Part I: Exploratory Data Analysis, Linear Regression, ANOVA

Load Packages

First, we must load the packages that will be used in the first part of this workshop.

```
library(pastecs, quietly = TRUE)
library(lm.beta,  quietly = TRUE)
library(lmtest,  quietly = TRUE)
library(foreign,  quietly = TRUE)
library(lattice,  quietly = TRUE)
library(lme4,  quietly = TRUE)
library(nlme,  quietly = TRUE)
library(survival,  quietly = TRUE)
library(dplyr,  quietly = TRUE)
library(ggfortify,  quietly = TRUE)
library(survminer,  quietly = TRUE)
library(rms,  quietly = TRUE)
```

Exploratory Data Analysis

Basic Statistical Analysis

Data set description

In this section, we will be using the iris data set. This data set contains measurement data of the flower of certain plant species. The data set has five variables:

- *Sepal.Length* - measurements of Sepal length
- *Sepal.Width* - measurements of Sepal width
- *Petal.Length* - measurements of Petal length
- *Petal.Width* - measurements of Petal width
- *Species* - species of the plant

```
data = iris
```

Basic Statistical Analysis

Univariate descriptive statistics

The function, `stat.desc`, can be used to do a statistical analysis of data. It returns the mean, median, maximum, minimum, etc of a data set.

```r
stat.desc(data$Sepal.Width)
```

```
##      nbr.val     nbr.null       nbr.na          min          max
## 150.00000000   0.00000000   0.00000000   2.00000000   4.40000000
##        range          sum       median         mean      SE.mean
##   2.40000000 458.60000000   3.00000000   3.05733333   0.03558833
## CI.mean.0.95          var      std.dev     coef.var
##   0.07032302   0.18997942   0.43586628   0.14256420
```

Basic Statistical Analysis

Descriptive statistics by groups

Using `tapply`, we compute the same descriptive statistics above but grouped species with the same sepal width. `tapply` takes

- *first argument*: the input data to which we will apply the statistical function
- *second argument*: the grouping data which tells the statistical function how to group the input data
- *third argument*: the statistical function.

We will be considering the statistical functions: `mean`,`sd` and `length`.

Basic Statistical Analysis

Descriptive statistics by groups

We will be considering the statistical functions: `mean`,`sd` and `length`.

```
mean <- tapply(data$Sepal.Length, data$Sepal.Width, mean)
standard_deviation <- tapply(data$Sepal.Length, data$Sepal.Width, sd)
number_of_observations <- tapply(data$Sepal.Length, data$Sepal.Width, length)
head(round(cbind(mean, standard_deviation, number_of_observations), digits = 6))
```

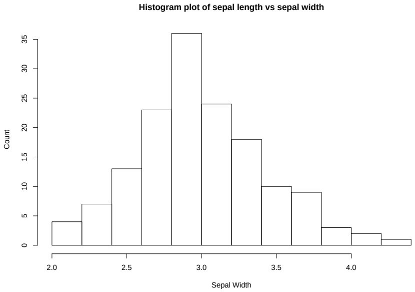
```
##         mean standard_deviation number_of_observations
## 2   5.000000                 NA                      1
## 2.2 6.066667           0.115470                      3
## 2.3 5.325000           0.767572                      4
## 2.4 5.300000           0.346410                      3
## 2.5 5.762500           0.625500                      8
## 2.6 6.160000           0.887694                      5
```

Basic Data Visualization

Histogram

It is important get an idea of the “structure” of your data. To do this, we can use histograms. We use the `hist` function to plot a distribution of sepal width.

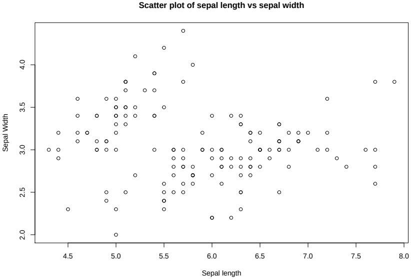
```
hist(data$Sepal.Width,
     xlab='Sepal Width',ylab='Count',
     main= 'Histogram plot of sepal length vs sepal width')
```



Scatter Plot

We use the `plot` function to create a scatter plot of sepal length vs sepal width.

```
plot(data$Sepal.Length, data$Sepal.Width,
     xlab='Sepal length', ylab='Sepal Width',
     main= 'Scatter plot of sepal length vs sepal width')
```



Basic Data Visualization

Stacking plots (without ggplot2)

We also differentiate the scatter plot above by species. We plot each species separately.

First, we call `plot` to create a canvas with an set of points associated with Setosa. If we were to call `plot` again, this would clear the previous plot.

Basic Data Visualization

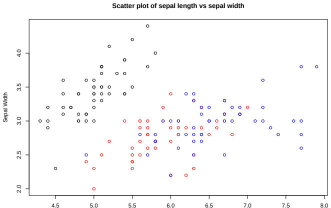
Stacking plots (without ggplot2)

Rather, we call `points` to add scatter plots to existing plot. `points` does not clear the previous plot.

```r
plot(data[data$Species == "setosa", ]$Sepal.Length,
     data[data$Species == "setosa", ]$Sepal.Width,
     xlab = 'Sepal Length',  ylab= 'Sepal Width',
     xlim = range(as.matrix(data$Sepal.Length)),
     ylim = range(as.matrix(data$Sepal.Width)),
     main= 'Scatter plot of sepal length vs sepal width')

points(data[data$Species == "versicolor", ]$Sepal.Length,
       data[data$Species == "versicolor", ]$Sepal.Width,
       col = 'red')

points(data[data$Species == "virginica", ]$Sepal.Length,
       data[data$Species == "virginica", ]$Sepal.Width, col = 'blue')
```



Basic Data Visualization

Scatter plot matrix

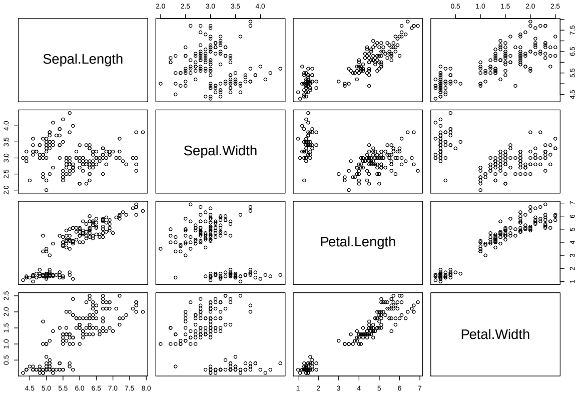
It is quite cumbersome to call to `plot` multiple times to create scatter plots for various pairs of explanatory variables.

There exists a convience function, `pairs`, that will create a matrix of scatter plots for all possible explanatory variable combination.

We give the `pairs` the first four columns of `data`. It will create a scatter plot matrix for sepal length, sepal width, petal length and petal width.

Scatter plot matrix

```
pairs(data[,c(1:4)])
```



Basic Data Visualization

Correlation matrix

From the scatter plot matrix above, we can see the qualitative correlation patterns between explanatory variables. We can also calculate these correlations explicitly and as a matrix using the `cor` function.

```r
cor(data[,c(1:4)])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Linear regression

Given a response variable, y, explanatory variables, X_i, and assuming that

$$y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_n X_{ni} + \varepsilon$$

where ε is a noise term, linear regression attempts the find the coefficients β_i that makes ε as small as possible.

The formula above assumes that response variable is a linear function explanatory variables. The noise term is added to account for the fact that most data is noisy and will not perfectly fit its ‘true’ function.

Linear regression also assumes that the noise is normally distributed with zero mean.

Linear regression

If these two assumptions are violated then

$$r_i = y_i - \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_n X_{ni}$$

and $\sum_{i=1}^{m} r_i^2$ is generally a large number. $\sum_{i=1}^{m} r_i^2$ is called the residual standard error.

Linear regression syntax

To regress the response variable, `Sepal.Width`, with explanatory variables, `Sepal.Length`, `Species`, `Petal.Length` and `Petal.Width`, we use the `lm` function.

The first argument is `lm` is the formula. The name of the column of the response variable is written first. It is followed by a tilde, `~`. After the tilde, we write names of the explanatory variable each separated by a `+`. `1` can also be added to the formula to represent a constant term.

Linear regression syntax

General Syntax: Constant Term

First, I use the formula `Sepal.Width ~ 1`. This formula is equivalent to

$$\text{Sepal.Width} = \beta_0 + \varepsilon.$$

Note that constant term is simply the mean response variable.

Linear regression syntax

General Syntax: Constant Term

```
ols <- lm(Sepal.Width ~ 1, data = data)
summary(ols)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ 1, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.05733 -0.25733 -0.05733  0.24267  1.34267
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.05733    0.03559   85.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4359 on 149 degrees of freedom
```

```
print(mean(data$Sepal.Width))
```

```
## [1] 3.057333
```

Linear regression syntax

General Syntax: Constant Term

Note that constant term is simply the mean response variable.

```
coef(ols)
```

```
## (Intercept)
##    3.057333
```

```
print(mean(data$Sepal.Width))
```

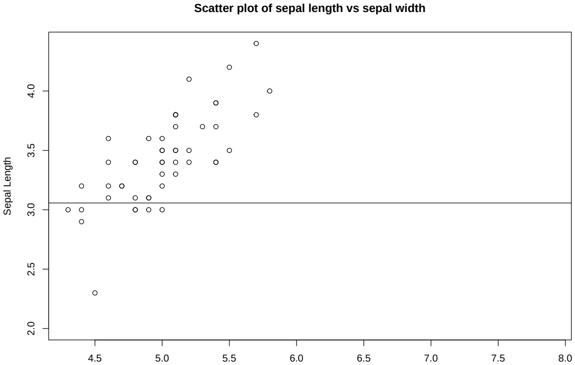
```
## [1] 3.057333
```

Linear regression syntax

General Syntax: Constant Term

```
plot(data[data$Species == "setosa", ]$Sepal.Length,
     data[data$Species == "setosa", ]$Sepal.Width,
     xlab = 'Sepal Width',  ylab= 'Sepal Length',
     xlim = range(as.matrix(data$Sepal.Length)),
     ylim = range(as.matrix(data$Sepal.Width)),
     main= 'Scatter plot of sepal length vs sepal width')

coefs = coef(ols)
abline(coefs[1],0)
```



Linear regression syntax

General Syntax: Explanatory Variable and Constant Term

I use then formula `Sepal.Width ~ 1 + Sepal.Length`. This formula is equivalent to

$$\text{Sepal.Width} = \beta_0 + \beta_1 \text{Sepal.Length} + \varepsilon$$

Linear regression syntax

General Syntax: Explanatory Variable and Constant Term

```r
ols <- lm(Sepal.Width ~ 1 + Sepal.Length, data = data)
summary(ols)
```

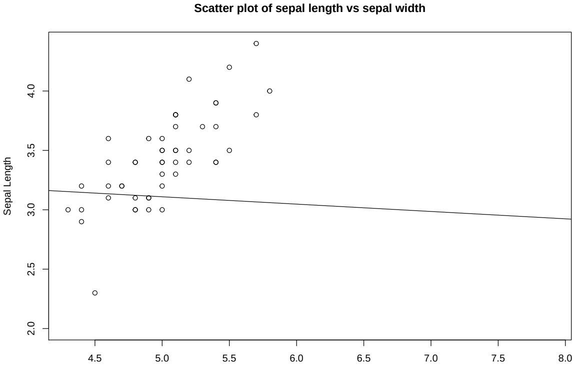
```
##
## Call:
## lm(formula = Sepal.Width ~ 1 + Sepal.Length, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1095 -0.2454 -0.0167  0.2763  1.3338
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.41895    0.25356   13.48   <2e-16 ***
## Sepal.Length -0.06188    0.04297   -1.44    0.152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4343 on 148 degrees of freedom
## Multiple R-squared:  0.01382,    Adjusted R-squared:  0.007159
## F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

Linear regression syntax

General Syntax: Explanatory Variable and Constant Term

```
plot(data[data$Species == "setosa", ]$Sepal.Length,
     data[data$Species == "setosa", ]$Sepal.Width,
     xlab = 'Sepal Width',  ylab= 'Sepal Length',
     xlim = range(as.matrix(data$Sepal.Length)),
     ylim = range(as.matrix(data$Sepal.Width)),
     main= 'Scatter plot of sepal length vs sepal width')

coefs = coef(ols)
abline(coefs[1],coefs[2])
```



Linear regression syntax

General Syntax: Factors

I use then formula `Sepal.Width ~ 1 + Sepal.Length + as.factor(Species)`. This formula is equivalent to

$$\text{Sepal.Width} = \beta_0 + \beta_1 \text{Sepal.Length} + \beta_2 I(\text{Species} = \text{Veriscolor}) + \beta_3 I(\text{Species} = \text{Virginica}) + \varepsilon$$

Linear regression syntax

General Syntax: Factors

```r
ols <- lm(Sepal.Width ~ 1 + Sepal.Length + as.factor(Species), data = data)
summary(ols)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ 1 + Sepal.Length + as.factor(Species),
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95096 -0.16522  0.00171  0.18416  0.72918
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.67650    0.23536   7.123 4.46e-11 ***
## Sepal.Length                  0.34988    0.04630   7.557 4.19e-12 ***
## as.factor(Species)versicolor -0.98339    0.07207 -13.644  < 2e-16 ***
## as.factor(Species)virginica  -1.00751    0.09331 -10.798  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.289 on 146 degrees of freedom
## Multiple R-squared:  0.5693, Adjusted R-squared:  0.5604
## F-statistic: 64.32 on 3 and 146 DF,  p-value: < 2.2e-16
```

Linear regression syntax

General Syntax: Factors

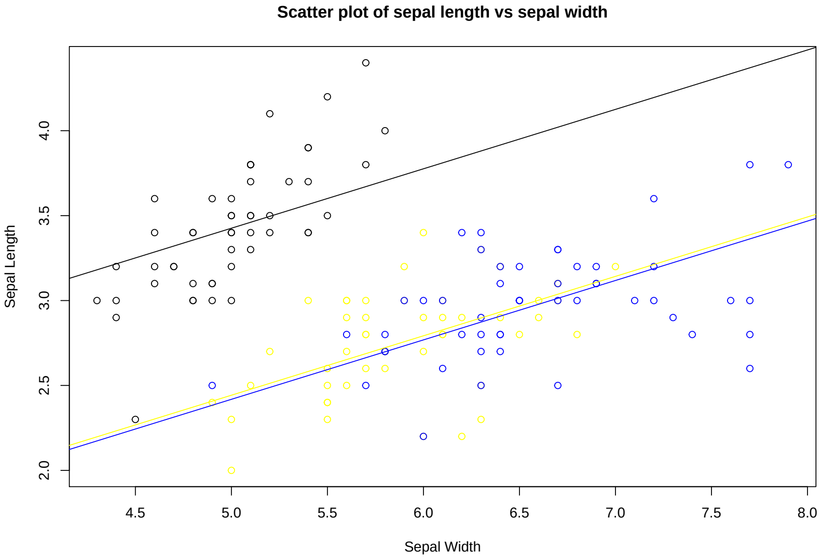
```r
plot(data[data$Species == "setosa", ]$Sepal.Length,
     data[data$Species == "setosa", ]$Sepal.Width,
     xlab = 'Sepal Width',  ylab= 'Sepal Length',
     xlim = range(as.matrix(data$Sepal.Length)),
     ylim = range(as.matrix(data$Sepal.Width)),
     main= 'Scatter plot of sepal length vs sepal width')

points(data[data$Species == "versicolor", ]$Sepal.Length,
       data[data$Species == "versicolor", ]$Sepal.Width,
       col = 'yellow')

points(data[data$Species == "virginica", ]$Sepal.Length,
       data[data$Species == "virginica", ]$Sepal.Width, col = 'blue')
coefs = coef(ols)
abline(coefs[1],coefs[2])
abline(coefs[3] + coefs[1],coefs[2],col='yellow')
abline(coefs[4] +coefs[1],coefs[2],col='blue')
```

Linear regression syntax

General Syntax: Factors



Linear regression syntax

Advanced Syntax: Nonlinear Regression

The `lm` function can be extended to nonlinear functions. For example, it possible include a quadratic term in our model.

$$\text{Sepal.Width} = \beta_1 \text{Sepal.Length} + \beta_2 \text{Sepal.Length}^2 + \varepsilon$$

To add a quadratic term to the model, add `I(Sepal.Length^2)` to the left side of the tilde, `~`. It is also possible to include higher order nonlinear terms, such as cubic, quintic, etc.

Linear regression syntax

Advanced Syntax: Nonlinear Regression

```
ols_quadratic <- lm(Sepal.Width ~ Sepal.Length + I(Sepal.Length^2), data = data)
summary(ols_quadratic)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + I(Sepal.Length^2),
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13070 -0.26310 -0.02446  0.25728  1.38725
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        6.41584    1.58499   4.048 8.33e-05 ***
## Sepal.Length      -1.08556    0.53625  -2.024   0.0447 *
## I(Sepal.Length^2)  0.08571    0.04476   1.915   0.0574 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4304 on 147 degrees of freedom
## Multiple R-squared:  0.03783,    Adjusted R-squared:  0.02474
## F-statistic:  2.89 on 2 and 147 DF,  p-value: 0.05877
```

Linear regression syntax

Advanced Syntax: Interaction term

It is also possible to include model interaction between explanatory variables.

$$\text{Sepal.Width} = \beta_1 \text{Sepal.Length} + \beta_2 \text{Petal.Length} + \beta_3 \text{Sepal.Length} \times \text{Petal.Length} + \varepsilon$$

Sepal.Length × Petal.Length models simple interaction between Sepal.Length and Petal.Length.

Linear regression syntax

Advanced Syntax: Interaction term

```r
ols_interaction <- lm(Sepal.Width ~ Sepal.Length + Petal.Length
                      + Sepal.Length*Petal.Length, data = data)
summary(ols_interaction)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length + Petal.Length + Sepal.Length *
##     Petal.Length, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.86960 -0.19846  0.00743  0.20704  0.72871
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                1.51011    0.64336   2.347 0.020257 *
## Sepal.Length               0.46940    0.12954   3.624 0.000400 ***
## Petal.Length              -0.42907    0.11832  -3.626 0.000397 ***
## Sepal.Length:Petal.Length  0.01795    0.02186   0.821 0.413063
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3239 on 146 degrees of freedom
## Multiple R-squared:  0.4589, Adjusted R-squared:  0.4478
## F-statistic: 41.28 on 3 and 146 DF,  p-value: < 2.2e-16
```

Linear regression syntax

Advanced Syntax: Interaction term

Note that using the formula Sepal.Width ~ Sepal.Length*Petal.Length produces the same result as Sepal.Length + Petal.Length + Sepal.Length*Petal.Length.

```
ols_interaction <- lm(Sepal.Width ~ Sepal.Length*Petal.Length, data = data)
summary(ols_interaction)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length * Petal.Length, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.86960 -0.19846  0.00743  0.20704  0.72871
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                1.51011    0.64336   2.347 0.020257 *
## Sepal.Length               0.46940    0.12954   3.624 0.000400 ***
## Petal.Length              -0.42907    0.11832  -3.626 0.000397 ***
## Sepal.Length:Petal.Length  0.01795    0.02186   0.821 0.413063
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3239 on 146 degrees of freedom
## Multiple R-squared:  0.4589, Adjusted R-squared:  0.4478
## F-statistic: 41.28 on 3 and 146 DF,  p-value: < 2.2e-16
```

Linear regression syntax

Advanced Syntax: Non-linear regression and Interaction term

```r
ols_q_i <- lm(Sepal.Width ~ Sepal.Length*as.factor(Species)
              + I(Sepal.Length^2), data = data)
summary(ols_q_i)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length * as.factor(Species) +
##     I(Sepal.Length^2), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71404 -0.15888  0.01535  0.16089  0.61676
##
## Coefficients:
##                                          Estimate Std. Error t value
## (Intercept)                              -2.17766    1.59715  -1.363
## Sepal.Length                              1.44156    0.60909   2.367
## as.factor(Species)versicolor              0.79275    0.93445   0.848
## as.factor(Species)virginica               0.84269    1.29009   0.653
## I(Sepal.Length^2)                        -0.06397    0.05959  -1.073
## Sepal.Length:as.factor(Species)versicolor -0.35909    0.17402  -2.064
## Sepal.Length:as.factor(Species)virginica  -0.36224    0.22839  -1.586
##                                          Pr(>|t|)
## (Intercept)                                0.1749
## Sepal.Length                               0.0193 *
## as.factor(Species)versicolor               0.3977
## as.factor(Species)virginica                0.5147
## I(Sepal.Length^2)                          0.2849
## Sepal.Length:as.factor(Species)versicolor  0.0409 *
## Sepal.Length:as.factor(Species)virginica   0.1149
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2722 on 143 degrees of freedom
## Multiple R-squared:  0.6257, Adjusted R-squared:   0.61
## F-statistic: 39.85 on 6 and 143 DF,  p-value: < 2.2e-16
```

Obtaining the residuals

Recall it is possible the measure the error between the model and response variables. Given least squares fit,

$$r_i = y_i - \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_n X_{ni}$$

r is the residual vector and $\sum_{i=1}^{m} r_i^2$ is the residual standard error.

Obtaining the residuals

It is possible to obtain residuals from the output of `lm`

```
res <- ols$residuals
head(data.frame(res=res))
```

```
##           res
## 1  0.03911127
## 2 -0.39091271
## 3 -0.12093668
## 4 -0.18594867
## 5  0.17409928
## 6  0.33414723
```

Obtaining the fitted values

It is also possible to get predicted model values from the output of `lm`.

```
pred <- ols$fitted.values
head(data.frame(pred=pred,orig=data$Sepal.Width ))
```

```
##       pred orig
## 1 3.460889  3.5
## 2 3.390913  3.0
## 3 3.320937  3.2
## 4 3.285949  3.1
## 5 3.425901  3.6
## 6 3.565853  3.9
```

Obtaining the ANOVA table

anova allows us to compare variance in the response captured by residuals and each term in the model.

```
anova(ols)
```

```
## Analysis of Variance Table
##
## Response: Sepal.Width
##                     Df  Sum Sq Mean Sq F value  Pr(>F)
## Sepal.Length         1  0.3913  0.3913  4.6851 0.03205 *
## as.factor(Species)   2 15.7225  7.8613 94.1304 < 2e-16 ***
## Residuals          146 12.1931  0.0835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
```

Obtaining the variance-covariance matrix

Covariance measures how well two variables vary together.

For an linear model, variance-covariance matrix is described by:

- (i, j) entry is the covariance between the ith coefficent and the j coefficient.
- (i, i) entry is the variance for the ith coefficent.

The `vcov` returns the variance-covariance matrix of a linear model.

```r
vcov(ols)
```

Obtaining the variance-covariance matrix

```
vcov_mat <- vcov(ols)
```

	(Intercept)	Sepal.Length	as.factor(Species)versicolor	as.factor(Species)virginica
(Intercept)	0.0553942	-0.0107319	0.0083104	0.0153076
Sepal.Length	-0.0107319	0.0021438	-0.0019937	-0.0033915
as.factor(Species)versicolor	0.0083104	-0.0019937	0.0051948	0.0048244
as.factor(Species)virginica	0.0153076	-0.0033915	0.0048244	0.0087059

Obtaining confidence intervals for the coefficients

We can use the `confint` to term the confidence interval of the coefficients of our linear model. The `level` argument specifies the percentage confidence.

```r
confint(ols, level = 0.95)
```

```
##                                   2.5 %     97.5 %
## (Intercept)                   1.2113479  2.1416523
## Sepal.Length                  0.2583728  0.4413874
## as.factor(Species)versicolor -1.1258331 -0.8409440
## as.factor(Species)virginica  -1.1919146 -0.8231061
```

Obtaining confidence interval of the mean response

We can use the `predict` to determine the predicted response values of data points from our linear model.

`predict` has arguments:

- 1st argument: the model object
- 2nd argument: dataframe of data points
- `interval`: this is an optional argument. This is the type of interval calculation
- `level`: this is an optional argument. This specifies the percentage confidence.
- `se.fit`: this is an optional argument. It specifies if stamdard errors are required.

Obtaining confidence interval of the mean response

Here, we take `interval="confidence"`.

```
predict(ols, data.frame(Sepal.Length=4.0, Species="setosa"),
        interval="confidence", level = 0.95, se.fit=TRUE)
```

```
## $fit
##        fit      lwr      upr
## 1 3.076021 2.953552 3.198489
##
## $se.fit
## [1] 0.06196695
##
## $df
## [1] 146
##
## $residual.scale
## [1] 0.288989
```

Obtaining prediction limits for new observation

Here, we take `interval="prediction"`.

```r
predict(ols, data.frame(Sepal.Length=4.0, Species="setosa"),
        interval="prediction", level = 0.95, se.fit=TRUE)
```

```
## $fit
##        fit      lwr      upr
## 1 3.076021 2.491896 3.660145
##
## $se.fit
## [1] 0.06196695
##
## $df
## [1] 146
##
## $residual.scale
## [1] 0.288989
```

Obtaining confidence band for the entire regression line

```r
alpha = 0.05
n = dim(iris)[1]
ci <- predict(ols, data.frame(Sepal.Length=4.0,
                               Species="setosa"),
              interval="confidence", level= 1-alpha,
              se.fit=TRUE)
yh.hat <- ci$fit[1]
se.yh.hat <- ci$se.fit
w <- sqrt(2*qf(1-alpha, 2, n-2))
lower_bound <- yh.hat - w*se.yh.hat
upper_bound <- yh.hat + w*se.yh.hat
band <- c(lower_bound, upper_bound)
band
```

```
## [1] 2.922793 3.229248
```

Standardized regression

It is difficult to compare size coefficients if the measurement units of the covariates differ widely.

Standardization removed the scale of the covariates. Standardization substracts the mean and divides by the standard deviation.

To standardize our model, we convert the unstandardize model to a standard model using `lm.beta`.

Standardized regression

```
lm.beta(ols)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ 1 + Sepal.Length + as.factor(Species),
##     data = data)
##
## Standardized Coefficients::
##                  (Intercept)                  Sepal.Length
##                    0.0000000                     0.6647082
## as.factor(Species)versicolor  as.factor(Species)virginica
##                   -1.0671319                    -1.0933079
```

Model selection

There are various statistical tests and estimators which can be used to compare models. In this workshop, we will be covering:

- F statistic
- AIC
- BIC
- adjusted R^2.

General linear test approach

The F test tells us if there is a statistically significant decrease in residual standard error. We use a significance level 0.05 of in this workshop.

We use the `anova` function to conduct the F-test between pairs of models. Note that the `anova` function can take more than two models.

Comparison of interaction model with and without quadratic term

Comparing the quadratic model with interaction, `ols_q_i`, and the model without the quadratic term, `ols_interaction`,

```
anova(ols_interaction,ols_q_i)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Width ~ Sepal.Length * Petal.Length
## Model 2: Sepal.Width ~ Sepal.Length * as.factor(Species) + I(Sepal.Length^2)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    146 15.316
## 2    143 10.595  3    4.7219 21.245 1.939e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is significant reduction in error after including the quadratic term.

Comparison of quadratic model with and without interaction term

Comparing the quadratic model with interaction, `ols_q_i`, and the model without the interaction term, `ols_quadratic`,

```
anova(ols_quadratic,ols_q_i)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Width ~ Sepal.Length + I(Sepal.Length^2)
## Model 2: Sepal.Width ~ Sepal.Length * as.factor(Species) + I(Sepal.Length^2)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    147 27.236
## 2    143 10.595  4    16.642 56.155 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is significant reduction in error after including the interaction term.

Comparison of linear models with and without quadratic term

Comparing the quadratic model, `ols_quadratic`, and the model without the quadratic term, `ols`,

```
anova(ols_quadratic, ols)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Width ~ Sepal.Length + I(Sepal.Length^2)
## Model 2: Sepal.Width ~ 1 + Sepal.Length + as.factor(Species)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    147 27.236
## 2    146 12.193  1    15.043 180.12 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is significant reduction in error after including the quadratic term.

Comparison of linear models with and without interaction term

Comparing the interaction model, `ols_interaction`, and the model without the interaction term, `ols`,

```
anova(ols,ols_interaction)
```

```
## Analysis of Variance Table
##
## Model 1: Sepal.Width ~ 1 + Sepal.Length + as.factor(Species)
## Model 2: Sepal.Width ~ Sepal.Length * Petal.Length
##   Res.Df    RSS Df Sum of Sq F Pr(>F)
## 1    146 12.193
## 2    146 15.316  0   -3.1234
```

There is significant reduction in error after including the interaction term.

AIC, BIC, and Adjusted R^2

```r
aic <- c(AIC(ols_quadratic), AIC(ols_interaction))
bic <- c(BIC(ols_quadratic), BIC(ols_interaction))
ar2 <- c(summary(ols_quadratic)$adj.r.squared,
         summary(ols_interaction)$adj.r.squared)
cbind(aic, bic, ar2)
```

```
##            aic      bic       ar2
## [1,] 177.76821 189.8107 0.0247357
## [2,]  93.42623 108.4794 0.4477939
```

Step wise selection

To use stepwise selection, you need to define a base model without predictors and a full model with all possible predictors to be considered. If you want quadratic or interaction terms, you need to add them explicitely; otherwise, .adds all the variables in the dataset.

```r
base <- lm(Sepal.Width ~ 1, data=data)
retailer_quadratic <- lm(Sepal.Width ~ . + I(Sepal.Length^2), data=data)
retailer_interaction <- lm(Sepal.Width ~ . + Sepal.Length*Species, data=data)
```

Step wise selection

Specifying `direction = "both"` selects step wise selection. If you're not interested in looking at the steps taken by the algorithm, you can select `trace = FALSE`.

```
step_1 <- step(base, scope = list(upper=retailer_quadratic,
                                  lower= ~1),
               direction = "both",
               trace=TRUE)
```

Step wise selection

```
## Start:  AIC=-248.13
## Sepal.Width ~ 1
##
##                     Df Sum of Sq    RSS     AIC
## + Species            2   11.3449 16.962 -320.95
## + Petal.Length       1    5.1960 23.111 -276.55
## + Petal.Width        1    3.7945 24.512 -267.72
## + Sepal.Length       1    0.3913 27.916 -248.22
## <none>                           28.307 -248.13
## + I(Sepal.Length^2)  1    0.3115 27.995 -247.79
##
## Step:  AIC=-320.95
## Sepal.Width ~ Species
##
##                     Df Sum of Sq    RSS     AIC
## + Sepal.Length       1    4.7689 12.193 -368.46
## + I(Sepal.Length^2)  1    4.1911 12.771 -361.52
## + Petal.Width        1    3.7554 13.207 -356.49
## + Petal.Length       1    2.4225 14.540 -342.07
## <none>                           16.962 -320.95
## - Species            2   11.3449 28.307 -248.13
##
## Step:  AIC=-368.46
## Sepal.Width ~ Species + Sepal.Length
##
##                     Df Sum of Sq    RSS     AIC
## + Petal.Width        1    1.5037 10.689 -386.21
## + I(Sepal.Length^2)  1    1.2775 10.916 -383.07
## <none>                           12.193 -368.46
## + Petal.Length       1    0.0210 12.172 -366.72
## - Sepal.Length       1    4.7689 16.962 -320.95
## - Species            2   15.7225 27.916 -248.22
##
## Step:  AIC=-386.21
```

Step wise selection

```
step_2 <- step(base, scope = list(upper=retailer_interaction,
                                  lower= ~1),
               direction = "both",
               trace=TRUE)
```

Step wise selection

```
## Start:  AIC=-248.13
## Sepal.Width ~ 1
##
##                Df Sum of Sq    RSS     AIC
## + Species       2   11.3449 16.962 -320.95
## + Petal.Length  1    5.1960 23.111 -276.55
## + Petal.Width   1    3.7945 24.512 -267.72
## + Sepal.Length  1    0.3913 27.916 -248.22
## <none>                      28.307 -248.13
##
## Step:  AIC=-320.95
## Sepal.Width ~ Species
##
##                Df Sum of Sq    RSS     AIC
## + Sepal.Length  1    4.7689 12.193 -368.46
## + Petal.Width   1    3.7554 13.207 -356.49
## + Petal.Length  1    2.4225 14.540 -342.07
## <none>                      16.962 -320.95
## - Species       2   11.3449 28.307 -248.13
##
## Step:  AIC=-368.46
## Sepal.Width ~ Species + Sepal.Length
##
##                        Df Sum of Sq    RSS     AIC
## + Petal.Width           1    1.5037 10.689 -386.21
## + Sepal.Length:Species  2    1.5132 10.680 -384.34
## <none>                              12.193 -368.46
## + Petal.Length          1    0.0210 12.172 -366.72
## - Sepal.Length          1    4.7689 16.962 -320.95
## - Species               2   15.7225 27.916 -248.22
##
## Step:  AIC=-386.21
## Sepal.Width ~ Species + Sepal.Length + Petal.Width
##
```

Step wise selection

```r
summary(step_1)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Species + Sepal.Length + Petal.Width +
##     I(Sepal.Length^2), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85363 -0.13206  0.00619  0.17297  0.78127
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.60830    1.14098  -2.286   0.0237 *
## Speciesversicolor -1.59481    0.12661 -12.596  < 2e-16 ***
## Speciesvirginica  -1.88631    0.19275  -9.786  < 2e-16 ***
## Sepal.Length       1.80619    0.38096   4.741 5.05e-06 ***
## Petal.Width        0.49857    0.11200   4.451 1.70e-05 ***
## I(Sepal.Length^2) -0.12422    0.03065  -4.052 8.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2581 on 144 degrees of freedom
## Multiple R-squared:  0.661,  Adjusted R-squared:  0.6493
## F-statistic: 56.16 on 5 and 144 DF,  p-value: < 2.2e-16
```

Step wise selection

```r
summary(step_2)
```

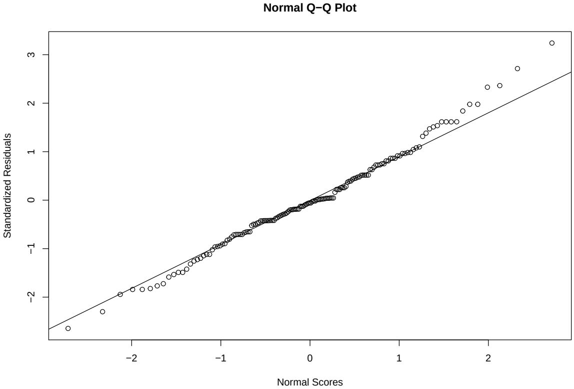
```
##
## Call:
## lm(formula = Sepal.Width ~ Species + Sepal.Length + Petal.Width +
##     Species:Sepal.Length, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.77834 -0.14765  0.02457  0.16428  0.62667
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -0.4731     0.5105  -0.927   0.3556
## Speciesversicolor                1.2981     0.6573   1.975   0.0502 .
## Speciesvirginica                 1.2252     0.6501   1.884   0.0615 .
## Sepal.Length                     0.7515     0.1021   7.363 1.31e-11 ***
## Petal.Width                      0.5662     0.1095   5.172 7.68e-07 ***
## Speciesversicolor:Sepal.Length  -0.5503     0.1239  -4.442 1.77e-05 ***
## Speciesvirginica:Sepal.Length   -0.5883     0.1163  -5.058 1.28e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2508 on 143 degrees of freedom
## Multiple R-squared:  0.6822, Adjusted R-squared:  0.6688
## F-statistic: 51.15 on 6 and 143 DF,  p-value: < 2.2e-16
```

Diagnostics

Normal probability plot

Each residual is plotted against its expected value under normality. Near linearity suggests normality.

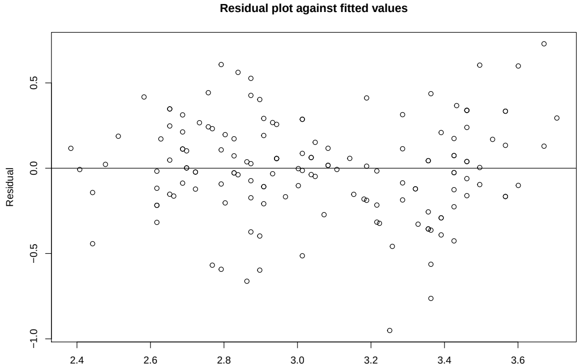
```r
std_res <- rstandard(ols_quadratic)
qqnorm(std_res, ylab="Standardized Residuals", xlab="Normal Scores")
qqline(std_res)
```



Residual plots

Under linearity and constant variance, this should appear as a random cloud of points centered at 0. As long as no residuals stand out from the others, the model fits all observations.

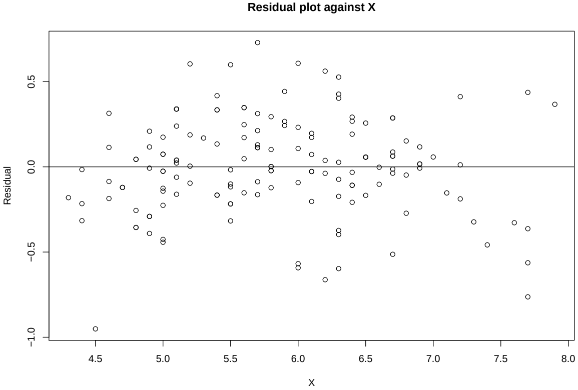
```r
plot(res ~ pred, xlab="Fitted", ylab="Residual",
     main="Residual plot against fitted values")
abline(h=0)
```



Residual plots

Under linearity and constant variance, this should appear as a random cloud of points centered at 0.

```r
plot(res ~ data$Sepal.Length, xlab="X",
     ylab="Residual", main="Residual plot against X")
abline(h=0)
```



Test for multicollinearity

This procedure works for numerical variables.

```r
# See also function vif() in package "car"
VIF <- 1/(1-summary(lm(Sepal.Length ~ as.factor(Species),
                          data = data))$r.squared)
VIF
```

```
## [1] 2.622646
```

Test for heteroskedasticity

This is the Breusch-Pagan test.

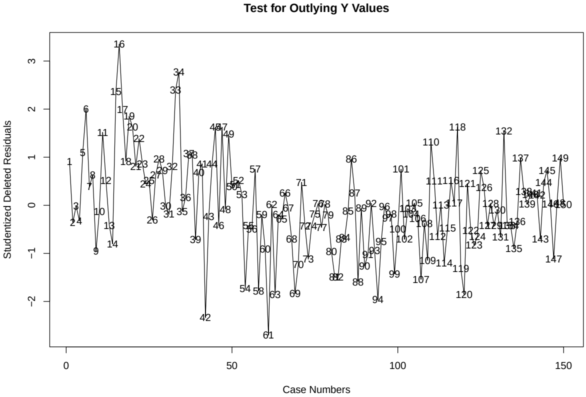
```
bptest(ols_quadratic, studentize = FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  ols_quadratic
## BP = 6.2857, df = 2, p-value = 0.04316
```

Test for outlying Y observations–studentized deleted residual

This procedure follows the recommendations of Kutner et al. (2005), Applied Linear Statistical Models.

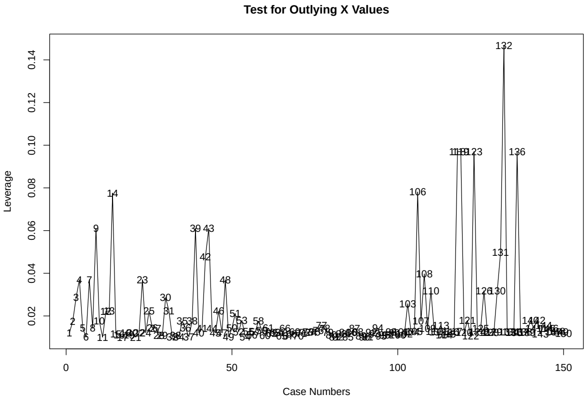
```
p <- 5 # numer of parameters
case <- c(1:n) # n defined above
plot(case, rstudent(ols_quadratic), type="l", xlab="Case Numbers",
     ylab="Studentized Deleted Residuals", main="Test for Outlying Y Values")
text(case, rstudent(ols_quadratic), case)
```



Test for outlying X observations–hat matrix leverage values

This procedure follows the recommendations of Kutner et al. (2005), Applied Linear Statistical Models.

```r
leverage <- hatvalues(ols_quadratic)
plot(case, leverage, type="l", xlab="Case Numbers",
     ylab="Leverage", main="Test for Outlying X Values")
text(case, leverage, case)
abline(h=0.5, col=2)
```

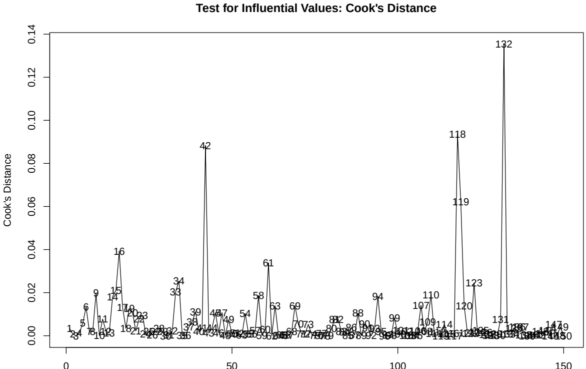


Tests for influential observations

Cook's distance

Usually less than 10% or 20% indicates little influence on the fitted values. Near 50% or more indicates influence on the fit of the regression function.

```
plot(case, cooks.distance(ols_quadratic), type="l", xlab="Case Numbers",
     ylab="Cook's Distance", main = "Test for Influential Values: Cook's Distance")
text(case, cooks.distance(ols_quadratic))
```



Cook's distance

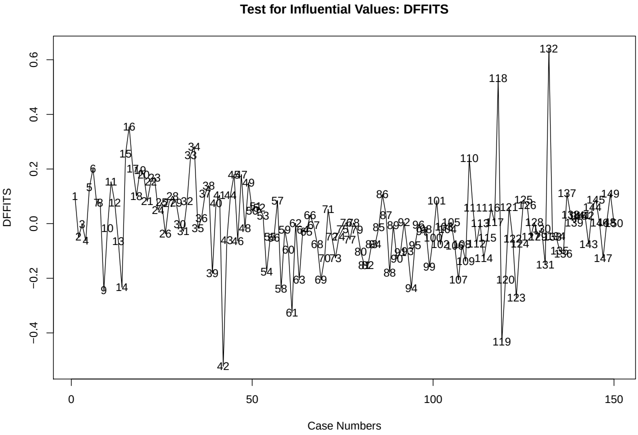
```
inf_obs <- which(cooks.distance(ols_quadratic)>0.5)
inf_obs
```

```
## named integer(0)
```

DFFITS

For small to medium-sized datasets, `abs(DFFITS)>1` indicates influence. The procedure below is for large datasets.

```r
plot(case, dffits(ols_quadratic), type="l", xlab="Case Numbers",
     ylab="DFFITS", main = "Test for Influential Values: DFFITS")
text(case, dffits(ols_quadratic))
```



DFFITS

```r
inf_obs2 <- which(abs(dffits(ols_quadratic))>2/sqrt(p/n))
inf_obs2
```

```
## named integer(0)
```

DFBETAS

For small to medium-sized datasets, `abs(DFBETAS)>1` indicates influence. The procedure below is for large datasets.

```r
inf_obs3 <- which(abs(dfbeta(ols_quadratic))>2/sqrt(n))
inf_obs3
```

```
##  [1]   9  14  15  16  19  23  34  39  42  58  61  63  69  88 118 119 120
## [18] 123 132 192 268 269 282
```