1808-351-351m-451-w01b-p-RIntro2-code.R

frenchrh

Thu Aug 30 10:14:14 2018

```
# Script Name: tea_time_class_1.R
# Purpose: Learning R
# Authors: Ethan Pickering, 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

# A pound symbol, or "Hashtag" indicates a comment in the code

# Assigining Operators

## PRO TIP ## - Control enter runs the line you are on or a highlighted section

# Assign number
  x <- 5

# Lets view it with function View()
  View(x)

# We can also look into what view is as a function and how it works
  ?View() # The question command shows us the help file
```

```
## starting httpd help server ... done
```

```
# Assign a vector or array of numbers, from 1 - 5
  x <- 1:5

# We can also use = instead of <- (supposeduly <- is better than =, not sure why)
# Assign a character string
  B <- "hello"

# Use print function to print to console
  print(B)
```

```
## [1] "hello"
```

```
  print(x)
```

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

```
# R Object and type of classes
# Characters, integers, numeric, complix, Logic (TRUE/FALSE)

# R Vectors
  a = c("hello","wolrd")
  A = c("hi", 5)
  c = c(1.23452, 2.1435, 3.14)

# The function c() here is "concatenate" so it will combine the values
  false = TRUE
```

```
# These will have attributes such as dimensions, object class, and length

# Lets check all of the attributes
  class(a)
```

```
## [1] "character"
```

```
  class(A)
```

```
## [1] "character"
```

```
  class(x)
```

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

```
  class(c)
```

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

```
  class(false)
```

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

```
  false
```

```
## [1] TRUE
```

```
# There are also factors - but we will dive into those later, they are categorical
# Oh and lists... probably the most dynamic object in R

# These classes are default by what R interprets them to be, but we can change them
  c = as.character(c)
  class(c)
```

```
## [1] "character"
```

```
  # Other class changing functions
  as.numeric(x)
```

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

```
  as.integer(x)
```

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

```
  as.POSIXct(1442866615,origin = "1970-01-01") # A Date Format
```

```
## [1] "2015-09-21 16:16:55 EDT"
```

```
  as.factor(x)
```

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

```
  as.list(x)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
```

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

```
  as.data.frame(x) # An incredibly useful object
```

```
##   x
## 1 1
## 2 2
## 3 3
## 4 4
## 5 5
```

```
# Lets check the lengths of the variables

  length(a)
```

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

```
  length(A)
```

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

```
  length(x)
```

```
## [1] 5
```

```
  length(c)
```

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

```
  length(false)
```

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

```
# Other nice functions to create objects, seq(), matrix(), vector(), array()
  m = seq(1, 24, by = 3)
  n = matrix(0, nrow = 3, ncol = 3)
  print(n)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
```

```
# Lets look at the dimensions
  dim(n)
```

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

```
# Lets also reset some of the variables
# The format for pulling values is [row, column]
  n[2,2] = 2
  print(n)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
```

```
## [2,]    0    2    0
## [3,]    0    0    0
```

```
# We can be even more efficient, say all of column 1 is 3
  n[,1] = 3
  print(n)
```

```
##      [,1] [,2] [,3]
## [1,]    3    0    0
## [2,]    3    2    0
## [3,]    3    0    0
```

```
# Next lets turn the matrix into a dataframe - this will allow multiple classes
  n = as.data.frame(n)
  n[1,2] = "character"
  n[,3] = TRUE
  print(n)
```

```
##   V1        V2   V3
## 1  3 character TRUE
## 2  3         2 TRUE
## 3  3         0 TRUE
```

```
  class(n)
```

```
## [1] "data.frame"
```

```
  class(n[,1])
```

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

```
  class(n[,2])
```

```
## [1] "character"
```

```
  class(n[,3])
```

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

```
# Another useful tool is to bind datasets with rbind() and cbind()
  a = 1:5
  b = 20:24
  rbind(a,b)
```

```
##   [,1] [,2] [,3] [,4] [,5]
## a    1    2    3    4    5
## b   20   21   22   23   24
```

```
  cbind(a,b)
```

```
##      a  b
## [1,] 1 20
## [2,] 2 21
## [3,] 3 22
## [4,] 4 23
## [5,] 5 24
```

```
  d = cbind(a,b)

# These must be of equal length along the dimension you wish to bind
# Sam rows/same columns
```

```r
# Other useful functions to mention, which(), mean(), sd(), min(), max()

  row = which(d[,1] == 2) # Note the two equal signs that are necessary to check for equivalence
  print(row)
```

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

```r
  mean(d)
```

```
## [1] 12.5
```

```r
  sd(d)
```

```
## [1] 10.12423
```

```r
  min(d)
```

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

```r
  max(d)
```

```
## [1] 24
```

```r
# For and IF statements, basic structure and example
#structure
  for (i in 1:5) {
      # How much it should loop, and the values
      # Input what you want it to do
  }

  if ( i == 5) {
     # Let it know what to check for
     # What you want to do if condition is met
  }
```

```
## NULL
```

```r
# Example
  length = length(d[,1])

  for (i in 1:length) {
      d[i,1] = 100

    if (i == 2) {
       d[i,1] = 200
    }

  }

  print(d)
```

```
##        a  b
## [1,] 100 20
## [2,] 200 21
## [3,] 100 22
## [4,] 100 23
## [5,] 100 24
```

```
# Writing and Reading data files

# Working directories
# Lets find where you are
  getwd()
```

```
## [1] "D:/Git/18f-dsci351-351m-451-prof/2-class"
```

```
# Now we can set the directory we want, you must have R point to the directory
# that you either want to save to or read from
#setwd("path")

# PRO TIP the path . gives you the current, and .. gives you the one before
# Say I need to go two folders back and up one to data, I would enter
# setwd("../../data)
# Or if data was the next folder in from where I was setwd("./data)

# Set the working directory where you want
# setwd("./class")


# Lets write a .csv
  write.csv(d, "d.csv")

# Now lets read it back in
  e = read.csv("d.csv")

# Notice there are now rownames in the object
# You can avoid this with

  write.csv(d, "d.csv", row.names = FALSE)
  e = read.csv("./d.csv")

# Other ways to read in data
  read.table("./d.csv", sep = ",")
```

```
##    V1 V2
## 1   a  b
## 2 100 20
## 3 200 21
## 4 100 22
## 5 100 23
## 6 100 24
```

```
#  read.delim("blah.txt", header = TRUE, sep = "\t") # For tab delimited files


# Packages
# One of the coolest things in R is that it is open sourced and you can
# download new analysis techniques from many authors on the fly and incorporate
# it into your work

# There are two steps, install and library
#  install.packages("ggplot2")
```

```r
library("ggplot2")
??ggplot2  # This allows us to check out the vignettes, or long form documentation
```