Getting Started with R

Sugitha Devarajan

What is R?

- **R** is a free and powerful statistical software for analyzing and visualizing data. R was developed by Ross Ihaka and Robert Gentleman in 1993.
- The official R software environment is written primarily in C, FORTRAN, and R itself.
- R and its libraries implement various statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, spatial and time-series analysis, classification, clustering, and others.
- R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages.
- R is an interpreted language; users typically access it through a command-line interpreter. If a user types 2+2 at the R command prompt and presses enter, the computer replies with 4.

What is R Studio

- **RStudio** is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.
- It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser.
- RStudio is divided into 4 “Panes”:
    - the Source for your scripts and documents (top-left, in the default layout),
    - the R Console (bottom-left),
    - your Environment/History (top-right), and
    - your Files/Plots/Packages/Help/Viewer (bottom-right).

Packages/Libraries

- Packages are collections of additional functions that can be loaded on demand. They commonly include example data that can be used to demonstrate those functions. Although R comes with many common statistical functions and models, most of our work requires additional packages. R they are calls them **libraries.**

Data types and structure in R

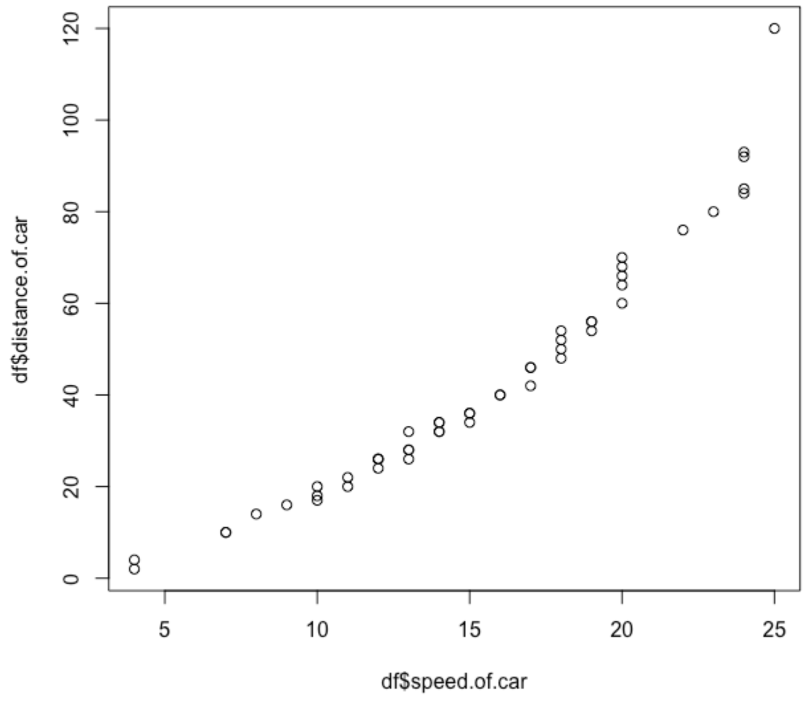
- R has 6 basic data types.
    - **Numeric**- Numbers with decimals. (Ex: 1.0, 10.5, 4.5, etc.)
    - **Integer Data**- Whole numbers (Ex: 11, 45, 78, etc.)
    - **Factor Data**- Categorical data (Ex: Red, Blue, Green, Purple, etc.)
    - **Ordinal Data**- Ordered data (Ex: educational levels, temperature, etc.)
    - **Character Data**- String values, which are characters (words) with quotes around them. (Ex: "Red", "Blue", "Green", "Purple", etc.)
    - **Logical**- TRUE or TRUE (Always capitalize TRUE or FALSE)
- R has many **data structures**. These include
    - atomic vector
    - list
    - matrix
    - data frame
    - factors

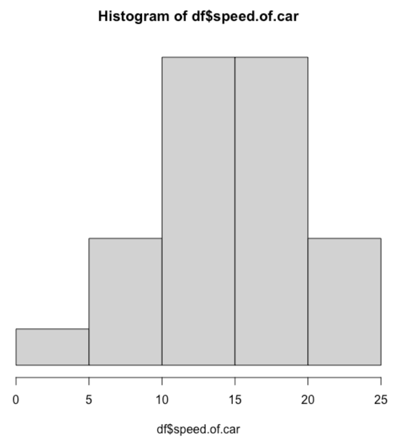
Missing Values

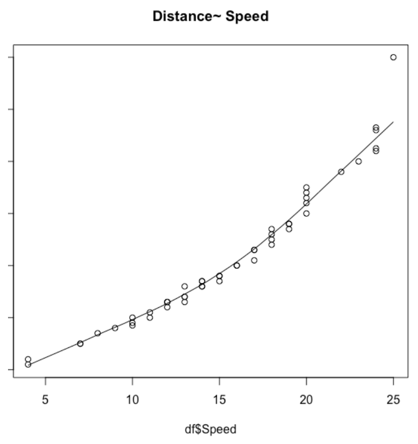
- A common task in data analysis is dealing with missing values. In R, missing values are often represented by NA or some other value that represents missing values (i.e. 99)
- To identify missing values use is.na() which returns a logical vector with TRUE in the element locations that contain missing values represented by NA. is.na() will work on vectors, lists, matrices, and data frames.
- We can exclude missing values in a couple different ways. First, if we want to exclude missing values from mathematical operations use the na.rm = TRUE argument. If you do not exclude these values most functions will return an NA.
- Another useful function in R to deal with missing values is na.omit() which delete incomplete observations.

Visualization in R – Cars

- From the histogram we infer that max speed in the data frame is 25 and 10 to 20 being repeated most.
- Speed and Distance have somewhat positive linear relation.
- Using scatter.smooth I can fit a curve to show the linear realtionship







Sample Data in R - Cars

- Typically, when you separate a **data set** into a **training set and testing set**, most of the **data** is used for **training**, and a smaller portion of the **data** is used for **testing.** After a model has been processed by using the **training set**, you **test** the model by making predictions against the **test set**.
- Statisticians often have to take samples of data and then calculate statistics. Taking a sample is easy with R because a sample is really nothing more than a subset of data. To do so, you make use of sample(), which takes a vector as input; then you tell it how many samples to draw from that list.
- The **sample**() **function** allows you to draw random **samples** of elements (scalars) from a vector.
- In R, you use the set.seed() function to specify your seed starting value. The argument to set.seed() is any integer value.
  The **set**. **seed**() function **sets** the starting number used to generate a sequence of random numbers – it ensures that you get the same result if you start with that same **seed** each time you run the same process. For example, if I use the sample() function immediately after setting a **seed**, I will always get the same sample.

```
set.seed(100)
trainingRowIndex <- sample(1:nrow(df), 0.7*nrow(df))  # row indices for training data
trainingData <- df[trainingRowIndex, ]  # model training data
testData  <- df[-trainingRowIndex, ]   # test data
```

Linear Regression Model – Cars

- Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is an explanatory variable, and the other is a dependent variable.
- Before attempting to fit a linear model to observed data, a modeler should first determine if there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association between the two variables. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.
- A linear regression line has an equation of the form ***Y = a + bX***, where ***X*** is the explanatory variable and ***Y*** is the dependent variable. The slope of the line is ***b***, and ***a*** is the intercept (the value of ***y*** when ***x*** = 0).
- In cars df scatter plot, we saw the linear behavior in slide 7 and the correlation coefficient is

```
> cor(df$Speed, df$Distance)
[1] 0.963559
```

Model and Prediction – Cars

- We split the data 70/30 for training and testing
- The goal here is to predict distance when only the speed of the car is known.
- The function used for building linear models is lm().
- The lm() function takes in two main arguments-Formula and Data
- The data is typically a data.frame object and the formula is an object of class formula.

Model Summary

lm(formula = Distance ~ Speed, data = trainingData). *# Formula used to fit the data*

Residuals: *#Residuals are essentially the difference between the actual observed response values (distance to stop distance in our case) and the response values that the model predicted. The Residuals section of the model output breaks it down into 5 summary points*

```
  Min     1Q Median     3Q    Max

-7.611 -3.640 -1.212  2.603 12.702
```

Coefficients: *#The coefficients are two unknown constants that represent the intercept and slope terms in the linear model.*

```
            Estimate Std. Error t value Pr(>|t|)

(Intercept) -26.6452     2.7589  -9.658 3.84e-11 ***
```

*#The intercept, in our example, is essentially the expected value of the distance required for a car to stop when we consider the average speed of all cars in the dataset. In other words, it takes an average car in our dataset **-26.64** feet to come to a stop*

```
Speed         4.4857     0.1763  25.448  < 2e-16 ***
```

*#Slope - the effect speed has in distance required for a car to stop. The slope term in our model is saying that for every 1 mph increase in the speed of a car, the required distance to stop goes up by **4.4857** feet.*

#The coefficient Standard Error measures the average amount that the coefficient estimates vary from the actual average value of our response variable. We'd ideally want a lower number relative to its coefficients.

#The coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0.

#The *Pr(>t)* acronym found in the model output relates to the probability of observing any value equal or larger than *t*. A small p-value indicates that it is unlikely we will observe a relationship between the predictor (speed) and response (distance) variables due to chance. Typically, a p-value of 5% or less is a good cut-off point.

#Note the 'signif. codes:' associated to each estimate. Three stars (or asterisks) represent a highly significant p-value. Consequently, a small p-value for the intercept and the slope indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between speed and distance.

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 5.31 on 33 degrees of freedom #Residual Standard Error is measure of the *quality* of a linear regression fit. Percentage error in our case is 16%. Simplistically, degrees of freedom are the number of data points that went into the estimation of the parameters used after taking into account these parameters (restriction)

Multiple R-squared: 0.9515,   Adjusted R-squared: 0.95

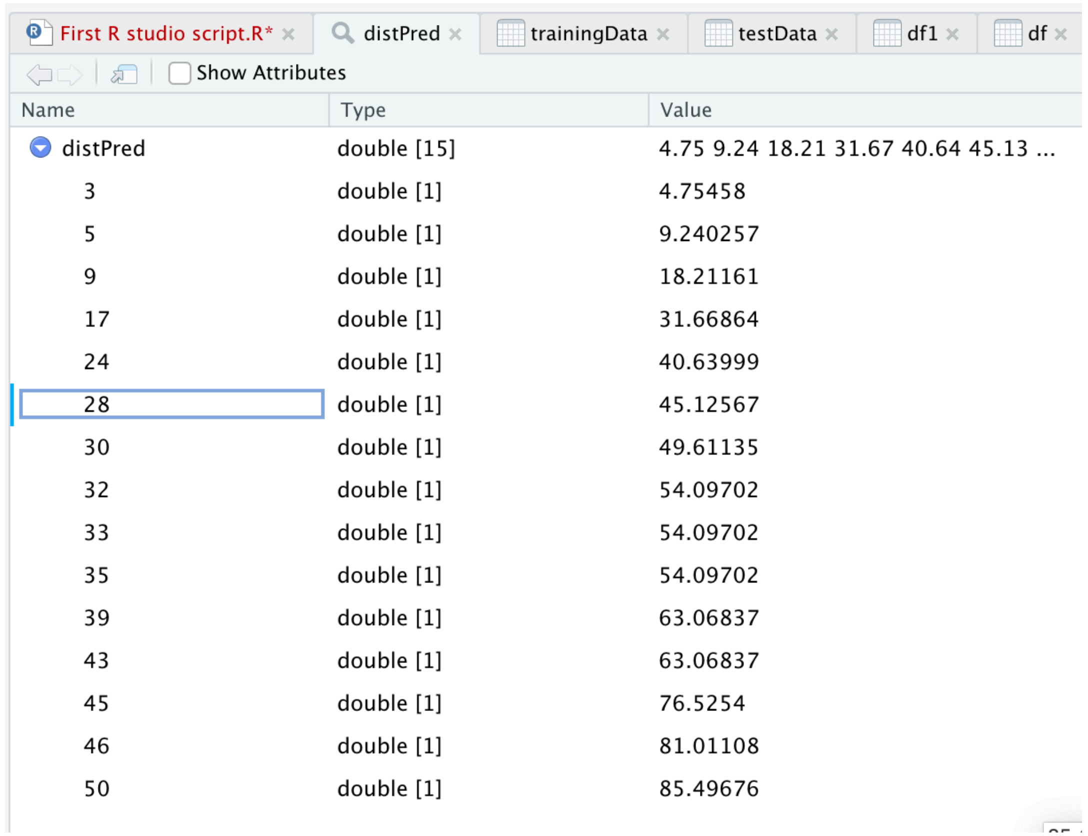
#The R-squared (R2) statistic provides a measure of how well the model is fitting the actual data. It takes the form of a proportion of variance. R2R2 is a measure of the linear relationship between our predictor variable (speed) and our response / target variable (dist). It always lies between 0 and 1 (i.e.: a number near 0 represents a regression that does not explain the variance in the response variable well and a number close to 1 does explain the observed variance in the response variable). In multiple regression settings, the R2 will always increase as more variables are included in the model. That's why the adjusted R2 is the preferred measure as it adjusts for the number of variables considered.

F-statistic: 647.6 on 1 and 33 DF, p-value: < 2.2e-16

#F-statistic is a good indicator of whether there is a relationship between our predictor and the response variables. The further the F-statistic is from 1 the better it is. However, how much larger the F-statistic needs to be dependents on both the number of data points and the number of predictors. The reverse is true as if the number of data points is small, a large F-statistic is required to be able to ascertain that there may be a relationship between predictor and response variables.

Predicted from test data

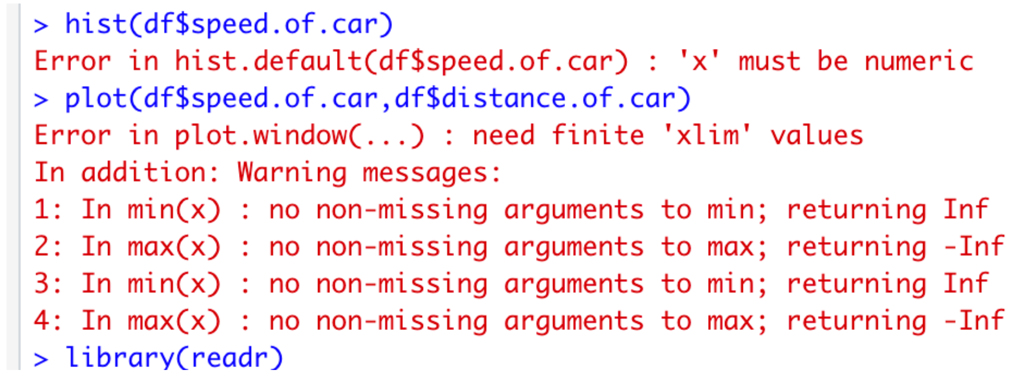
```
DMwR::regr.eval(actuals_preds$actuals,
actuals_preds$predicteds)

      mae        mse       rmse       mape
6.1398015 98.2094570  9.9100685  0.1476248
```



Name	Type	Value
distPred	double [15]	4.75 9.24 18.21 31.67 40.64 45.13 ...
3	double [1]	4.75458
5	double [1]	9.240257
9	double [1]	18.21161
17	double [1]	31.66864
24	double [1]	40.63999
28	double [1]	45.12567
30	double [1]	49.61135
32	double [1]	54.09702
33	double [1]	54.09702
35	double [1]	54.09702
39	double [1]	63.06837
43	double [1]	63.06837
45	double [1]	76.5254
46	double [1]	81.01108
50	double [1]	85.49676

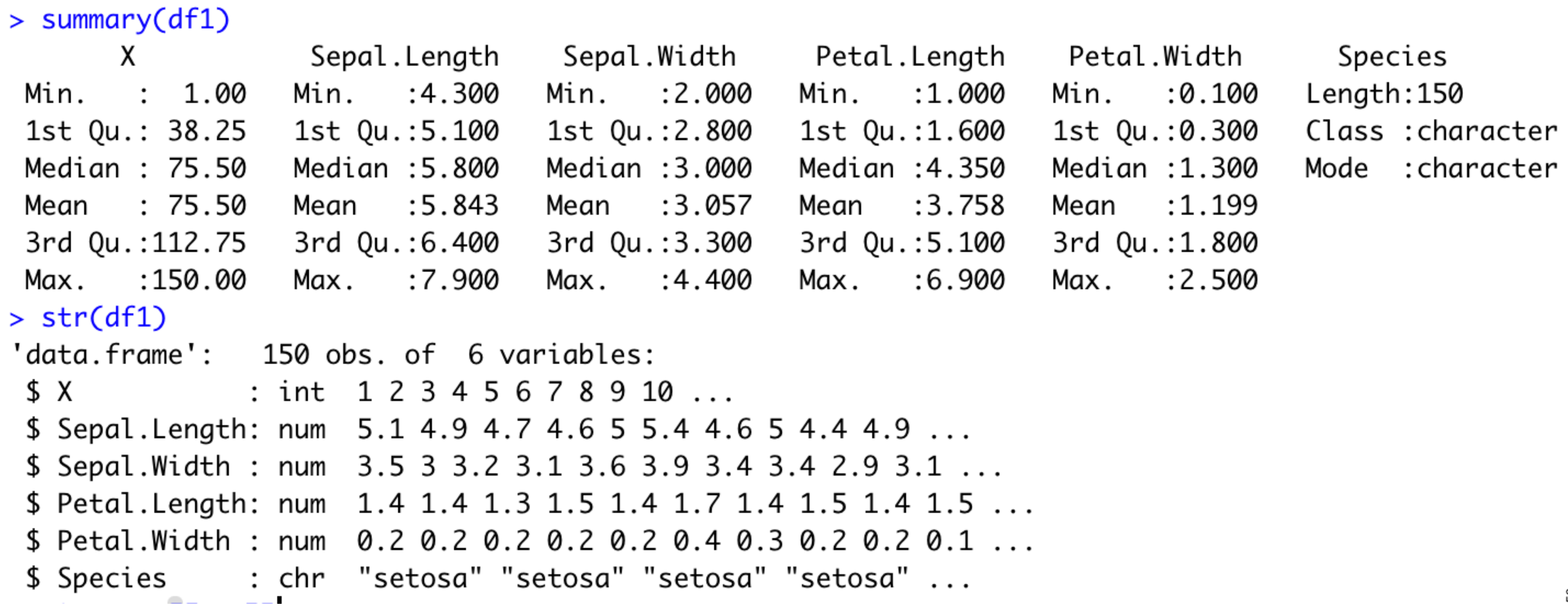
Errors Encountered – cars

- I had difficulties installing the package but I used the menu option to install and it was easier than commend line.
- I was not able to read the file even after I set the directory using setwd. So, I found a method to choose the file and then read the file using f <- file.choose().
- When I closed and open the Rstudio and was running a plot directly I had errors I thought it was recognizing the variables so I 'run' the R script from top and this error got fixed.



```
> hist(df$speed.of.car)
Error in hist.default(df$speed.of.car) : 'x' must be numeric
> plot(df$speed.of.car,df$distance.of.car)
Error in plot.window(...) : need finite 'xlim' values
In addition: Warning messages:
1: In min(x) : no non-missing arguments to min; returning Inf
2: In max(x) : no non-missing arguments to max; returning -Inf
3: In min(x) : no non-missing arguments to min; returning Inf
4: In max(x) : no non-missing arguments to max; returning -Inf
> library(readr)
```

Iris Data



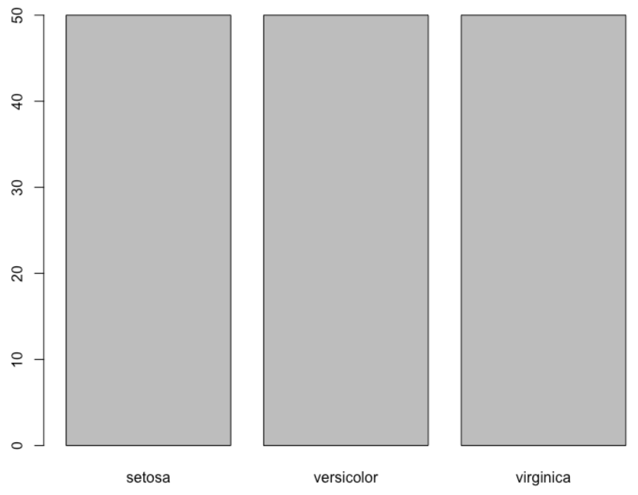
```
> summary(df1)
       X           Sepal.Length    Sepal.Width     Petal.Length    Petal.Width       Species         
 Min.   :  1.00   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Length:150        
 1st Qu.: 38.25   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Class :character  
 Median : 75.50   Median :5.800   Median :3.000   Median :4.350   Median :1.300   Mode  :character  
 Mean   : 75.50   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                     
 3rd Qu.:112.75   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                     
 Max.   :150.00   Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                     
> str(df1)
'data.frame':	150 obs. of  6 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : chr  "setosa" "setosa" "setosa" "setosa" ...
```

```
> names(df1)
[1] "X"            "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

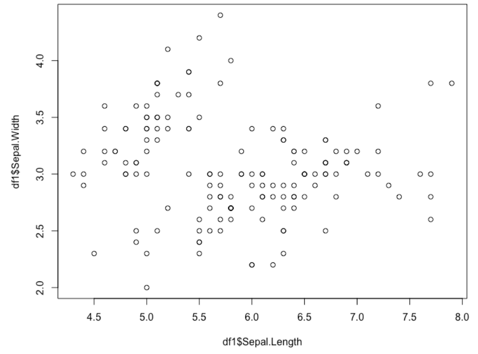
```
> hist(df1$Species)
Error in hist.default(df1$Species) : 'x' must be numeric
```

Used barplot as a solution

```
barplot(height=table(df1$Species))
```



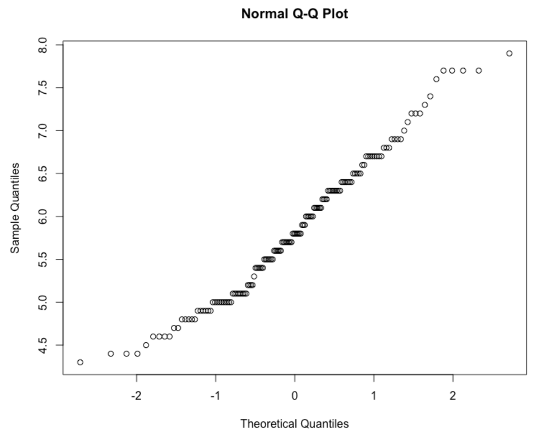
- plot(df1$Sepal.Length,df1$Sepal.Width)



- > qqnorm(df1) – include the column

Error in FUN(X[[i]], ...) :

only defined on a data frame with all numeric variables

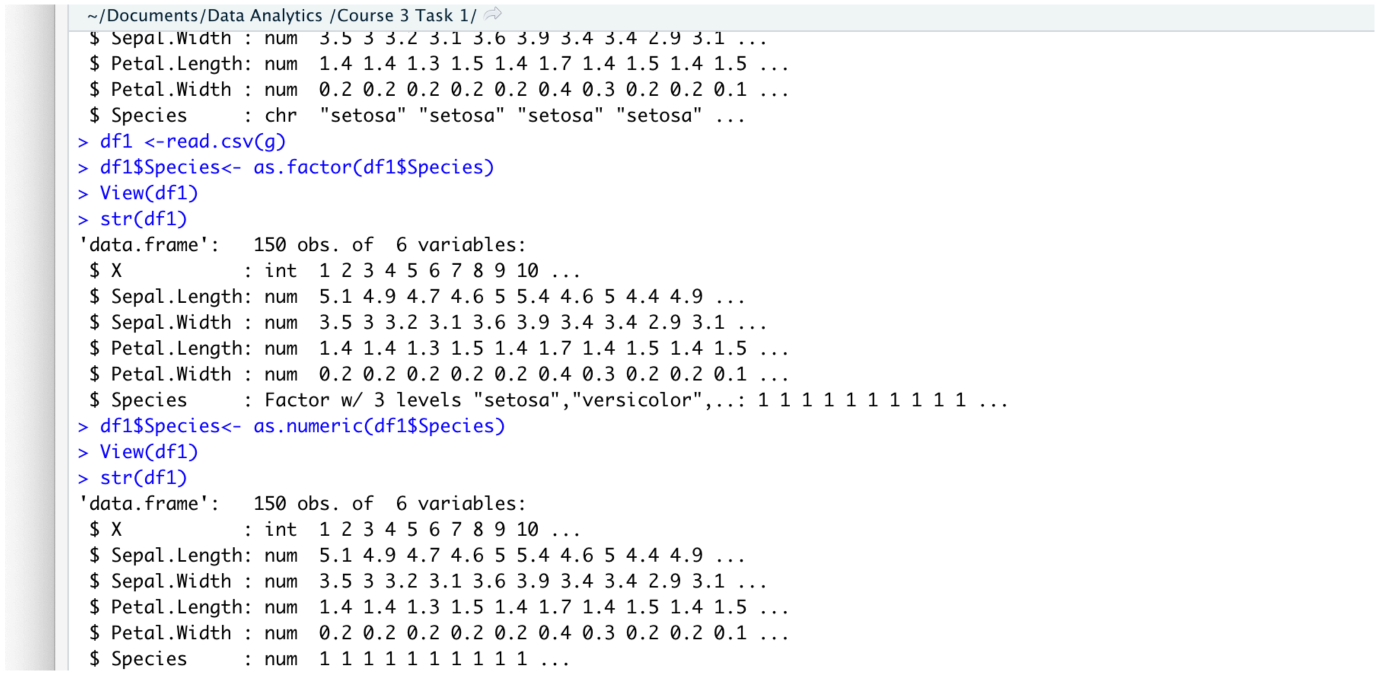


- > df1$Species<- as.numeric(df1$Species)

Warning message:

NAs introduced by coercion

Solution: Converted to factor and then to numeric



```
~/Documents/Data Analytics /Course 3 Task 1/
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : chr  "setosa" "setosa" "setosa" "setosa" ...
> df1 <-read.csv(g)
> df1$Species<- as.factor(df1$Species)
> View(df1)
> str(df1)
'data.frame':	150 obs. of  6 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> df1$Species<- as.numeric(df1$Species)
> View(df1)
> str(df1)
'data.frame':	150 obs. of  6 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : num  1 1 1 1 1 1 1 1 1 1 ...
```

- > trainSize <- round(nrow(df1) * 0.2)

```
> trainSize
[1] 30
```

- > testSize <- nrow(df1) - trainSet

```
Error: object 'trainSet' not found
```

Solution assign trainSet

```
trainingRowIndex1 <- sample(1:nrow(df1), 0.7*nrow(df1))  # row indices for training data
View(trainingRowIndex1)
trainSet <- df1[trainingRowIndex1, ]  # model training data
```

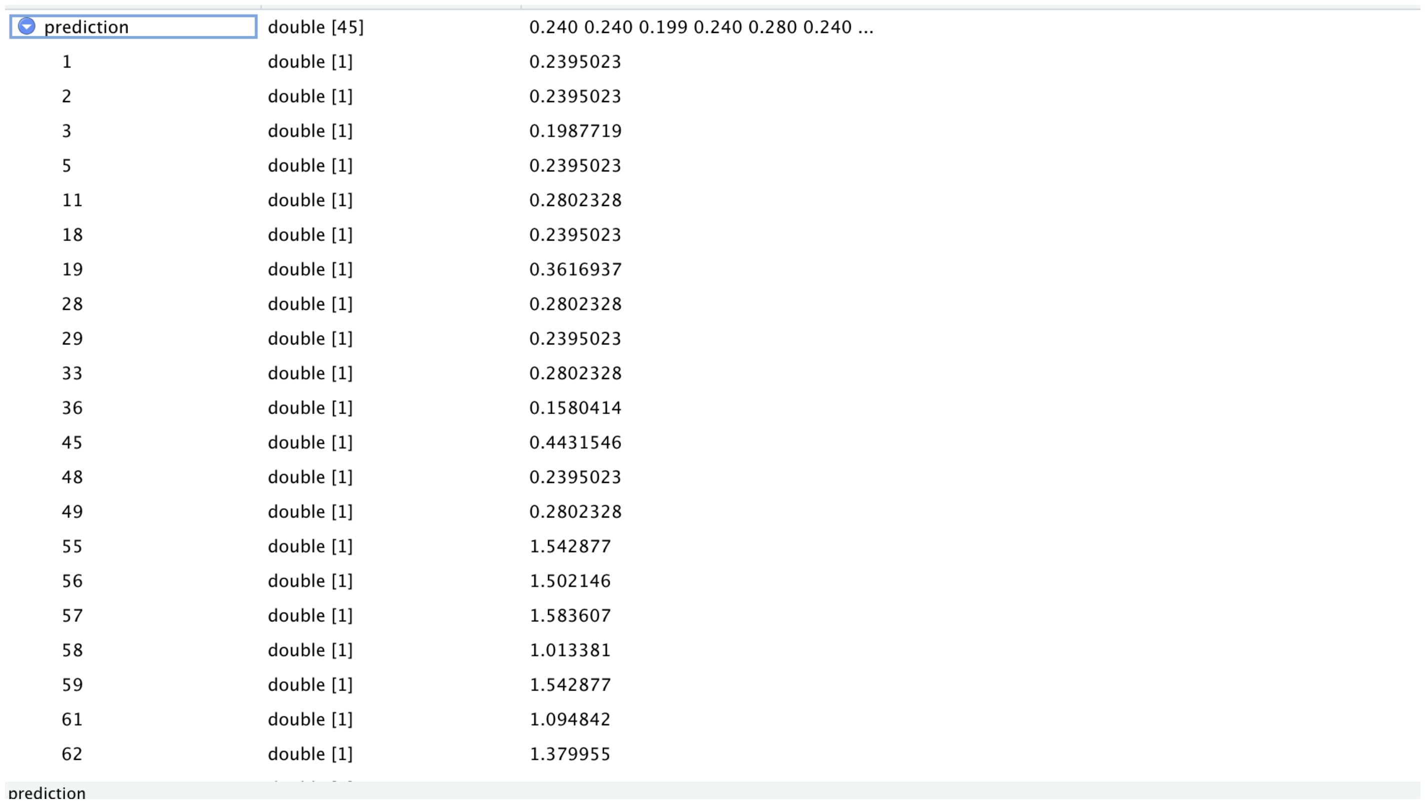
```
set.seed(405) # changing the set seed.
trainSet <- df1[trainingRowIndex1, ]  # model training data
View(trainSet)
testSet  <- df1[-trainingRowIndex1, ]   # test data
View(testSet)

> LinearModel<- lm(trainSet$Petal.Width ~ testSet$Petal.Length)
Error in model.frame.default(formula = trainSet$Petal.Width ~ 
testSet$Petal.Length,  :
  variable lengths differ (found for 'testSet$Petal.Length')
```

Solution- Two parameters missing one and send train set not test

```
LinearModel<- lm(Petal.Width ~ Petal.Length,trainSet)
```

```
> summary(LinearModel)

Call:
lm(formula = Petal.Width ~ Petal.Length, data = trainSet)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.43529 -0.13950 -0.00215  0.13493  0.65347 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -0.33072    0.04691   -7.05 2.11e-10 ***
Petal.Length  0.40730    0.01126   36.18  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2066 on 103 degrees of freedom
Multiple R-squared:  0.9271,	Adjusted R-squared:  0.9264 
F-statistic:  1309 on 1 and 103 DF,  p-value: < 2.2e-16
```

Prediction of the Petal width- Iris



prediction	double [45]	0.240 0.240 0.199 0.240 0.280 0.240 ...
1	double [1]	0.2395023
2	double [1]	0.2395023
3	double [1]	0.1987719
5	double [1]	0.2395023
11	double [1]	0.2802328
18	double [1]	0.2395023
19	double [1]	0.3616937
28	double [1]	0.2802328
29	double [1]	0.2395023
33	double [1]	0.2802328
36	double [1]	0.1580414
45	double [1]	0.4431546
48	double [1]	0.2395023
49	double [1]	0.2802328
55	double [1]	1.542877
56	double [1]	1.502146
57	double [1]	1.583607
58	double [1]	1.013381
59	double [1]	1.542877
61	double [1]	1.094842
62	double [1]	1.379955

prediction

Conclusion

- Was it straightforward to install R and RStudio?

Yes, very easy to install.

- Was the tutorial useful? Would you recommend it to others?

Definitely, I liked the search phrase help in each section for google search engine.

- What are the main lessons you've learned from this experience?

R is very similar to Python but easier and more visual.

- What recommendations would you give to other employees who need to get started using R and doing predictive analytics in R instead of Rapidminer?

I briefly used Rapidminer in the beginning of the course as part of intro which was visual but very complex when comes to data processing. R is very user friendly and has sophisticated packages. R seems more popular and powerful statistical tool for machine learning.