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.

```r
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.

```
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`.

```r
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))
```

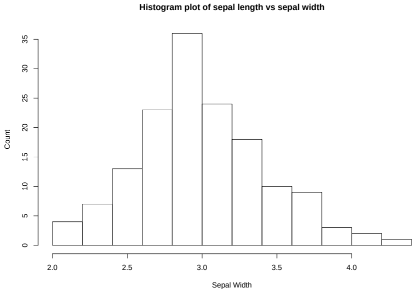
```
##         mean standard_deviation number_of_observations
## 2   5.000000                 NA                      1
## 2.2 6.066667           0.115470                      3
## 2.3 5.325000           0.767572                      4
## 2.4 5.300000           0.346410                      3
## 2.5 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.

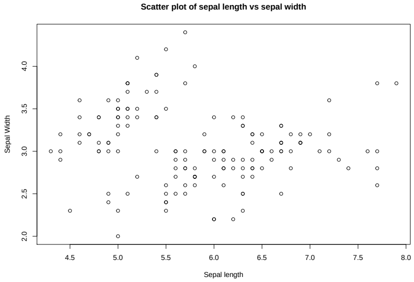
```r
hist(data$Sepal.Width,
     xlab='Sepal Width',ylab='Count',
     main= 'Histogram plot of sepal length vs sepal width')
```



Scatter Plot

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

```r
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

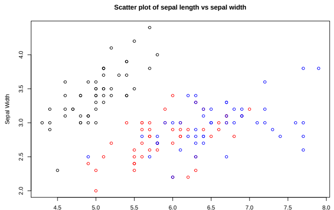
Stacking plots (without ggplot2)

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

```
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

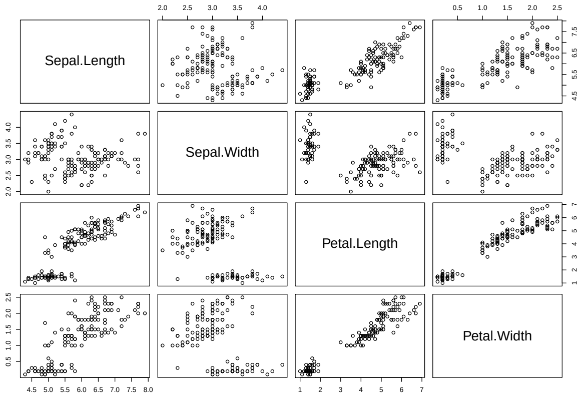
It is quite cumbersome to call to `plot` multiple times to create scatter plots for various pairs of explanatory variables.

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

```r
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.

```
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))
```

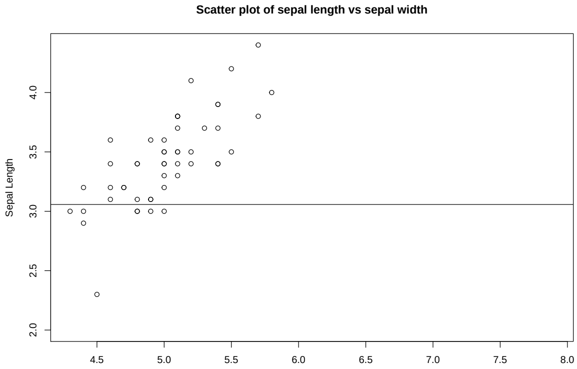
```
## [1] 3.057333
```

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

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

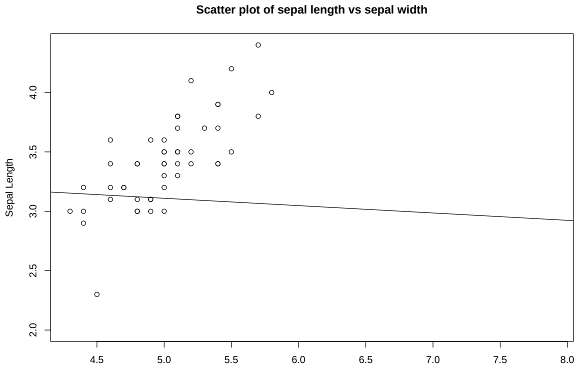
```
##
## Call:
## lm(formula = Sepal.Width ~ 1 + Sepal.Length, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1095 -0.2454 -0.0167  0.2763  1.3338
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.41895    0.25356   13.48   <2e-16 ***
## 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

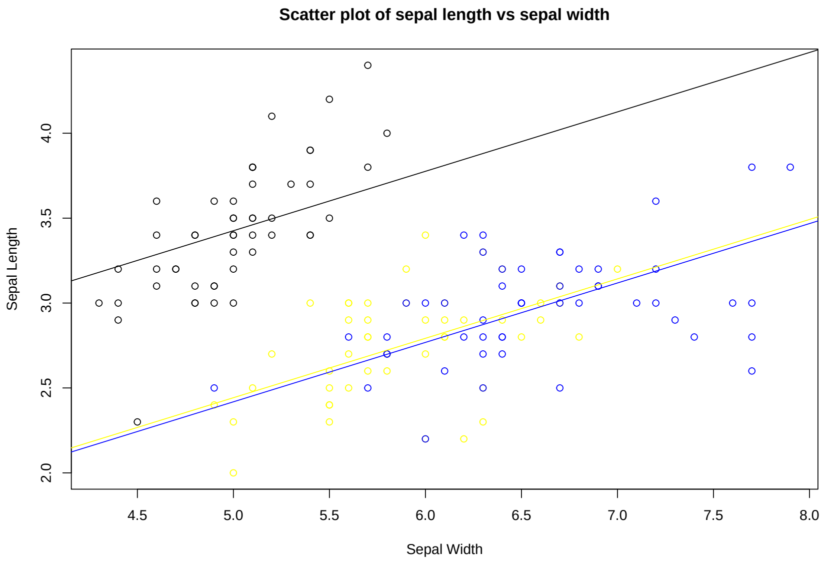
```r
plot(data[data$Species == "setosa", ]$Sepal.Length,
     data[data$Species == "setosa", ]$Sepal.Width,
     xlab = 'Sepal Width',  ylab= 'Sepal Length',
     xlim = range(as.matrix(data$Sepal.Length)),
     ylim = range(as.matrix(data$Sepal.Width)),
     main= 'Scatter plot of sepal length vs sepal 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 $\times$ Petal.Length models simple interaction between Sepal.Length and Petal.Length.

Linear regression syntax

Advanced Syntax: Interaction term

```
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
```

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`

```r
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
```