Regression

Linear Regression

We do linear regression using the lm function. For the function, you need your regression equation, written like dependent_variable ~ independent_variable_1 + independent_variable_2 + independent_variable_3.

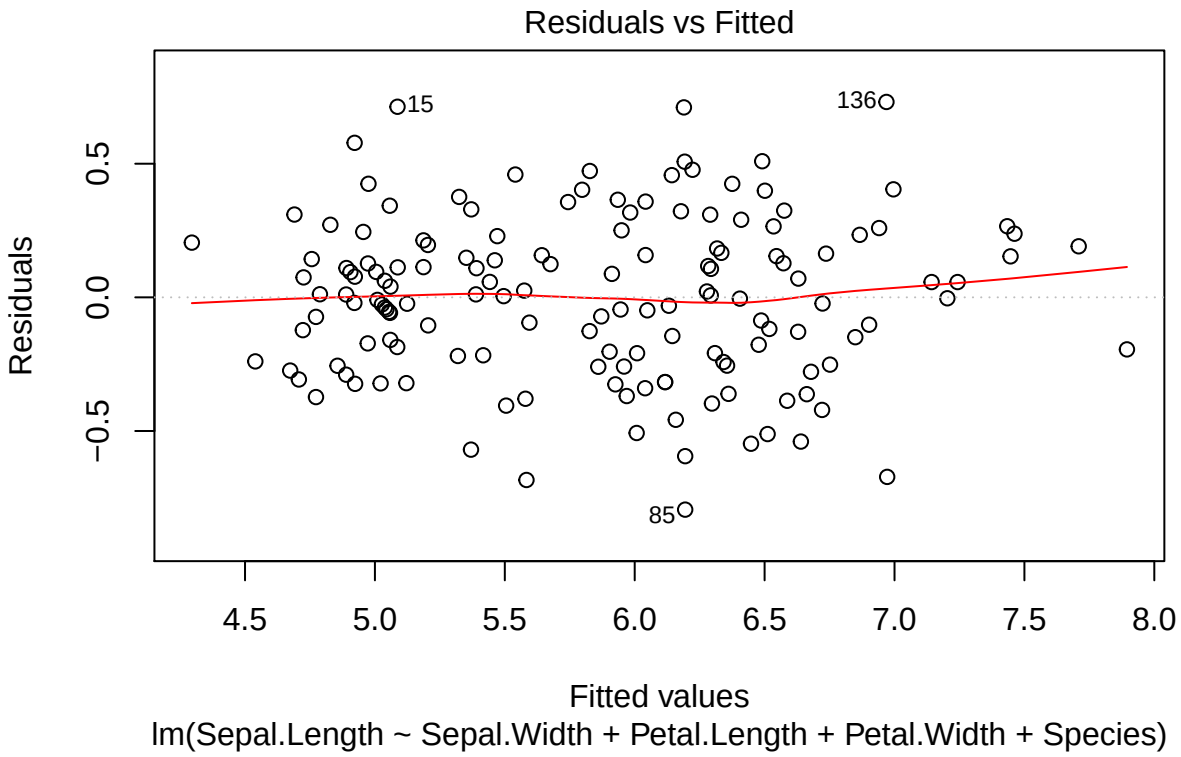
As usual, we create an object for an example. For this example, we use the iris dataset. The summary function gives an overview of all the attributes of the model.

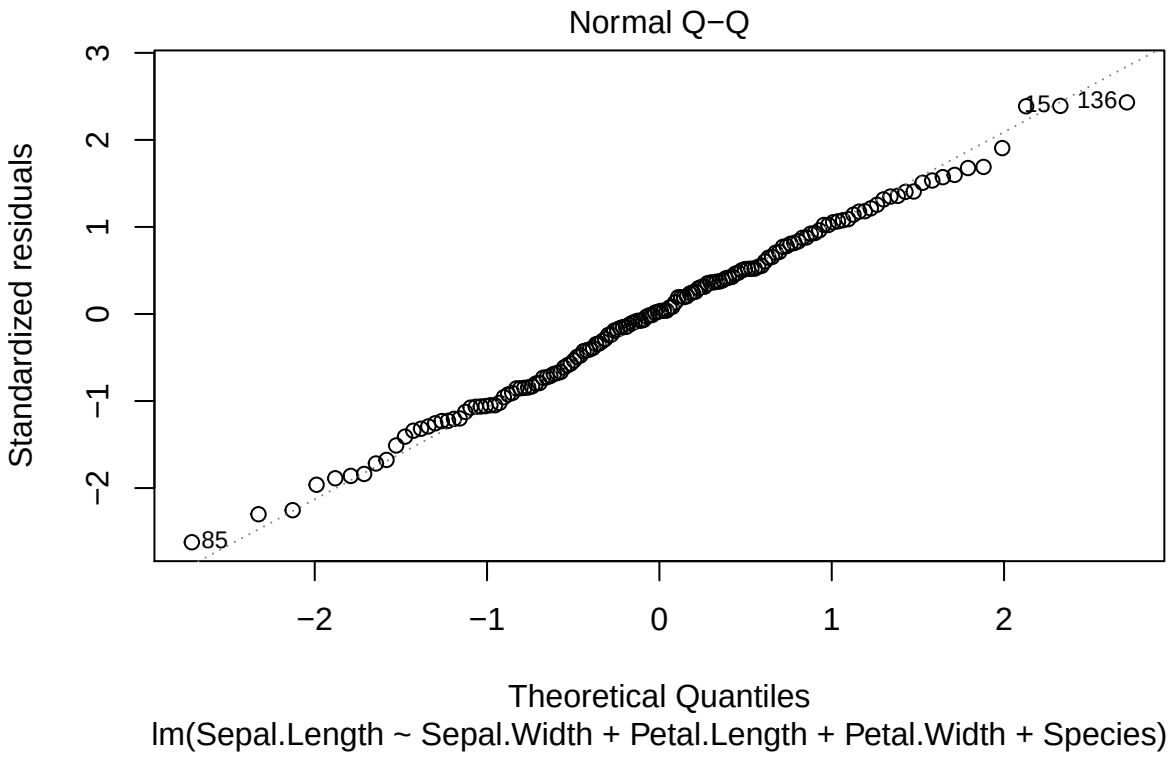
```
linear_reg <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species, data=iris)


summary(linear_reg)
```

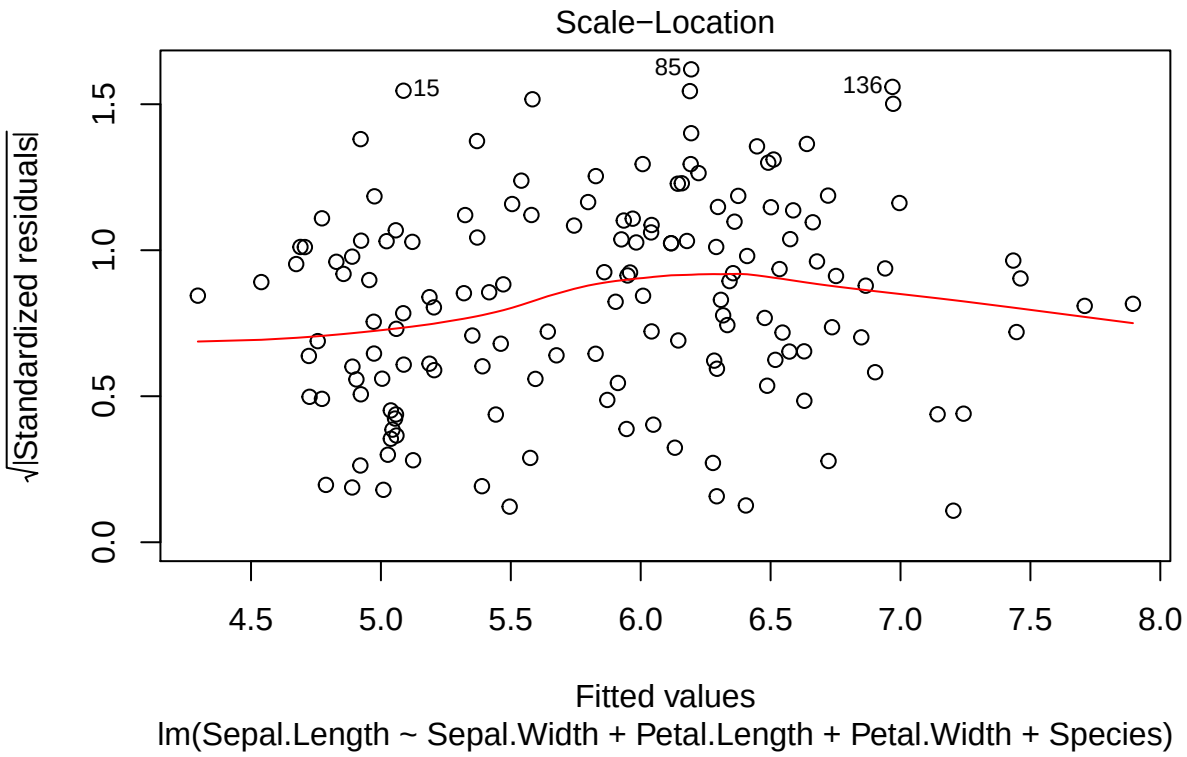
```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width +
##     Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79424 -0.21874  0.00899  0.20255  0.73103
##
## Coefficients:
##                   Estimate Std. Error t value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

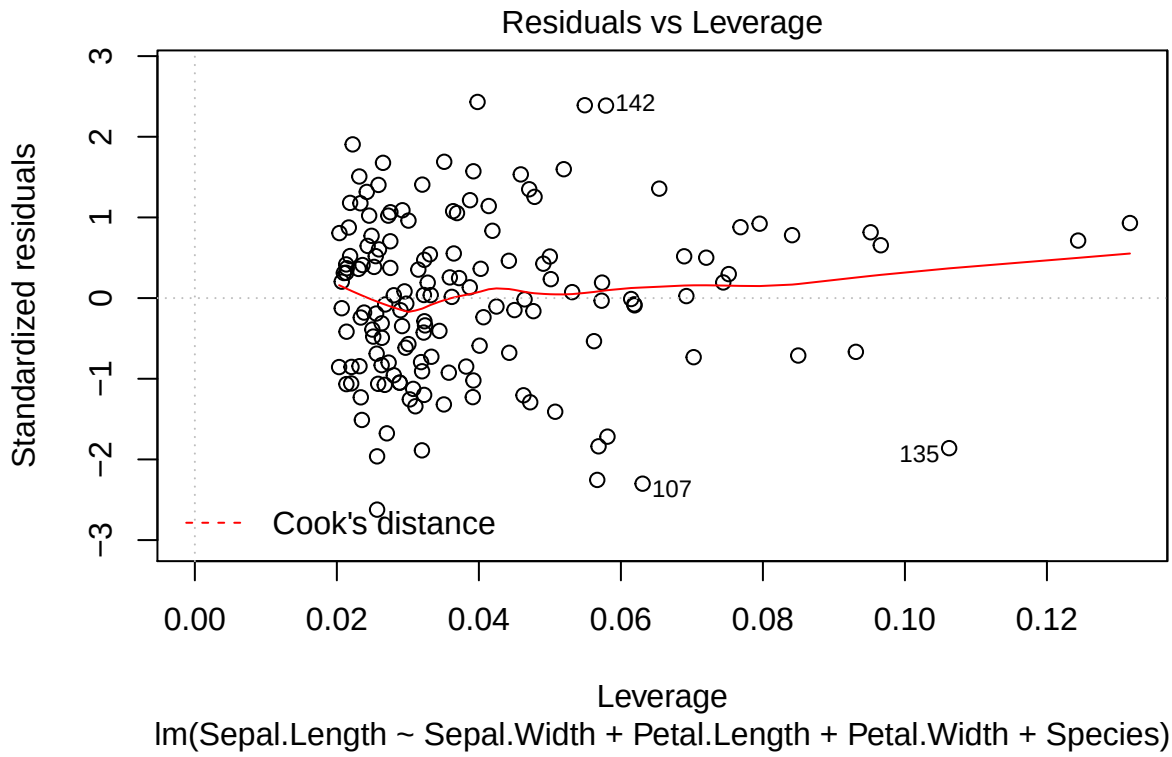
Using plot() on the object creates a number of diagnostic scatterplots

```
plot(linear_reg)
```


Residuals vs Fitted
15
136
85
Residuals
0.5
0.0
−0.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Fitted values
lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
85
15
136
−2
−1
0
1
2
Theoretical Quantiles
lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
15
85
136
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Fitted values
lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species)



A correlation matrix of the variables in our LM:

```
indep <- iris[,-1]
indep <- indep[,-4] #remove the last column - species. It is categorical soo can't compute a correlatio
names(indep)
```

```
## [1] "Sepal.Width"  "Petal.Length" "Petal.Width"
```

```
#simply use the cor function on the matrix of variables:
cor(indep)
```

```
##              Sepal.Width Petal.Length Petal.Width
## Sepal.Width    1.0000000   -0.4284401  -0.3661259
## Petal.Length  -0.4284401    1.0000000   0.9628654
## Petal.Width   -0.3661259    0.9628654   1.0000000
```

We see Petal Width and Length are very strongly correlated, but let's proceed. Might also want to know the variance inflation factor(s):

```
library(car)
vif(linear_reg)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## Sepal.Width   2.227466  1        1.492470
## Petal.Length 23.161648  1        4.812655
## Petal.Width  21.021401  1        4.584910
## Species      40.039177  2        2.515482
```

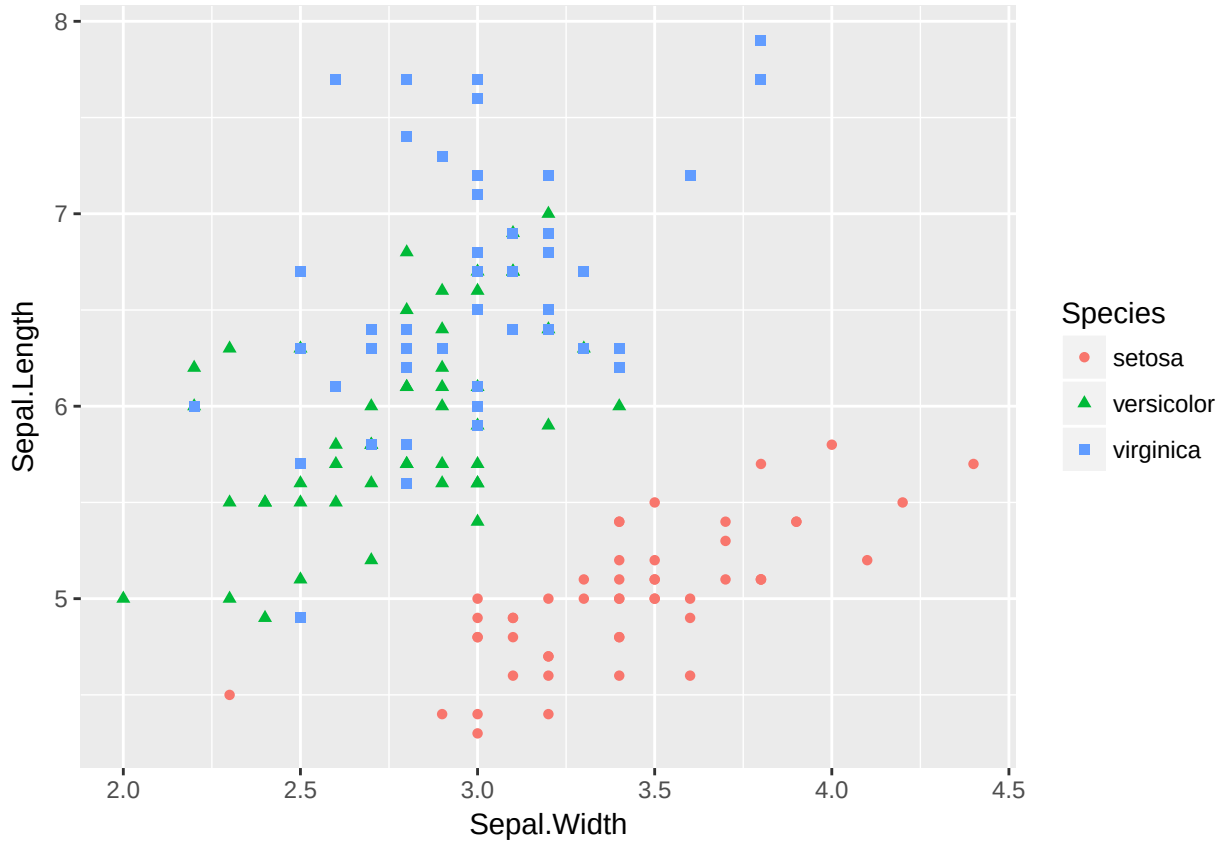
Look at what types of flowers we got in the iris dataset.

```
summary(iris$Species)
```

```
##     setosa versicolor  virginica
##         50         50         50
```

Now plot sepal length against sepal width, but color the species. We are going to use a different package than before, ggplot2. We use the function qplot() here - it is a shortcut for quickly making graphs with the package.

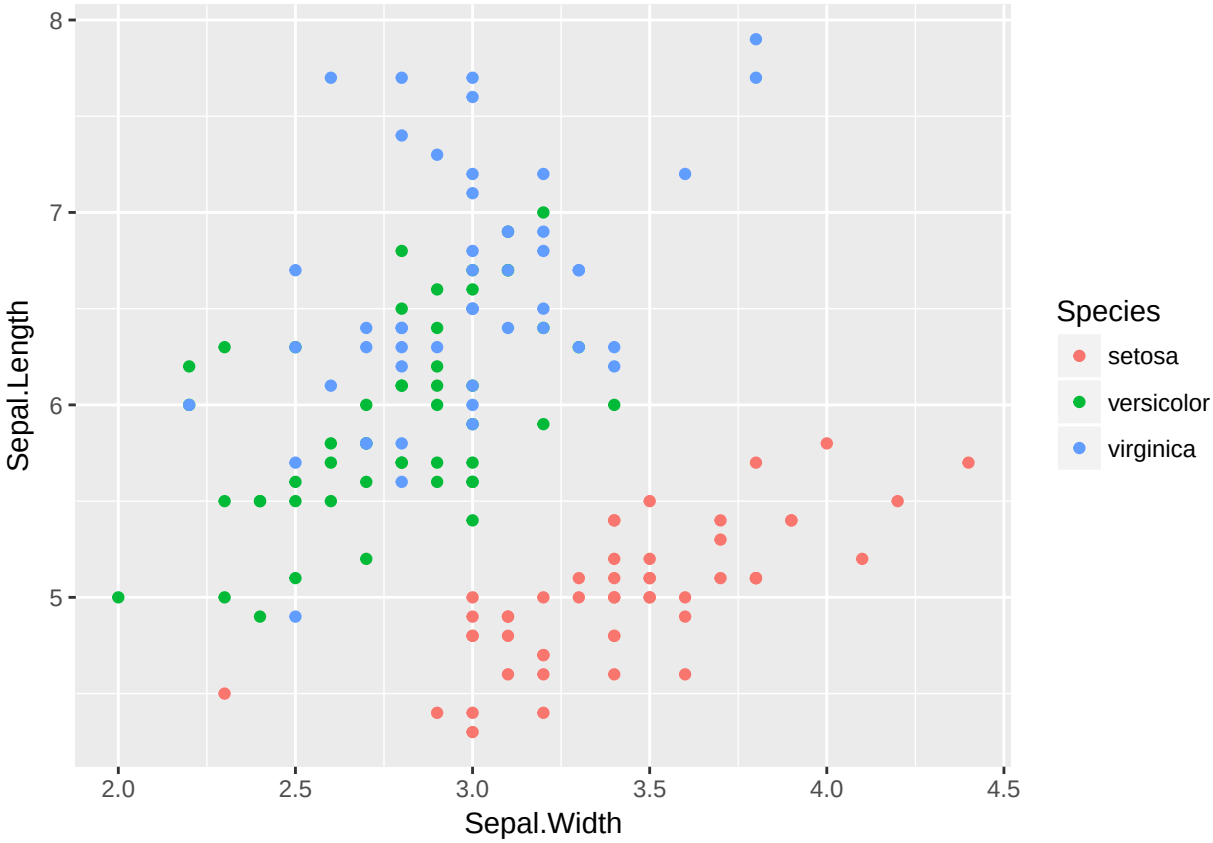
```
#don't forget to load the library (install the package first!)
library(ggplot2)

qplot(x=Sepal.Width, y=Sepal.Length,
      data=iris,
      colour=Species,
      shape=Species)
```



Function reference: http://ggplot2.tidyverse.org/reference/qplot.html

Plotting it without the quickplot function - using the ggplot package 'directly':

```
library(ggplot2)
ggplot(
  data=iris,
  aes(x=Sepal.Width, y=Sepal.Length)) +
  #aesthetics defines your graph field (axes).
  geom_point(aes(color=Species))
```




```
#use pluses to add stuff (try running this function without geom_point part). We add geom_point - the
```

Function reference: http://ggplot2.tidyverse.org/reference/ggplot.html

For this example, there's no difference, but the first function can't do more complex plots.

Let's plot our prediction results.

```
ggplot() +
  #first we make an empty plot(above), then add our first set of points like before
geom_point(data=iris,
           aes(x=Sepal.Width, y=Sepal.Length),
           colour='blue') +
  #now we add another set of points. for y-coordinates, we use the predict function. It predicts values
geom_point(data=iris,
           aes(x=Sepal.Width, y=predict(linear_reg)),
           colour='red')
```

Dummy Variables and Plotting regressions

So far we have been working with a multiple regression. We might want to be able to plot simple linear regression lines though. First, create a new regression object.

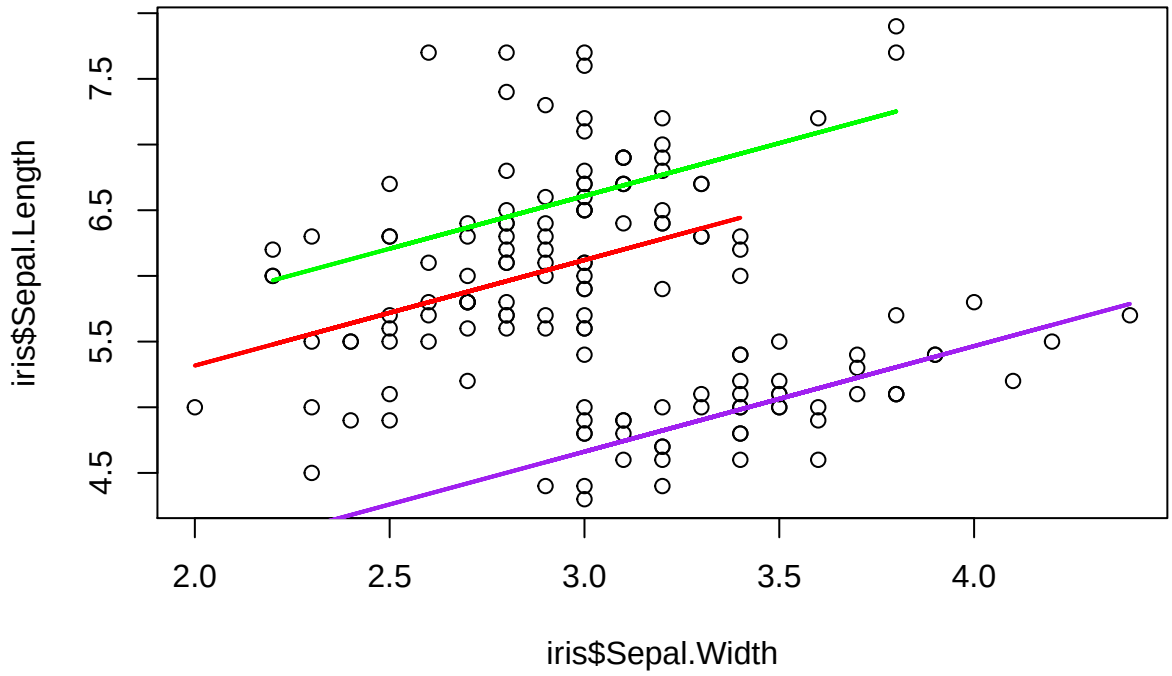
```
simple_linear_reg <- lm(formula=iris$Sepal.Length~iris$Sepal.Width+iris$Species)
summary(simple_linear_reg)
```

```
##
## Call:
## lm(formula = iris$Sepal.Length ~ iris$Sepal.Width + iris$Species)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30711 -0.25713 -0.05325  0.19542  1.41253
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               2.2514     0.3698   6.089 9.57e-09 ***
## iris$Sepal.Width          0.8036     0.1063   7.557 4.19e-12 ***
## iris$Speciesversicolor    1.4587     0.1121  13.012  < 2e-16 ***
## iris$Speciesvirginica     1.9468     0.1000  19.465  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.438 on 146 degrees of freedom
## Multiple R-squared:  0.7259, Adjusted R-squared:  0.7203
```

```
## F-statistic: 128.9 on 3 and 146 DF,  p-value: < 2.2e-16
```

We have a dummy variable in this model - species. A plot of the individual regression lines for each species can show the effect of the dummy variable. Two ways to do it are shown here: one with pre-built graphics of R, second with ggplot2.

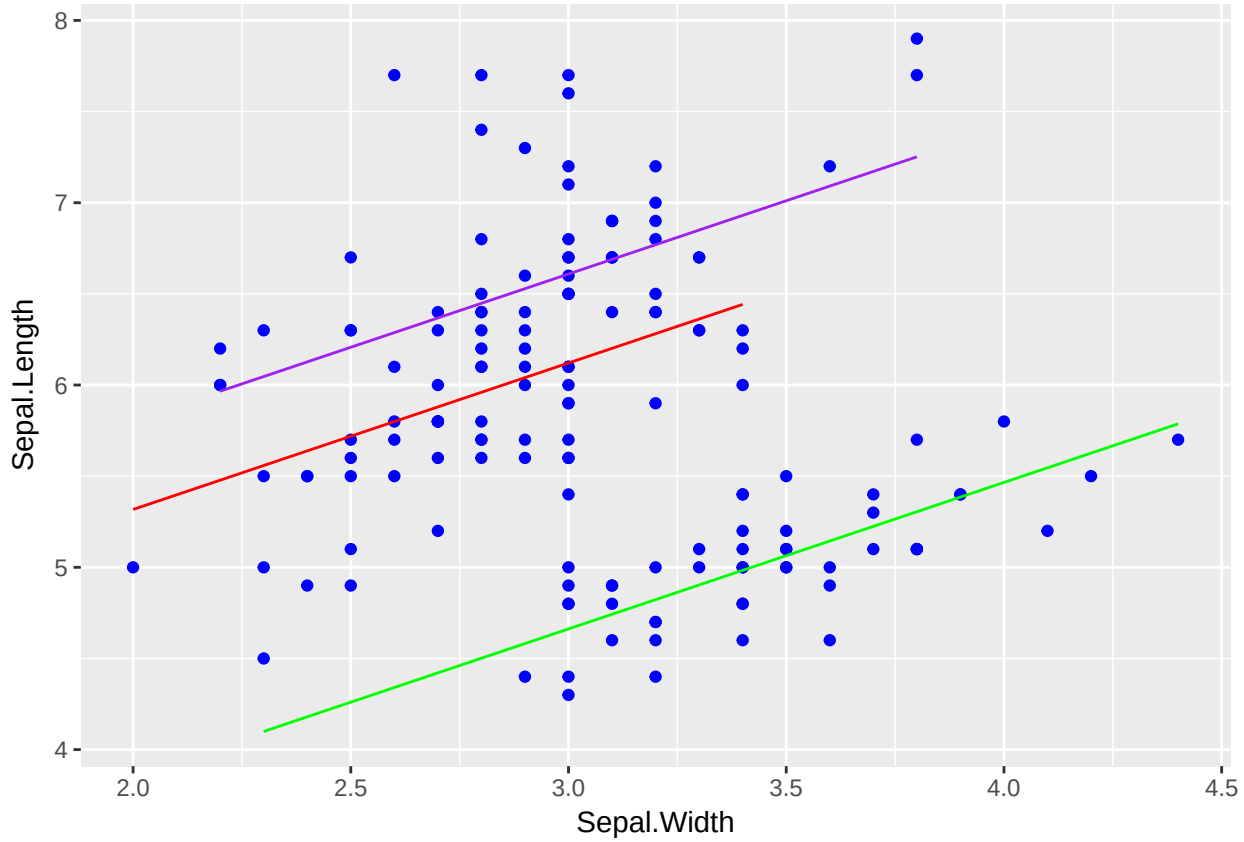
```
#using normal plot
plot(iris$Sepal.Length~iris$Sepal.Width)
lines(iris$Sepal.Width[iris$Species=="versicolor"],fitted(simple_linear_reg)[iris$Species=="versicolor"]
lines(iris$Sepal.Width[iris$Species=="virginica"],fitted(simple_linear_reg)[iris$Species=="virginica"],
lines(iris$Sepal.Width[iris$Species=="setosa"],fitted(simple_linear_reg)[iris$Species=="setosa"],lwd=2,
```



source: http://geog.uoregon.edu/GeogR/topics/dummy.html

```
#using ggplot2
ggplot() +
 geom_point(data=iris,
           aes(x=Sepal.Width, y=Sepal.Length),
           colour='blue') +

geom_line(data=iris[iris$Species=="versicolor",],
           aes(x=Sepal.Width, y=fitted(simple_linear_reg)[iris$Species=="versicolor"]),
           colour='red')+

  geom_line(data=iris[iris$Species=="virginica",],
           aes(x=Sepal.Width, y=fitted(simple_linear_reg)[iris$Species=="virginica"]),
           colour='purple')+

    geom_line(data=iris[iris$Species=="setosa",],
```

```
          aes(x=Sepal.Width, y=fitted(simple_linear_reg)[iris$Species=="setosa"]),
          colour='green')
```




```
#also note this ggplot version works, but isn't tidy.
```

model selection

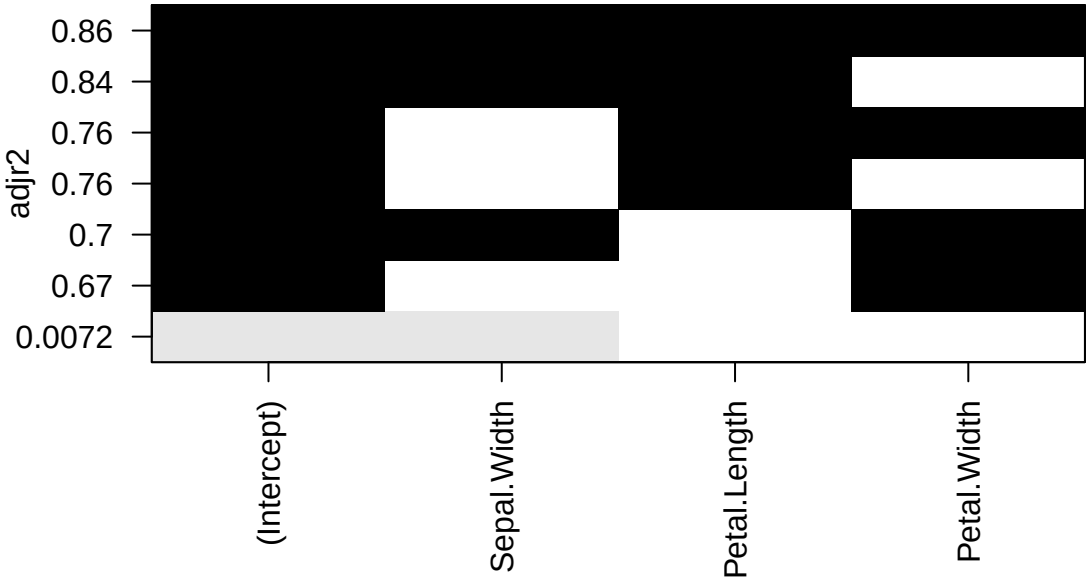
For automatic model selection, the leaps package can be used. There are other methods, but this one generates nice visual output. So first install the leaps package (preferrably from the interface.)

```
library(leaps)
aleaps<- regsubsets(data=iris,
                    Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,nbest=3)
summary(aleaps)
```

```
## Subset selection object
## Call: regsubsets.formula(data = iris, Sepal.Length ~ Sepal.Width +
##     Petal.Length + Petal.Width, nbest = 3)
## 3 Variables  (and intercept)
##              Forced in Forced out
## Sepal.Width      FALSE      FALSE
## Petal.Length     FALSE      FALSE
## Petal.Width      FALSE      FALSE
## 3 subsets of each size up to 3
## Selection Algorithm: exhaustive
##          Sepal.Width Petal.Length Petal.Width
## 1  ( 1 ) " "         "*"          " "
```

```
## 1  ( 2 ) " "         " "          "*"
## 1  ( 3 ) "*"         " "          " "
## 2  ( 1 ) "*"         "*"          " "
## 2  ( 2 ) " "         "*"          "*"
## 2  ( 3 ) "*"         " "          "*"
## 3  ( 1 ) "*"         "*"          "*"
```

```
plot(aleaps, scale="adjr2")
```




```
# by using the scale argument, select which statistic to use - r^2, adjr^2, BIC or AIC.
#note: this might return an error if your plot area isn't large enough. To fix, just maximize RStudio.
```

Time Series Regression

Regressing time series against it's lags:

```
#let's return to our unemployment dataset from before. don't forget that we need the seasonal package f
library(seasonal)
sample_dataset <- unemp
delayed_regression <- lm(sample_dataset ~ lag(sample_dataset, k=10), data=sample_dataset)

summary(delayed_regression)
```

```
## Warning in summary.lm(delayed_regression): essentially perfect fit: summary
## may be unreliable
```

```
##
## Call:
## lm(formula = sample_dataset ~ lag(sample_dataset, k = 10), data = sample_dataset)
```

```
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.800e-12 -1.425e-13 -9.120e-14 -1.580e-14  2.612e-11
##
## Coefficients:
##                              Estimate Std. Error   t value Pr(>|t|)
## (Intercept)                 3.239e-12  2.949e-13 1.098e+01   <2e-16 ***
## lag(sample_dataset, k = 10) 1.000e+00  3.231e-17 3.095e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.474e-12 on 321 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.579e+32 on 1 and 321 DF,  p-value: < 2.2e-16
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(car)

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

```
##
##  Durbin-Watson test
##
## data:  delayed_regression
## DW = 2.1361, p-value = 0.8794
## alternative hypothesis: true autocorrelation is greater than 0
```

ARIMA modeling

In the last section of regressions we will cover ARIMA modeling. ARIMA stands for Auto Regressive Integrated Moving Average. It therefore integrates regression concepts with moving average methods of predictions. The kewl part of the forecast package and the Arima function, is that it completely optimizes itself. The creation of the model therefore is really simple and you can call the results with the summary function as shown below. The forecast can also be mapped with the forecast function.

```
library(forecast)

arima_model <- auto.arima(sample_dataset)

summary(arima_model)
```

```
## Series: sample_dataset
## ARIMA(2,1,2)(1,0,0)[12]
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1
```

```
##       1.6492  -0.7938  -1.7043  0.9110  0.8532
## s.e.  0.0567   0.0593   0.0381  0.0488  0.0289
##
## sigma^2 estimated as 87294:  log likelihood=-2293.49
## AIC=4598.99   AICc=4599.26   BIC=4621.64
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -1.396359 292.6984 228.389 0.01202556 2.669917 0.2194074
##                     ACF1
## Training set 0.04555886
```

```r
library(forecast)
library(ggplot2)

arima_model <- auto.arima(sample_dataset)
plot(forecast(arima_model, h=100))
```