $Y = Sepal.Length$
- $x_1 = Sepal.Width$
- $x_2 = Petal.Length$
- $x_3 = Petal.Width$
- $x_4 = I(Species = versicolor)$
- $x_5 = I(Species = virginica)$

Based on the above results, it appears that each of the parameter estimates are statistically significant at the 0.05 level. So, our linear regression model is given by:

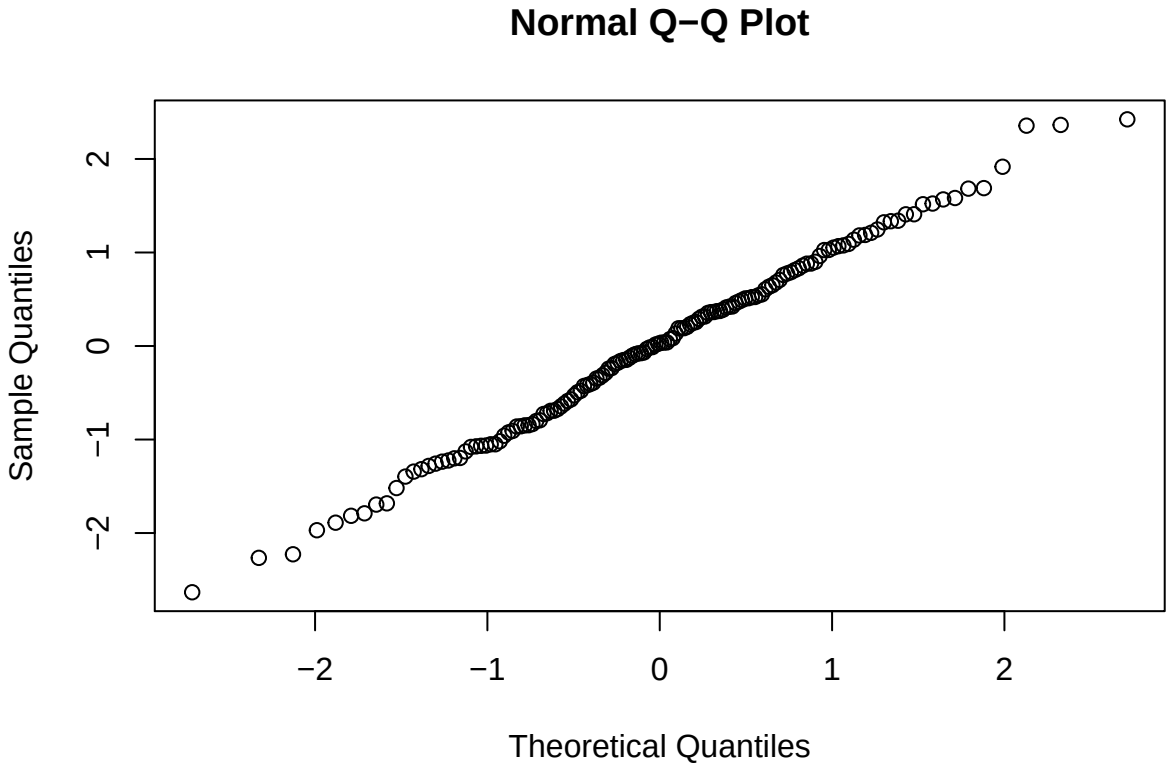
$$\hat{Y}_i = 2.17127 + 0.49589x_{1,i} + 0.82924x_{2,i} - 0.31516x_{3,i} - 0.72356x_{4,i} - 1.02350x_{5,i}$$

Note on Interpretation: Here the variables x_4 and x_5 are indicator variables describing the species type. In situations where a variable is categorical (like *Species* – having only 3 possibilities), parameters associated with indicator variables describe the change in the response when the variable takes a certain value *relative* to a *base* category. In this case, the *base* category is *Species* = 3. See the scatterplot to understand why this is the case.

Now that we understand our model, let's check the assumption that the residuals of the fit are approximately normal. We will first plot a qqplot and then run the Kolmogorov-Smirnov test of normality.

```
# calculate the standardized residuals
e <- reg.iris$residuals
std.residuals <- (e - mean(e)) / sd(e)

# plot the qqplot of the residuals
qqnorm(std.residuals)
```



Now let's use the Kolmogorov-Smirnov test to test for normality of the residuals.

```
ks.test(std.residuals, rnorm(1000))
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  std.residuals and rnorm(1000)
## D = 0.046, p-value = 0.9454
## alternative hypothesis: two-sided
```

In the Kolmogorov-Smirnov test, we fail to reject the null hypothesis that the residuals are normally distributed. Thus, we can be satisfied that our assumptions are met.

We can now use our regression model to predict *Sepal.Length* given new data, and the confidence intervals / prediction intervals can be once again constructed based on the Normality assumption that we made in our model.

Concluding Remarks

This lesson still only scratched the surface for the basics in Linear Regression. Other aspects of regression include:

- Model selection
- Outlier detection
- Tests of heteroscedacity (non-constant variance)
- Transformations on the response and variables to fit the normality / linearity assumptions

Also, there are many other types of regression which will may be discussed in future workshops such as:

- Logistic regression: regression for binary data
- Poisson regression: regression for count data

- Generalized linear models: linear models for exponential random families