1808-351-351m-451-w01b-p-RIntro4-code.R

frenchrh

Thu Aug 30 10:29:21 2018

```
# Script Name: tea_time_class_1.R
# Purpose: Learning R
# Authors: Alan Curran, Roger French
# License: Creative Commons Attribution-NonCommercial-ShareAlike- 4.0 International License.
# Latest Changelog Entires: v0.00.01 - tea_time_class1.R - Ethan Started it

# Basics of R
# Math operations
  2 + 2
```

```
## [1] 4
```

```
  8 / 4
```

```
## [1] 2
```

```
  2 * 3
```

```
## [1] 6
```

```
  3 ^ 3
```

```
## [1] 27
```

```
  a = 3
  b = 3
  a + b
```

```
## [1] 6
```

```
# Assignment operator is usually used instead of =
# It is directional
# = works in most cases too but may cause problems

  a <- 3
  b <- 6

  a <- b
  a
```

```
## [1] 6
```

```
  a <- 3
  b <- 6

  a -> b
  a
```

```
## [1] 3
```

```
# Object classes
# numerics
  a <- 5
  class(a)
```

```
## [1] "numeric"
```

```
# integers
  b <- as.integer(42)
  class(b)
```

```
## [1] "integer"
```

```
# logicals
  c <- TRUE
  class(b)
```

```
## [1] "integer"
```

```
# characters
  d <- "hello world"
  class(c)
```

```
## [1] "logical"
```

```
# factors
  e <- as.factor(c("1", "1", "a", "1", "c", "a"))
  class(e)
```

```
## [1] "factor"
```

```
  e
```

```
## [1] 1 1 a 1 c a
## Levels: 1 a c
```

```
# quick note it you want to convert a factor to a numeric
# You have to convert it to a character first, then a numeric


# R uses brackets to reference a data index
# data["row","column"]
# Standard organization for data set has varaibles as columns and observeations as rows
# Keep in mind that R indexing starts from 1, not 0
# Load a test data set
  data("iris")

  iris[1,2]
```

```
## [1] 3.5
```

```
# Leaving a row or column input blank puts all values
# First column
  iris[,1]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
# First row
  iris[1,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```r
# Data frames have associated column names
  colnames(iris)
```

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

```r
# Columns can be called by name using $
# Rstudio features tab completion for thing like column names
  iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

```r
# Functions are processes that take an input and give an output
# Rstudio has tab completion for function inputs
  max(iris$Petal.Length)
```

```
## [1] 6.9
```

```r
  mean(iris$Sepal.Width)
```

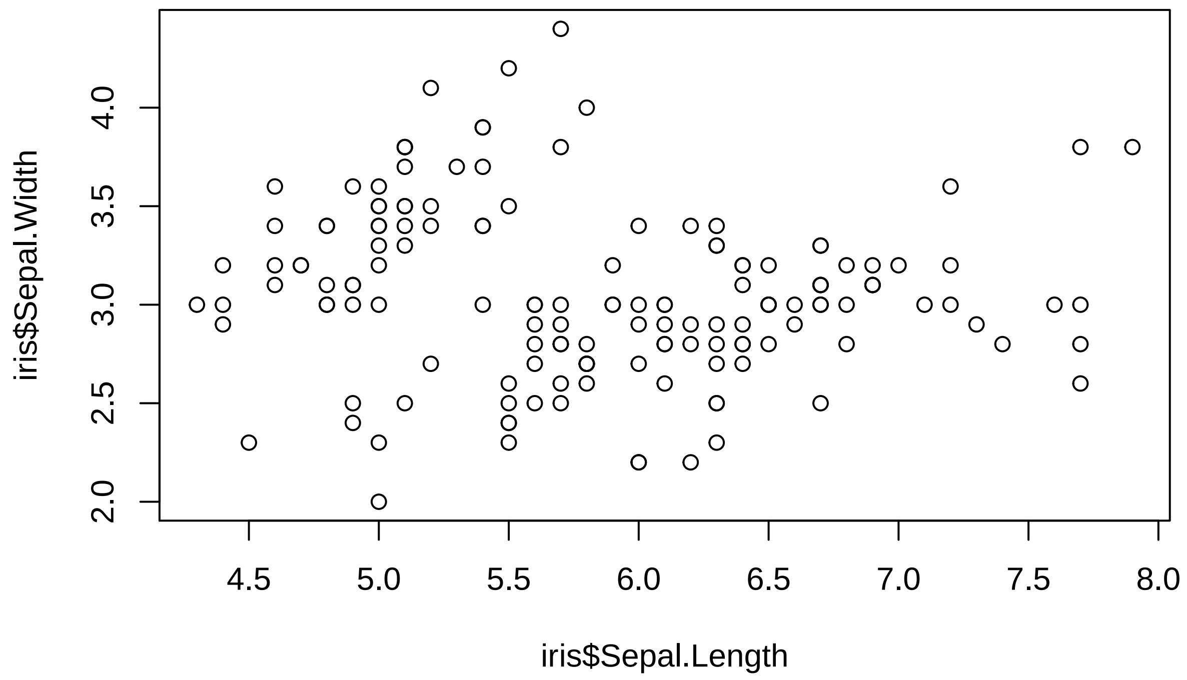
```
## [1] 3.057333
```

```r
  sd(iris$Petal.Width)
```

```
## [1] 0.7622377
```

```r
# Functions can take multiple inputs, they can be named in the call or placed in order
  plot(x = iris$Sepal.Length, y = iris$Sepal.Width)

# x and y can be specified with x = ... in any order or the inputs can be given in order
# This plot is the same as the previous
  plot(iris$Sepal.Length, iris$Sepal.Width)
```



```
# Matrix operations
  mat <- matrix(data = 1:9, nrow = 3, ncol = 3)
  mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
# Element multiplication
  mat*mat
```

```
##      [,1] [,2] [,3]
## [1,]    1   16   49
## [2,]    4   25   64
## [3,]    9   36   81
```

```
# Matrix multiplication
  mat %*% mat
```

```
##      [,1] [,2] [,3]
## [1,]   30   66  102
## [2,]   36   81  126
## [3,]   42   96  150
```

```
# t() function is for transposing
  t(mat)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
  mat %*% t(mat)
```

```
##      [,1] [,2] [,3]
## [1,]   66   78   90
## [2,]   78   93  108
## [3,]   90  108  126
```

```
# Inverse matrix
  mat[2,3] <- 18
  solve(mat)
```

```
##       [,1] [,2]        [,3]
## [1,] -1.05  0.1  0.61666667
## [2,]  0.60 -0.2 -0.06666667
## [3,] -0.05  0.1 -0.05000000
```

```
  solve(mat) %*% mat
```

```
##      [,1]          [,2]         [,3]
## [1,]    1  0.000000e+00 8.881784e-16
## [2,]    0  1.000000e+00 2.220446e-16
## [3,]    0 -5.551115e-17 1.000000e+00
```

```
# Structures in R
# for loops
  for (i in 1:5) {
      print(i)
  }
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
# While loops
  i <- 10
  while (i > 5) {
        i <- i - 1
        print(i)
  }
```

```
## [1] 9
## [1] 8
## [1] 7
## [1] 6
## [1] 5
```

```
# if statments
  dave <- TRUE

# if (dave) {} also works
  if (dave == TRUE) {
      print("good morning dave")
  }
```

```
## [1] "good morning dave"
```

```r
# User defined functions
  math <- function(a,b) {
    c <- a + b*2
    # return defines what the output of the function is
    return(c)
  }
  math(2,6)
```

```
## [1] 14
```