~~~
---
title: "Lab2"
author: "SAF"
date: "`r Sys.Date()`"
output: pdf_document
---
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

# Part I: Introduction to modeling using basic R syntax
.*Congratulations, you have puppy fever! As someone who is inflicted with puppy
fever, you would like to buy as many puppies as you perceivable can. To help
you determine if you can afford all of the puppies you want or to determine
how many puppies you can afford, you opt to use R to help you. In this
exercise, please print the contents of each variable after you declare it.*

**Integer:** *puppies* variable holds the number of puppies you’d like to have.
```{R Part 1a, echo=FALSE}
#I'd like to have 3 puppies.
puppies <- 2L
cat("I'd like to buy", puppies,"puppies.")
```

**Numeric:** *puppy_price* variable holds the price of a single puppy.
```{R Part 1b, echo=FALSE}
#Price of each puppy is $500.
puppy_price <- 500.00
cat("The price of each puppy is $", puppies*puppy_price,".")
```

**Integer:** *max_puppies* variable holds Qty of puppies you can afford for $1,000.
```{R Part 1c, echo=FALSE}
max_puppies <- (1000/puppy_price)
cat("I can afford", max_puppies, "puppies.")
```

**Numeric:** *total_cost* variable holds the total price of all puppies.
```{R Part 1d, echo=FALSE}
#Total price of all puppies I'd like to buy.
total_cost <- (puppies*puppy_price)
cat("It's $", total_cost,"for all the puppies i want.")
```

**Logical-Bool** *too_expensive* Variables return TRUE if the cost is greater than
$1,000.
```{R Part 1e, echo=FALSE}
BoolResult <- total_cost > 1000L
cat("Is the cost of the puppies I want greater than $1000?", BoolResult)
```
~~~

~~~
# Part II: Manipulating variables and learning how to use new functions

*You work as a data analyst for a new company and are asked to create id tags for
everyone at work. Your goal is to make it informative as well as personal to help
facilitate collaboration in the work place.To do this, you first want to gather
information about each employee.*

**2a.Character** *my_name* This assigns your name to the variable.
Assign *my_name* to a variable in (4) different ways.
```{R Part 2a, echo=FALSE}
#Load variable with my name
my_name <- "Sal Figueroa"

message("1. Using print() function:")
print(my_name)
message("2. Using message() function:")
message(my_name)
message("3. Using cat() function:")
cat(my_name, "\n")
message("4. Using paste() function:")
paste(my_name)
```



**2b.Character** *favorite_day* holds your favorite day of the week.
```{R Part 2b, echo=FALSE}
favorite_day <- "Thursday"
cat("My favorite day is", favorite_day)
```



**2c.Integer** *my_height* Assigns your height in whole inche values.
```{R Part 2c, echo=FALSE}
my_height <- 71L
cat("My height is",my_height,"inche/s")
```



**2d.Characters** *favorite_quote* Holds your favorite quote.
```{R Part 2d, echo=FALSE}
favorite_quote <- "Whiskey is for drinking; water is for fighting over."
count <- nchar(favorite_quote)
cat("My favorite qoute is about
    water,", favorite_quote, "It has",count,"characters.")
```



**2e.Type of Data Objects** Verify the data types of *my_name*, *my_height*,
*favorite_day*, and *favorite_quote*
~~~

~~~~
```{R Part 2e, echo=FALSE}

cat("my_name variable type is a", typeof(my_name))

cat("\nfavorite_day variable type is a", typeof(favorite_day))

cat("\nmy_height variable type is an", typeof(my_height))

cat("\nfavorite_qoute variable type is a", typeof(favorite_quote))

```



**2f.Coerce these variables** to a *numeric* and describe what happens.
```{R Part 2f, echo=FALSE}
message("The height variable doesn't generate an error message. The remaining
variables
        display an (NA) and --Warning: NAs introduced by coercion\n")
cat(" my_name as a numeric:", as.numeric(my_name),"\n")
cat(" favorite_day as numeric:", as.numeric(favorite_day),"\n")
cat("my_height as numeric:", as.numeric(my_height),"\n")
cat(" favorite_qoute as numeric:", as.numeric(favorite_quote))
```

**2g.Create of Vector** named "id" that contains *my_name, my_height,
favorite_day,* and *favorite_quote*.
```{R Part 2g, echo=FALSE}
#creates a vector named id with char variables.
id <- c(my_name, my_height, favorite_day, favorite_quote)
cat("id vector: ",id)
```

**2h.Class of Vector** What class is "id"? Did the classes change for the variables
themselves?
```{R Part 2h, echo=FALSE}
cat("\n The vector id is a type of:", typeof(id),"\n\n")
message("\n Below are the data types for each variable. Verfiy if variables weren't
altered")
cat("\n my_name type:", typeof(my_name),"\n")
cat("\n favorite_day type:", typeof(favorite_day),"\n")
cat("\n my_height type:", typeof(my_height),"\n")
cat("\n favorite_qoute type:", typeof(favorite_quote),"\n")
```

**2i.Employee's Information** Your employer wants you to be able to print each
employee’s id while displaying each variable of information line by line. As a
beginner with R, however, you are unfamiliar with how to do this so your employer
gives you a hint to use the functions cat and paste.
```{R Part 2i, echo=FALSE}
cat("cat function(id):\n")
~~~~

```
cat(id, sep=", ")
message("\n\n paste function(id):")
paste(id, sep=", ")
cat("\n f(g(x))-Using paste function:\n")
cat(paste(id), sep=", ","\n")
message("\n g(f(x))-Using cat function:")
paste(cat(id), sep=", ")
```
Try using cat and paste with id as a function argument.
How do the results differ?
The cat() function prints out the vector in a row with (,) including for separation.
While the paste function() places each vector variable in a separate row with a [i] bracket numbering each line

What happens when we use cat and paste at the same time--(i.e. f(g(x)))?
Using cat(paste(id)) configuration it produces a similar output as cat(id).
The sep="," doesn't seem to function properly, note the last coma position

What happens if we change the order we use them--(i.e. g(f(x)))?
Rearranging the functions we get the same out put with a character(0) error at the end.
The sep="," spacing modifier doesn't seem function either.

**2j.Difference between cat and paste **
How would you determine the difference between
cat and paste using R documentation (from within RStudio)?
This could be accomplished by adding a (?) to the function to bring up the R documentation for each function
example: ?paste

What is a great internet resource to use as discussed in the book? RStudipo
https://forum.posit.co/
https://www.rdocumentation.org/

What do sep and collapse arguments for paste do?
They control the spacing between each string fragment (word). Where collapse removes all spaces and the Sep command allows the user to dictate the char that will be used to speratarte the string sements.

If we wanted to append each character variable in our vector id with a new line (i.e. \\n) would we use sep or collapse?
To add a new line (\\n) to each character variable in a vector, you would first add the new line character to each element using paste() and then collapse them into a single string
Appending each character var in the vector with a new line would be best using the collapse argument within the paste() function.

2k. cat and paste Display the contents of the id function using a combination

~~~
of cat and paste with the appropriate arguments for paste.
```{R Part 2 tmp, echo=FALSE}
#Loads the variable with id vector that has paste, length, sep function acting on
it.
#1:length(id) sets the numbering limit using the vector as a final length limit.
numbered_list <- paste(1:length(id), id, sep = ". ")
cat(numbered_list, sep = "\n")
```


# Part III: Accessing data in GitHub and mastering order of operations

*Please calculate a (using min), b (using max), and c (using mean) from data. Pay
careful attention to follow order of operations (PEMDAS)*
```{R Part 3, echo=FALSE}
#Loads data from upstream repository provided
load("W:/CSIT165_RProgramming/_repos/CSIT165-ModData/Module-4/data.RData")
#loads default vector name to Mod4Data vector
Mod4Data <- data
#Loads the min, max, mean values from Mod4Data vector into their respective
variables.
a <- min(Mod4Data)
b <- max(Mod4Data)
c <- mean(Mod4Data)
#Displays the values for min, max, mean
print(paste("The min value is:", a))
print(paste("The max value is:", b))
print(paste("The mean value is:", c))
#Equations that solves for both root values.
Xplus <- (-b+sqrt((b^2)-(4*a*c)))/(2*a)
Xminus <- (-b-sqrt((b^2)-(4*a*c)))/(2*a)
#Displays, prints both roots.
print(paste("The X+ root is:", Xplus))
print(paste("The X- root is:", Xminus))
```
~~~