title: "DS311 - Basic R Lab Exercise" author: "Your Name" date: "8/23/2022" output: pdf_document: default html_document: default

subtitle: R Lab Exercise

```{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)

```
 # OUTPUT
> knitr::opts_chunk$set(echo = TRUE)

Basic R Exercise

Section 1 - Data Type

Key Functions
- typeof()
- as.numeric()
- as.charater()

Numeric

```{r}
# Numeric - Double precision by default

n1 <- 15
n1
typeof(n1)

n2 <- 1.5
n2
typeof(n2)
```

OUTPUT

> n1 <- 15
> n1 [1] 15 typeof(n1)

n2 <- 1.5

> typeof(n1) [1] "double"
>
> n2 <- 1.5

Character

```{r}

# Character

c1 <- "c" c1 typeof(c1)
```

c2 <- "a string of text" c2 typeof(c2)

```
 # OUTPUT
>
> c1 <- "c"
c1
typeof(c1)
> c1
[1] "c"
> typeof(c1)
[1] "character"
>
> c2 <- "a string of text"
> c2
[1] "a string of text"
> typeof(c2)
[1] "character"


### Logical

```{r}
# Logical

l1 <- TRUE
l1
typeof(l1)

l2 <- F
l2
typeof(l2)
```

OUTPUT

> l1 <- TRUE l1 l1 [1] TRUE typeof(l1) [1] "logical"

Transforming Numerics and Characters

``` {r}

# Transforming numeric into characters

num <- 10 numToChar <- as.character(num) paste("num Type: ", typeof(num), " | numToChar: ", typeof(numToChar))

# Transforming characters into numeric

char <- "10" charToNum <- as.numeric(char) paste("char Type: ", typeof(char), " | charToNum: ", typeof(charToNum))

```
 # OUTPUT
> num <- 10
> numToChar <- as.character(num)
> paste("num Type: ", typeof(num), " | numToChar: ", typeof(numToChar))
[1] "num Type: double | numToChar: character"

>
> char <- "10"
charToNum <- as.numeric(char)
paste("char Type: ", typeof(char), " | charToNum: ", typeof(charToNum))
> charToNum <- as.numeric(char)
> paste("char Type: ", typeof(char), " | charToNum: ", typeof(charToNum))
[1] "char Type: character | charToNum: double"

Challenge:

Complete the following tasks:

``` {r}
# Check the data type of the following variables
a <- as.integer(500)
b <- as.double(500)
c <- as.character(500)

# Enter your code here!
paste("a: ", a, " | TYPE: ", typeof(a))
paste("b: ", b, " | TYPE: ", typeof(b))
paste("c: ", c, " | TYPE: ", typeof(c))


# Check the data type of the following variable
d <- a / b

# Enter your code here!
paste("d: ", d, " | TYPE: ", typeof(d))
```

OUTPUT

> a <- as.integer(500) b <- as.double(500) b <- as.double(500) c <- as.character(500)
>
> paste("a: ", a, " | TYPE: ", typeof(a)) [1] "a: 500 | TYPE: integer" paste("b: ", b, " | TYPE: ", typeof(b)) paste("c: ", c, " | TYPE: ", typeof(c))
>
> paste("b: ", b, " | TYPE: ", typeof(b)) [1] "b: 500 | TYPE: double" paste("c: ", c, " | TYPE: ", typeof(c))

d <- a / b

> paste("c: ", c, " | TYPE: ", typeof(c)) [1] "c: 500 | TYPE: character"

d <- a / b
```

paste("d: ", d, " | TYPE: ", typeof(d))

> d <- a / b
>
> paste("d: ", d, " | TYPE: ", typeof(d)) [1] "d: 1 | TYPE: double"

Section 2 - Data Structure

- is.vector()
- is.matrix
- cbind()
- as.data.frame()

Vector

```{r}

# Vector

v1 <- c(1, 2, 3, 4, 5) v1 is.vector(v1)

v2 <- c("a", "b", "c") v2 is.vector(v2)

v3 <- c(TRUE, TRUE, FALSE, FALSE, TRUE) v3 is.vector(v3)
```

```
 # OUTPUT
>
> v1 <- c(1, 2, 3, 4, 5)
v1
> v1
[1] 1 2 3 4 5
> is.vector(v1)
[1] TRUE

v2 <- c("a", "b", "c")
>
> v2 <- c("a", "b", "c")
> v2
[1] "a" "b" "c"
> is.vector(v2)
[1] TRUE

### Matrix

```{r}
# Matrix

m1 <- matrix(c(T, T, F, F, T, F), nrow = 2)
m1
is.matrix(m1)

m2 <- matrix(c("a", "b",
               "c", "d"),
               nrow = 2,
               byrow = T)
m2
is.matrix(m2)
```

OUTPUT

> m1 <- matrix(c(T, T, F, F, T, F), nrow = 2) m1 [,1] [,2] [,3] [1,] TRUE FALSE TRUE [2,] TRUE FALSE FALSE is.matrix(m1) [1] TRUE
>
> m2 <- matrix(c("a", "b", + "c", "d"), + nrow = 2, + byrow = T) m2 [,1] [,2] [1,] "a" "b" [2,] "c" "d" is.matrix(m2) [1] TRUE

Challenge:

1. Create a vector of the 26 alphabet lower case letters in sequence.
2. Create a 2 by 13 matrix for the 26 English upper case letter in sequence.

Hint: Check out the "letters" and "LETTERS" key words in R.

```{r}

# Enter your code here.

matrix1 <- matrix() is.matrix(matrix1) vector <- c() chr <- function(n) { rawToChar(as.raw(n)) } for (x in 97:122) vector <-

append(vector, chr(x)) print(vector) matrix1<-matrix(toupper(vector),nrow=2, byrow = T) print(matrix1)

```
 # OUTPUT
> matrix1 <- matrix()
is.matrix(matrix1)
vector <- c()
chr <- function(n) { rawToChar(as.raw(n)) }
for (x in 97:122)
  vector <- append(vector, chr(x))
print(vector)
matrix1<-matrix(toupper(vector),nrow=2, byrow = T)
print(matrix1)
> is.matrix(matrix1)
[1] TRUE
> vector <- c()
> chr <- function(n) { rawToChar(as.raw(n)) }
> for (x in 97:122)
+   vector <- append(vector, chr(x))
> print(vector)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
> matrix1<-matrix(toupper(vector),nrow=2, byrow = T)
> print(matrix1)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"   "K"   "L"   "M"
[2,] "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W"   "X"   "Y"   "Z"

DataFrame

```{r}
# Data Frame

# Can combine vectors of the same length
vNumeric   <- c(1, 2, 3)
vCharacter <- c("a", "b", "c")
vLogical   <- c(T, F, T)

df1 <- cbind(vNumeric, vCharacter, vLogical)
df1  # Coerces all values to most basic data type

df2 <- as.data.frame(cbind(vNumeric, vCharacter, vLogical))
df2  # Makes a data frame with three different data types
```

OUTPUT

Data Frame

vNumeric <- c(1, 2, 3) vCharacter <- c("a", "b", "c") vLogical <- c(T, F, T)
```

> df1 <- cbind(vNumeric, vCharacter, vLogical) df1 # Coerces all values to most basic data type vNumeric vCharacter vLogical [1,] "1" "a" "TRUE"
> [2,] "2" "b" "FALSE" [3,] "3" "c" "TRUE"

df2 <- as.data.frame(cbind(vNumeric, vCharacter, vLogical)) df2 # Makes a data frame with three different data types

> df2 <- as.data.frame(cbind(vNumeric, vCharacter, vLogical)) df2 # Makes a data frame with three different data types vNumeric vCharacter vLogical 1 1 a TRUE 2 2 b FALSE 3 3 c TRUE

Section 3 - Setup Working Directory and Installing Packages

Key Functions: - getwd() - setwd() - install.packages() - library()

Setting up your working directory

``` {r}

# Check your current working directory

initial <- "/Users/robert/sfsu/ds311" # Just to be sure that I will go back in my initial directory setwd(initial) wd1 <- getwd() paste("Current Working Directory: ", wd1)

# Setting the working directory for a project

p <- paste(wd1, "/Hello-World", sep="") setwd(p) wd2 <- getwd() paste("Current Working Directory: ", wd2) setwd(initial)
```

```
 # OUTPUT
> initial <- "/Users/robert/sfsu/ds311"    # Just to be sure that I will go back in my initial directo
> setwd(initial)
> wd1 <- getwd()
> paste("Current Working Directory: ", wd1)
[1] "Current Working Directory:  /Users/robert/sfsu/ds311"
>
> p <- paste(wd1, "/Hello-World", sep="")
> setwd(p)
> wd2 <- getwd()
> paste("Current Working Directory: ", wd2)
[1] "Current Working Directory:  /Users/robert/sfsu/ds311/Hello-World"
> setwd(initial)

### Installing and Loading Packages

```{r, include=FALSE}
# Install a new package, note the quotation marks
install.packages("mass")

# Install multiple packages at once
install.packages(c("dplyr", "ggplot2"))

# Loading the package, note no quotation marks
library(dplyr)

# Checking the package version
packageVersion("dplyr")

# List all functions in a package
ls("package:ggplot2")

# Loading a function from package
ggplot2::geom_line

# Update all packages
update.packages()

# Unload a package
detach(package:ggplot2, unload=TRUE)

# Help function
help(dplyr)

# Checking the session info
sessionInfo()
```

OUTPUT
```

> install.packages("mass") --- Please select a CRAN mirror for use in this session ---

install.packages(c("dplyr", "ggplot2"))

library(dplyr)

packageVersion("dplyr") ls("package:g Secure CRAN mirrors

1: 0-Cloud [https] 2: Australia (Canberra) [https] 3: Australia (Melbourne 1) [https] 4: Australia (Melbourne 2) [https] 5: Australia (Perth) [https] 6: Austria [https] 7: Belgium (Brussels) [https] 8: Brazil (PR) [https] 9: Brazil (RJ) [https] 10: Brazil (SP 1) [https] 11: Brazil (SP 2) [https] 12: Bulgaria [https] 13: Canada (MB) [https] 14: Canada (ON 3) [https] 15: Chile (Santiago) [https] 16: China (Beijing 2) [https] 17: China (Beijing 3) [https] 18: China (Hefei) [https] 19: China (Hong Kong) [https] 20: China (Guangzhou) [https] 21: China (Lanzhou) [https] 22: China (Nanjing) [https] 23: China (Shanghai 2) [https] 24: China (Shenzhen) [https] 25: Colombia (Cali) [https] 26: Costa Rica [https] 27: Cyprus [https] 28: Czech Republic [https] 29: Denmark [https] 30: East Asia [https] 31: Ecuador (Cuenca) [https] 32: Ecuador (Quito) [https] 33: France (Lyon 1) [https] 34: France (Lyon 2) [https] 35: France (Marseille) [https] 36: France (Paris 1) [https] 37: Germany (Erlangen) [https] 38: Germany (Leipzig) [https] 39: Germany (Göttingen) [https] 40: Germany (Münster) [https] 41: Germany (Regensburg) [https] 42: Greece [https] 43: Hungary [https] 44: Iceland [https] 45: India [https] 46: Indonesia (Banda Aceh) [https] 47: Iran (Mashhad) [https] 48: Italy (Milano) [https] 49: Italy (Padua) [https] 50: Japan (Tokyo) [https] 51: Japan (Yonezawa) [https] 52: Korea (Gyeongsan-si) [https] 53: Korea (Seoul 1) [https] 54: Korea (Ulsan) [https] 55: Malaysia [https] 56: Mexico (Mexico City) [https] 57: Mexico (Texcoco) [https] 58: Morocco [https] 59: Netherlands (Dronten) [https] 60: New Zealand [https] 61: Norway [https] 62: South Africa (Johannesburg) [https] 63: Spain (A Coruña) [https] 64: Spain (Madrid) [https] 65: Sweden (Borås) [https] 66: Sweden (Umeå) [https] 67: Switzerland (Zurich 1) [https] 68: Taiwan (Taipei) [https] 69: Turkey (Denizli) [https] 70: Turkey (Istanbul) [https] 71: Turkey (Mersin) [https] 72: UK (Bristol) [https] 73: UK (London 1) [https] 74: USA (IA) [https] 75: USA (MI) [https] 76: USA (MO) [https] 77: USA (OH) [https] 78: USA (OR) [https] 79: USA (TN) [https] 80: USA (TX 1) [https] 81: Uruguay [https] 82: (other mirrors)

Selection: Enter an item from the menu, or 0 to exit Selection: # Install multiple packages at once Enter an item from the menu, or 0 to exit Selection: install.packages(c("dplyr", "ggplot2")) Enter an item from the menu, or 0 to exit Selection: Enter an item from the menu, or 0 to exit Selection: # Loading the package, note no quotation marks Enter an item from the menu, or 0 to exit Selection: library(dplyr) Enter an item from the menu, or 0 to exit Selection: Enter an item from the menu, or 0 to exit Selection: # Checking the package version Enter an item from the menu, or 0 to exit Selection: packageVersion("dplyr") Enter an item from the menu, or 0 to exit Selection: Enter an item from the menu, or 0 to exit Selection: # List all functions in a package Enter an item from the menu, or 0 to exit Selection: ls("package:g Enter an item from the menu, or 0 to exit Selection:

Section 4 - Problem Solving

Write the code that accomplish the following tasks:

Part a: Assign 4 to variable x

Part b: Assign 12 to variable y

Part c: Print both x and y to check their values

Part d: Divide y by x and assign it to variable z

part e: Print a statement to report your answer in Part d.

Once you finished and knit the RMarkdown file into html file, you should be able to see the message "Congratulation!! You completed the first exercise in this section!!" in the html document.

```{r}

# Write your code here!

# Part a

x <- 4

# Part b
```

y <- 12

Part c

print(paste(x, sep = " ", y))

Part d

z <- y/x

Part e

print(paste("y divided by x is equal to ", z))

Do not need to change the following code!

if (exists("x") == TRUE | exists("y") == TRUE | exists("z") == TRUE){ if (x == 4 & y == 12 & z == 3) { print("Congratulation!! You completed the first activity in this class!!") } else { print("Sorry, you got it wrong!") } } else { print("You did not complete the last problem!") } ```

OUTPUT

> x <- 4
>
> y <- 12
>
> print(paste(x, sep = " ", y)) [1] "4 12"
>
> z <- y/x
>
> print(paste("y divided by x is equal to ", z)) [1] "y divided by x is equal to 3"
>
> if (exists("x") == TRUE | exists("y") == TRUE | exists("z") == TRUE){ + if (x == 4 & y == 12 & z == 3) { + print("Congratulation!! You completed the first activity in this class!!") + } else { + print("Sorry, you got it wrong!") + } + } else { + print("You did not complete the last problem!") + } [1] "Congratulation!! You completed the first activity in this class!!"