**Calculation Questions (using R) - Use R to calculate the following:**

**NOTE: Submit a word file with the R codes in text format and its screenshot that shows the result.**

* 1. **1. (20 points) Basic Math – each question has 5% credits a. The log of a positive number.**

1. # Part a - Function to compute the log of a positive number
2. log(99)
3. log(99, base = exp(1))
4. > log(99)
5. [1] 4.59512
6. > log(99, base = exp(1))
7. [1] 4.59512
   1. **b. What is the default base for the log function? Calculate the log of your previous number with a different base.**
8. # Part b - Understanding default base for the log function
9. # After referring to help documentation, looks like the
10. # the function computes the natural logarithm by default.
11. log10(99)
    * 1. > log10(99)
      2. [1] 1.995635
      3. > log(99, base = 10)
      4. [1] 1.995635
    1. c. The log of a negative number. (explain the answer)
12. # Part c - the log of a negative number
13. # The natural logarithm function ln(x) is defined only for
14. # positive numbers x > 0. The natural logarithm of a negative
15. # number is undefined which is why R returns "NaNs produced"
16. # suggesting that there is a missing value or "Not a Number"
17. log(-99)

> log(-99)

[1] NaN

Warning message:

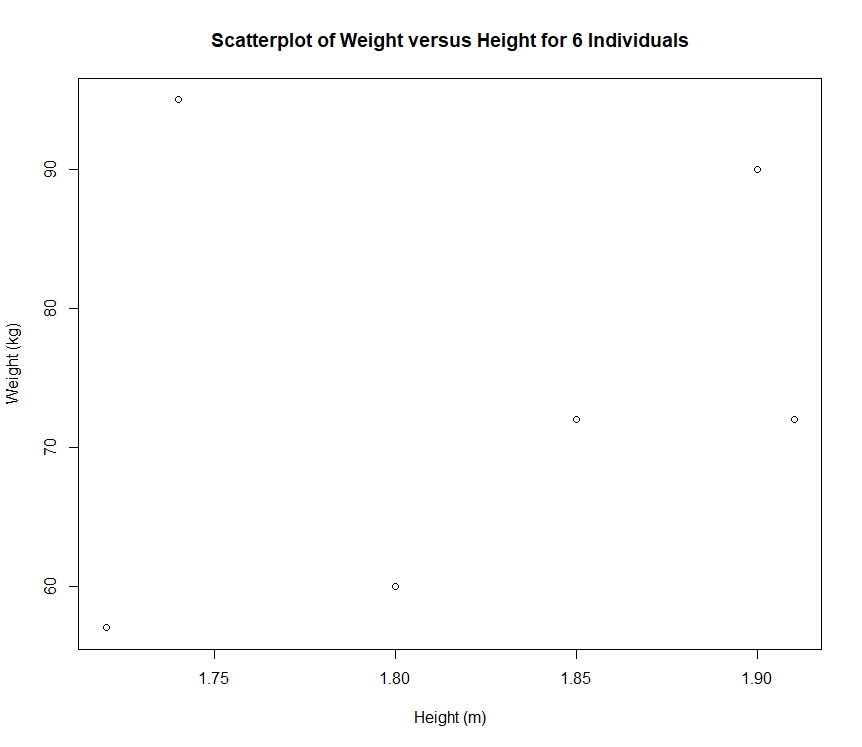
In log(-99) : NaNs produced

* 1. **d. The square-root of a positive number.**

1. # Part d - the square-root of a positive number
2. sqrt(16)
3. > sqrt(16)
4. [1] 4
   1. **2. (15 points) Random number generation.**
   2. **a. Create a vector of 15 standard normal random variables. Calculate its mean and SD (Standard Deviation).**
5. # Question 2 - Random Number Generation
6. # Part a - Create a vector of 15 standard normal random
7. # variables calculate its mean and SD (Standard Deviation)
8. Vector <- rnorm(15)
9. Vector
10. mean(Vector)
11. sd(Vector)
12. > Vector <- rnorm(15)
13. > Vector
14. [1] -1.201459089 -0.863258210 0.705352243 -0.853599322 -0.003320467 -0.259417411
15. [7] -0.026945561 -0.404018528 0.663917169 -0.690213247 -1.209137380 1.219899275
16. [13] -1.017265909 2.472775268 -0.131907535
17. > mean(Vector)
18. [1] -0.1065732
19. > sd(Vector)
20. [1] 1.01805
21. >
    1. **b. Change the mean to 10 and the SD to 2 and recalculate the vector of 15 random normal variables. Calculate its mean and SD.**
22. # Part b - Change the mean to 10 and the SD to 2 to recalculate
23. # the vector of 15 random normal variables. Calculate its mean
24. # and SD
25. Vector <- rnorm(15, m = 10, sd = 2)
26. Vector
27. mean(Vector)
28. sd(Vector)
29. > # Part b - Change the mean to 10 and the SD to 2 to recalculate
30. > # the vector of 15 random normal variables. Calculate its mean
31. > # and SD
32. > Vector <- rnorm(15, m = 10, sd = 2)
33. > Vector
34. [1] 12.757686 11.004395 10.734429 13.200425 12.100082 10.076014 9.364236 9.589241
35. [9] 7.468947 12.395423 10.792038 14.449243 11.299861 11.932961 9.007611
36. > mean(Vector)
37. [1] 11.07817
38. > sd(Vector)
39. [1] 1.811198
    1. **c. Why are the means and SD not exactly the same as the means and SDs specified in the function?**
40. # the rnorm function generates a sample of random numbers from a
41. # normal distribution with the specified mean and standard
42. # deviation properties input into the function by the user.
43. # This means that every time the rnorm function is called upon
44. # to generatre 15 numbers with a mean of 10 and standard deviation
45. # of 2, it simply grabs 15 numbers at random from a normal
46. # distribution with the specified properties (mean and sd) and will
47. # be different every time the function is called upon. The random
48. # numbers are being generated from the same population with a normal
49. # distribution, however, each sample of random numbers will have
50. # variation in their mean and standard deviation.
    1. **3. (40 points) Vector Operations a. The weights of 6 individuals in kg are 60, 72, 57, 90, 95, 72.**

**b. Their heights (in m) are 1.80, 1.85, 1.72, 1.90, 1.74, 1.91.**

1. **c. Enter these vectors into R.**
2. # Question 3 - Vector Operations
3. # Part a -> c - Create 2 vectors for heights and weights of 6 individuals
4. weights <- c(60, 72, 57, 90, 95, 72)
5. weights
6. heights <- c(1.80, 1.85, 1.72, 1.90, 1.74, 1.91)
7. heights
8. > weights <- c(60, 72, 57, 90, 95, 72)
9. > weights
10. [1] 60 72 57 90 95 72
11. > heights <- c(1.80, 1.85, 1.72, 1.90, 1.74, 1.91)
12. > heights
13. [1] 1.80 1.85 1.72 1.90 1.74 1.91
14. **d. Create a scatterplot of weight vs. height. Interpret the scatterplot.**



# Part d - Create a scatterplot of weight vs. height

plot(heights,

weights,

main = "Scatterplot of Weight versus Height for 6 Individuals",

xlab = "Height (m)",

ylab = "Weight (kg)")

# Interpret the scatterplot

# Scatterplot is showing a slight positive correlation, estimated at

# r ~ 0.7, however, there is one indivdual with a height of 1.72 meters but

# a very high corresponding weight of 95 kilograms resulting in a large

# residual if a linear regression model had been fit onto the plot. Another

# point to note is the tall individual with a height of 1.91 meters and a

# rather low weight of 72 kilograms that would result in high leverage on

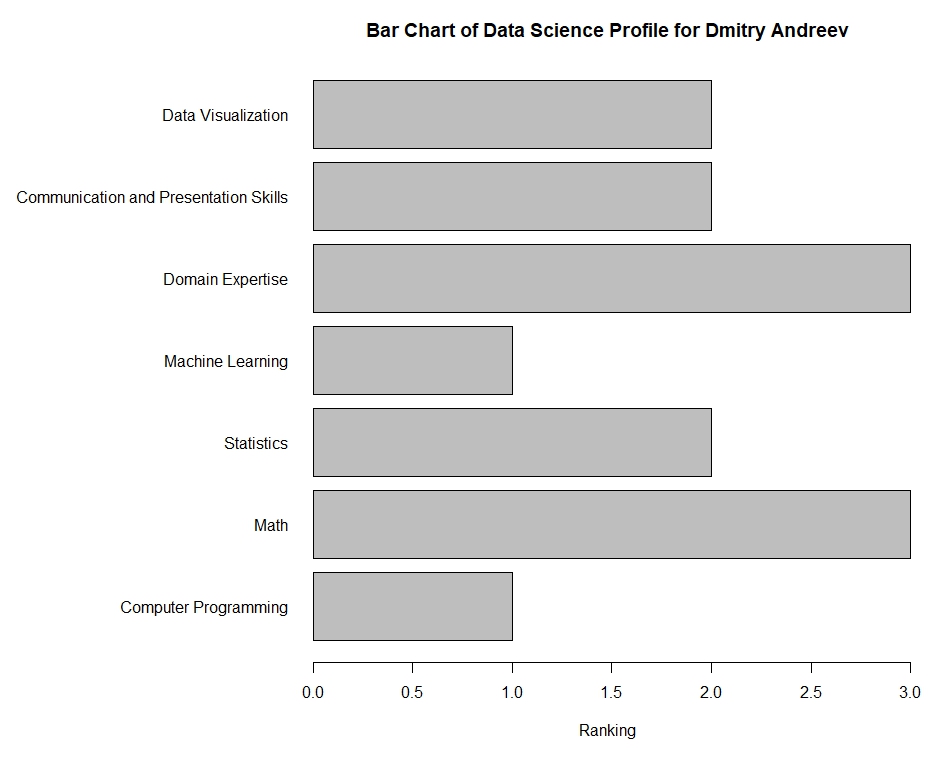
# a linear regression model.

1. **e. Calculate the BMI for each individual. (BMI = weight in kg divided by the square of the height in m)**
2. # Part e - Calculate the BMI for each individual
3. # BMI = weight in kg divided by the square of the height in m
4. BMI <- weights / (heights ^ 2)
5. BMI
6. > BMI <- weights / (heights ^ 2)
7. > BMI
8. [1] 18.51852 21.03725 19.26717 24.93075 31.37799 19.73630
9. **f. Calculate the mean for weight.**
10. # Part f - Calculate the mean for weight
11. mean\_weight <- mean(weights)
12. mean\_weight
13. > mean\_weight <- mean(weights)
14. > mean\_weight
15. [1] 74.33333
16. **g. Subtract the mean from each value of weight.**
17. # Part g - Subtract the mean for each value of weight
18. residual\_weights <- weights - mean\_weight
19. residual\_weights
20. > residual\_weights <- weights - mean\_weight
21. > residual\_weights
22. [1] -14.333333 -2.333333 -17.333333 15.666667 20.666667 -2.333333
23. **h. Sum the result. Now you know why we square the deviations from the mean to calculate a standard deviation!**
24. # part h - Sum the result
25. # Now you know why we square the deviations from the mean to calculate a
26. # standard deviation!
27. sum\_of\_residual\_weights <- sum(residual\_weights)
28. sum\_of\_residual\_weights

> sum\_of\_residual\_weights <- sum(residual\_weights)

> sum\_of\_residual\_weights

[1] 2.842171e-14

1. **4. (25 points) Your data science profile. Enter your data science profile into R as a data frame with two columns. Call it by your first name. The categories are computer programming, math, statistics, machine learning, domain expertise, communication and presentation skills, and data visualization. Your ranking for each category 1 5, with 5 as best. Create a bar graph of your data science profile. When you submit your work, please submit your code, including the data entry piece.**
2. # Question 4 - Enter your data science profile in R as a data frame
3. # Data frame consists of two columns - data science categories, ranking
4. # Categories - Computer Programming, math, statistics, machine learning, domain
5. # expertise, communication and presentation skills, and data visualization
6. # Ranking - 1 as worst, 5 as best for best ranking for each category
7. # Create a bargraph of your data science profile
8. Data\_Science\_Categories <- c("Computer Programming",
9. "Math",
10. "Statistics",
11. "Machine Learning",
12. "Domain Expertise",
13. "Communication and Presentation Skills",
14. "Data Visualization")
15. Ranking <- c(1, 3, 2, 1, 3, 2, 2)
16. Dmitry <- data.frame(Data\_Science\_Categories, Ranking)
17. # fitting labels
18. par(las = 1) # make lable text perpendicular to axis
19. par(mar = c(5, 16, 3, 1)) # increase y-axis margin
20. barplot(Dmitry$Ranking,
21. names.arg = Dmitry$Data\_Science\_Categories,
22. horiz = TRUE,
23. xlab = "Ranking",
24. main = "Bar Chart of Data Science Profile for Dmitry Andreev")
25. 
26. **5. (50 points) Install the swirl package and complete modules 1-7 (see the document RSwirlUnit01.docx for instructions). Please upload an R script (or txt file) containing your code from modules 1-7.**

install.packages("swirl")

library("swirl")

install\_from\_swirl("R Programming")

swirl()

# Lesson 1 - Basic Building Blocks

5 + 7

x <- 5 + 7

x

y <- x - 3

y

z <- c(1.1, 9, 3.14)

?c

z

c(z, 555, z)

z \* 2 + 100

my\_sqrt <- sqrt(z - 1)

my\_sqrt

my\_div <- z / my\_sqrt

my\_div

c(1, 2, 3, 4) + c(0, 10)

c(1, 2, 3, 4) + c(0, 10, 100)

z \* 2 + 1000

my\_div

# Lesson 2 - Workspace and Files

getwd()

ls()

x <- 9

ls()

list.files()

dir()

?list.files

args(list.files())

old.dir <- getwd()

dir.create("testdir")

setwd("testdir")

file.create("mytest.R")

list.files()

file.exists("mytest.R")

file.info("mytest.R")

file.rename("mytest.R", "mytest2.R")

file.copy("mytest2.R", "mytest3.R")

file.path("mytest3.R")

file.path("folder1", "folder2")

?dir.create

dir.create(file.path("testdir2", "testdir3"), recursive = TRUE)

setwd(old.dir)

# Lesson 3 - Sequences of Numbers

1:20

pi:10

15:1

?`:`

seq(1, 20)

seq(0, 10, by = 0.5)

my\_seq <- seq(5, 10, length = 30)

length(my\_seq)

1:length(my\_seq)

seq(along.with = my\_seq)

seq\_along(my\_seq)

rep(0, times = 40)

rep(c(0, 1, 2), times = 10)

rep(c(0, 1, 2), each = 10)

# Lesson 4 - Vectors

num\_vect <- c(0.5, 55, -10, 6)

tf <- num\_vect < 1

2

tf

num\_vect >= 6

1

1

2

my\_char <- c("My", "name", "is")

my\_char

paste(my\_char, collapse = " ")

my\_name <- c(my\_char, "Dmitry")

my\_name

paste(my\_name, collapse = " ")

paste("Hello", "world!", sep = " ")

paste(1:3, c("X", "Y", "Z"), sep = "")

paste(LETTERS, 1:4, sep = "-")

# Lesson 5 - Missing Values

x <- c(44, NA, 5, NA)

x \* 3

y <- rnorm(1000)

z <- rep(NA, 1000)

my\_data <- sample(c(y, z), 100)

my\_na <- is.na(my\_data)

my\_na

my\_data == NA

sum(my\_na)

my\_data

0/0

Inf - Inf

# Lesson 6 - Subsetting Vectors

x

x[1:10]

2

x[is.na(x)]

y <- x[!is.na(x)]

y

3

y[y > 0]

x[x > 0]

x[!is.na(x) & x > 0]

x[c(3, 5, 7)]

x[0]

x[3000]

x[c(-2, -10)]

x[-c(2, 10)]

vect <- c(foo = 11, bar = 2, norf = NA)

vect

names(vect)

vect2 <- c(11, 2, NA)

names(vect2) <- c("foo", "bar", "norf")

identical(vect, vect2)

3

vect["bar"]

vect[c("foo", "bar")]

# Lesson 7 - Matrices and Data Frames

my\_vector <- 1:20

my\_vector

dim(my\_vector)

length(my\_vector)

dim(my\_vector) <- c(4, 5)

dim(my\_vector)

attributes(my\_vector)

my\_vector

class(my\_vector)

my\_matrix <- my\_vector

?matrix

my\_matrix2 <- matrix(1:20, nrow = 4, ncol = 5)

identical(my\_matrix, my\_matrix2)

patients <- c("Bill", "Gina", "Kelly", "Sean")

cbind(patients, my\_matrix)

my\_data <- data.frame(patients, my\_matrix)

my\_data

class(my\_data)

cnames <- c("patient", "age", "weight", "bp", "rating", "test")

colnames(my\_data) <- cnames