```
# ****************************************#
# ***Name: Class 1 Presentation Code*********#
# ***Author: R Tutor************************#
# ***Date: March 9, 2017********************#
# ***Require: baseballFull.csv***************#
# ****************************************#

#remove all existing objects from workspace
rm(list=ls())


####Introduction#####


  #Example Script
  1+1

  #create a very simple object, a, with numeric elements 1 and 2
  a <- c(1,2)
  a

  A

  #Setting Working Directory
  input <- "//AGBOSNAS1/DATA/SHARE/R/2017/Class 1/Code & Examples/Input"
  output <- "//AGBOSNAS1/DATA/SHARE/R/2017/Class 1/Code & Examples/Output"
  temp <- "//AGBOSNAS1/DATA/SHARE/R/2017/Class 1/Code & Examples/Temp"

  # Note the forward slash! Aargh!

  setwd(input)
  #assuming the "Code & Examples" folder has the
  #following subfolders: Input, Output, and Temp
  baseball <- read.csv("baseballFULL.csv", header=TRUE)

  #Switch to another working directory by invoking those objects
  setwd(output)
  write.csv(baseball, file="baseball_out.csv", row.names=FALSE)


  #Example graphics
  install.packages("rgl", dependencies=TRUE)
  library(rgl)
  library(MASS)

  #demo(rgl)
  #demo(abundance)

  demo(regression)

  demo(bivar)

####Data Types and Structure####


#### A. Scalars & Vectors ####
  #Vector Creation#
  pi
  is.vector(pi)

  #make object x with a vector containing elements 1,2,3,and 4
  x <- c(1,2,3,4)
  x
  typeof(x)
  is.numeric(x)
  is.double(x)
  length(x)

  #using the combine function to create a vector from other vectors
  x1 <- c(3,2,1)
  x2 <- c(4,5,6)
  x3 <- c(7,8,9)
  x4 <- c(x1,x2,x3)
  x4

  #examples of using vectors with special properties

  #a vector of consecutive integers
  y <- 1:4
  y

  #a vector with repeating values, using the rep() function
  x <- rep(1,10)
  x
```

```
  #a vector whose components are part of a sequence, using the seq() function
  ?seq
  x <- seq(1,20,2)
  x


  #vector with elements that are characters
  AG_Office <- c("Beijing", "Boston", "Chicago", "Dallas",
                 "Denver", "Los Angeles", "Menlo Park",
                 "Montreal", "New York", "San Francisco", "Washington")
  AG_Office

  #Indexing Elements of a vector; subvector <- Vector[index]#

    #a: Index vector of positive integers
      AG_Office[4]

      AG_Office[c(2,5)]


      #gives us the second to last element of the vector AG_Office -- Possible Question Time
      second_to_last <- AG_Office[length(AG_Office)-1]
      second_to_last


      #adding an office to the AG_office vector

      AG_Office <- c("Beijing", "Boston", "Chicago", "Dallas",
                     "Denver", "Los Angeles", "Menlo Park",
                     "Montreal", "New York", "San Francisco", "Washington",
                     "Brussels", "London")
      second_to_last <- AG_Office[length(AG_Office)-1]
      second_to_last

      #trying to use an index value greater than length(x)
      AG_Office[15]

    #b: index vector of negative integers; Subvector Returns all except 4th element
      #all of AG_Office except for the third element

      AG_Office_Ex_Dallas <- AG_Office[-4]
      AG_Office_Ex_Dallas

    #c: vector of character string/names
      #first, change names of positons in the vector;
      #attach alphanumeric names to vector
      #elements

      fruit_prices <- c(1.1,3.5,5.4,2.3)
      fruit_prices
      names(fruit_prices) <- c("orange", "banana", "apple", "peach")
      fruit_prices
      lunch <- fruit_prices[c("apple", "orange")]
      lunch

    #d: logical Vector -- conditional indexing
    #vector that evaluates to boolean; Boston = #1 "AG_Office" so 1st position evaluates to True
      AG_Office[AG_Office=="Boston"]

      is_Boston_Office <- AG_Office=="Boston"
      is_Boston_Office

      AG_Office[is_Boston_Office]

#### Vector Functions ####

  #functions are performed on vectors element by element -- like an internal loop
      a <- 1:10
      a
      # Question time: What will happen when we add 1 to a?
      a+1

  #summary statistics
  #Create a vector of 100 random normally distributed values with mean 0 and standard deviation 1

  #Question time: What should we feed this function? What does it output?
  ?rnorm

  x <- rnorm(n=100,mean=0,sd=1)
  length(x)

  summary(x)

  mean(x)
```

```
  sd(x)

  #arithmetic operations

  x <- c(1,2,3)
  y <- c(4,5,6)

  x+y


  #%in%
  x <- c(1,3,5)
  z <- c(3,5,7)

  x %in% z

  #sample error
  x <- c(1,2,3,4)
  y <- c(6,0,9,20,22,23)
  x+y

  #using NA
  #try to take the mean of a vector which contains a single NA value
  x <- c(100, NA, 200, 300, 400)
  x

  mean(x)

  mean(x, na.rm=TRUE)


#### B. Matrices ####
  #creation of a two-by-two matrix from a four-element vector
  X <- matrix(c(1,2,3,4), nrow=2,ncol=2)
  X

  #byrow argument in the matrix() function
  X <- matrix(c(1,2,3,4),nrow=2,byrow=TRUE)
  X
  is.matrix(X)

  typeof(X)
  length(X)
  nrow(X)
  ncol(X)

  #combine the same three vectors first using rbind() and cbind()
  x1 <- c(1,2,3)
  x2 <- x1+3
  x3 <- x1+4
  y <- rbind(x1,x2,x3)
  y

  y <- cbind(x1,x2,x3)
  y

  #combine vector x of numeric values with vector y of character values
  #into a matrix called M -- coercion
  x <- c(1,2,3)
  y <- c("one", "two", "three")
  M <- cbind(x,y)
  M

  M[,"x"]
  is.character(M[,"x"])

#### C. Lists ####

  #creating a list
  x <- c("one","two","three")
  x1 <- c(1,2,3)
  x2 <- x1+3
  x3 <- x1+4
  y <- rbind(x1,x2,x3)
  y

  L <- list(x,y)
  L
  is.list(L)

  typeof(L)


  #we can refer to list components by using one of the
  #four types of index vectors
```

```
  #requires double brackets instead of single ones
  L <- list(vector=x,matrix=y)
  L

  L[[1]]

  L[["vector"]]

  #refer to list components by their names using the $ sign
  L$vector

  L$vector[1]

  L$matrix

  L$matrix[,2]

  #unlist command -- Question Time: what data structure does it change to? what data type?
  unlist(L)

  #R saves regression results into a list with many components
  #regress salary on years, runs
  ?lm
  View(baseball)

  reg1 <- lm(salary~years+runs, data = baseball)
  reg1

  reg1_summary <- summary(reg1)
  reg1_summary

  is.list(reg1)
  is.list(reg1_summary)
  names(reg1_summary)

  #get the matrix of coefficients
  reg1_coef <- reg1_summary$coefficients
  reg1_coef
  is.matrix(reg1_summary$coefficients)


#### D. Data Frames ####
  #Basic Data Frame
   x1 <- c(1,2,3)
   x2 <- c("AB","BC","CD")
   D <- data.frame(x1,x2)
   D

  #Reading in Dataframes
    input <- "//AGBOSNAS1/DATA/SHARE/R/2016/Class 1/Code & Examples/Input"
    setwd(input)
    #Specialized command: read.csv()
    ?read.csv
    baseball <- read.csv("baseballFull.csv")

  #Saving / Reading an R data frame
    #save permanent R dataset called BaseballDataset
    temp <- "//agbosnas1/data/Share/R/2017/Class 1/Code & Examples/Temp"
    setwd(temp)
    save( baseball, file = "BaseballDataset.Rda" )

    #read in BaseballDataset
    load("BaseballDataset.Rda")

  #Accessing Data within Dataframes

  #Method A: Similarly to a vector/matrix
    #Access row 1, column 5
    baseball[1,5]

    #Access columns 2 and 3
    baseball[ , c(2,3)]

    #Access all rows except for row 2
    baseball[-c(2),]

  #Method B: Using dataframe properties
    #Find the names of the dataframe's variables
    names(baseball)
    colnames(baseball)
    rownames(baseball)

    #Once we know the names, we can index datasets using the name
    baseball[,"team"]
```

```
    #...or we can use the $ syntax
    baseball$team

    #Tab example
      #if multiple possible variables, will provide a list
      baseball$h

      #if only one possible variable, will auto fill
      baseball$n

      #go to console with function example

  #Creating variables in a Dataframe
    baseball$BA <- baseball$hits/baseball$atbat
    baseball$BA

  #creating new variables with conditions based on existing variables
  #will be covered in Class 2

  #Renaming variables
    #create a small example data frame
    Var1 <- c(1,2,3,4)
    Var2 <- c("A","B","C","D")
    Example_Data <- data.frame(Var1,Var2)

    #rename variables in data frame
    colnames(Example_Data) <- c("Numbers","Letters")
    Example_Data

    #load reshape package (Install in advance)
    library("reshape")

    #view syntax for rename
    rename
    #rename the "Numbers" variable
    Example_Data <- rename(Example_Data, c(Numbers="Numbers_new"))
    Example_Data

  #Subsetting a Dataframe
    #Method A: Index Syntax
      #Select all obs whose team is BOS
      baseballBOS <- baseball[baseball$team == "BOS", ]

      which( baseball$team == "BOS" )

      baseballBOS <- baseball[which( baseball$team == "BOS" ), ]

      #Select all obs whose team is not BOS
      baseballNoBOS <- baseball[-which( baseball$team == "BOS" ), ]
      baseballNoBOS2 <- baseball[which( !(baseball$team == "BOS") ), ]

      #%in%:
      baseballALEast <- baseball[which( baseball$team %in% c("BOS", "NYA",
                                                              "BAL", "TOR", "TAM") ),]

    #Method B: Subset function

      #subset to observations whose team is BOS
      baseballBOS <- subset( baseball, baseball$team == "BOS"  )

      #subset to observations whose team is BOS and only keep name, team, position, and hits variables
      baseballBOS2 <- subset( baseball, baseball$team == "BOS", c(name, team, position, hits) )



    #variable dropping using subset
    baseball_sub <- subset(baseball, select=-c(name, salary))


#### Factors ####
  # Are there factor variables in our data?
  str(baseball)
  levels(baseball$league)

  #Creating a factor from a numeric vector
  x <- c(1991,1992,1996,1993,1994,1995,1995,1993)
  xf <- factor(x)
  is.vector(xf)
  x
  xf
  attributes(xf)

  #importance of factors: used very often in categorical analysis
  #summary statistics by group: aggregate(), by(), tapply() covered in Class 2
  #categorical analyses: ANOVA, Chi-Square Test
```

```
#Factors v Character Variables
  #create vectors - one character, and one factor
  character_vector <- c("BIRD","MOLE","FISH","MYNOCK", "BIRD")
  factor_vector <- as.factor(c("SKY","EARTH","SEA","SPACE", "SKY"))

  character_vector
  factor_vector

  #attempt to create matrix with these two vectors
  matrix <- cbind(character_vector,factor_vector)
  matrix

  #to combine as a matrix while keeping the desired strings
  matrix2 <- cbind(character_vector,factor_vector=as.character(factor_vector))
  matrix2
```