Student id: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

COMP 120: Test 1 (Part B)

(Semester 2, 2020)

# Overview

The goal of this test is to evaluate your ability to:

1. perform basic programming tasks in R;
2. develop and use small functions in R;
3. perform basic plotting in R using ggplot2;
4. perform basic type conversion in R;
5. read and write data to and from R using data frames (tibbles); and
6. perform data manipulation operations

This part of the test is worth 25% of your final mark. There are four questions, with some containing several sub-parts. **You must answer all the questions.** The marks for each question are indicated beside the question. If you need clarification about a question, you are welcome to consult with staff during the test on Zoom. Note that these questions should not be about asking for hints or tips. You must complete the test in 90 minutes. You must submit test-1.R file by the deadline (5pm on Friday, 21st August).

## Documents and Instructions

For this test, you have been given *six* documents:

1. The instructions and questions file (this document you are reading) which is 5 pages long.
2. The following *five* files are available on Blackboard in Course Documents -> Tests -> Test 1 (Part B – Programming Questions) folder.
   1. An R file called test-1.R is the file in which you will be recording your answers (code). Write your code in the placeholders for each question in this file.
   2. A data file called scales.csv that contains data and you will need to load and perform data manipulation for question 3.
   3. Two data files called scales-cleaned.csv and scales-modified.csv that may be *optionally* used for question 3. Another data file called gapminder\_data.csv may be *optionally* used for question 4.

If you haven’t done so already, before you start answering the questions, save the above-mentioned files in an appropriate folder under COMP120 folder (e.g., *practical\_test\_1* folder) and make this folder your working directory. Also, remember to set the .libPaths() correctly!

Keep saving your test-1.R file at regular intervals of time so that you do not lose your work.

(QUESTIONS START IN THE NEXT PAGE - PLEASE TURN OVER)

# Questions

# 

1. You are given the following four lines of code. These four lines create four variables. These are *cat\_names*, *cat\_ages, cat\_genders* and *cat\_owners* that contain names of five cats, their ages in years, their genders and their owner names respectively.

cat\_names <- c("Fuzzy", "Bubbly", "Daisy", "Aldo", "Jasper")

cat\_ages <- c(7, 5, 4, 9, 2)

cat\_genders <- c("Male", "Female", "Female", "Male", "Male")

cat\_owners <- c("Jay", "Nathan", "Ruth", "Nathan", "Sandra")

Answer the following questions based on the variables defined above.

1. Using a *single command* print the average age of the cats. [0.5 mark]
2. Using appropriate indexing approach, print the names of all cat owners who own male cats in a *single command*. Use appropriate variables out of the four variables given. [1.5 mark]
3. Using appropriate indexing approach, print the name(s) of all female cats that are 4 years or older in a *single command*. Use appropriate variables out of the four variables given.

[2 marks]

1. Create a function called compute\_BMI that takes two arguments height and weight, and returns a vector containing the Body Mass Index (BMI) value that is computed based on the two arguments. The BMI value is computed using the formula given below: [2 marks]

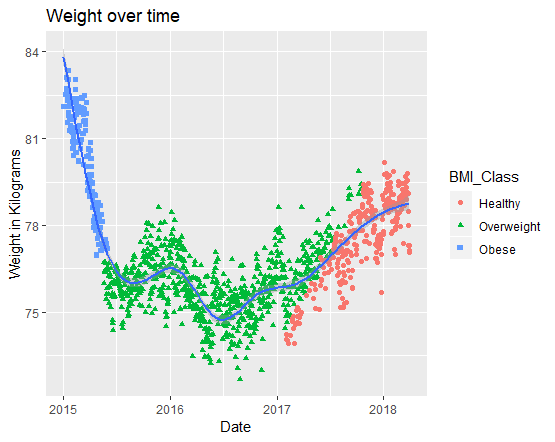
Test this function by calling: compute\_BMI(1.75, 75). Here 1.75 and 75 are values for height (in metres) and weight (in kilograms) respectively. The function should return the value: 24.4898. Also, test the function by calling compute\_BMI(c(1.75, 1.85), c(75, 85)). It should return the following values: 24.48980 and 24.83565.

1. Download the provided file *scales.csv* from Blackboard and inspect the file. The file has three columns: Date, Weight (in kilograms) and Height (in centimetres). These are the readings for a teenager from January 2015 to March 2018. Now, perform the following tasks (a to g below).
2. Read the *scales.csv* file into a tibble (a special type of data frame) called *growth\_tbl* using an appropriate function. When you read the data from the file make sure to look out for any missing values, and replace them with NA values. [1 mark]
3. The first column of *growth\_tbl*, the Date column should have a date data type, and not a character data type. Perform appropriate conversion to make the Date column into an actual date data type. If for some reason you are unable to replace missing values with NAs (in step above), you can load the data from the file named *scales-cleaned.csv* into *growth\_tbl*. Then, you should convert the data type of the Date column as described above. [1 mark]
4. The Height column is in centimetres. Convert this column to metres. Hint: 1 metre = 100 centimetres. [1 mark]

1. As indicated before, the contents of the file are the readings for a teenager. The teenager’s height and weight changed as they grew. Now, add a new column to your tibble (*growth\_tbl*) called BMI. The formula for BMI was given as a part of question 1. Use the function you created as a solution to question 1 to create this new column. If you have been unsuccessful in creating the function, you can use the formula directly. [2 marks]
2. Using the cut() function, create a new column called BMI\_Class in your tibble (*growth\_tbl*) to reflect the BMI classification given below: [2 marks]

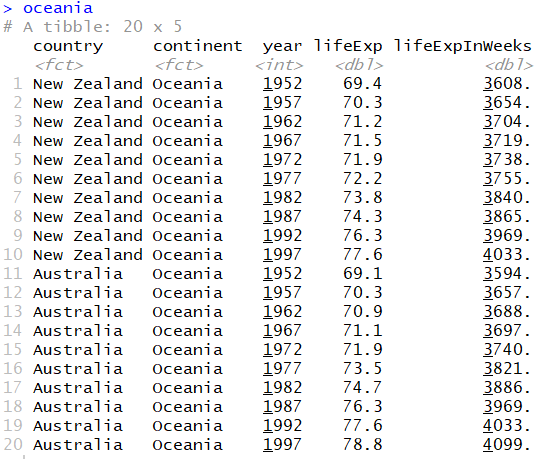
|  |  |  |
| --- | --- | --- |
| **BMI >** | **BMI** ≤ | **Classification** |
| 0 | 17 | Underweight |
| 17 | 24 | Healthy |
| 24 | 29 | Overweight |
| 29 | 39 | Obese |
| 39 | ∞ | Extremely Obese |

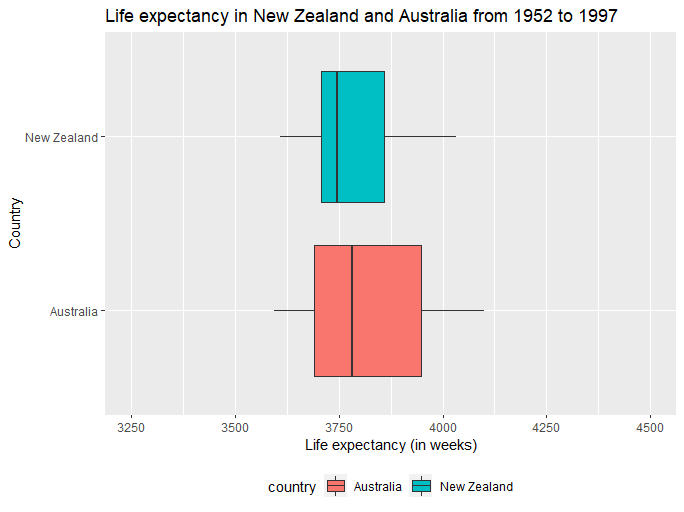
1. In *growth\_tbl*, retain only those rows that do not contain NA values in the Weight column. [1 mark]
2. Use ggplot and the updated tibble (growth\_tbl) to create the plot given below. If you think your growth\_tbl may not be accurate you can use scales-modified.csv which contains the correct columns and values (incorporating steps detailed above). If you use scales-modified.csv, make sure that you load the data into growth\_tbl and also convert the Date column into a date data type (if required). Note: the look and feel of the graph might be slightly different to what is given below (e.g. ordering of colours) if you load the data directly from the file and that is OK. [3 marks]



1. Write code to install the package (or library) called *gapminder* and load the package. Inspect the columns in the loaded dataset by typing *gapminder*. If you are unable to install and load the package, load the data provided in *gapminder\_data.csv[[1]](#footnote-1)*. This dataset curated by Hans Rosling depicting health and wealth of countries contains 1704 rows and 6 columns. For this question, you are expected to write code that produces the dataset shown in the snapshot given below and to produce a visualisation shown on the next page. You are expected to use appropriate *data manipulation functions* and *plotting functions* (i.e., ggplot()).

The columns of interest to us that exist in the gapminder dataset are: *country*, *continent*, *year* and *lifeExp* (the first four columns shown in the snapshot below). These columns represent country name, the “continent” the country belongs to, the year of data collection, and life expectancy of people in the country for the given year (in years). The *lifeExpInWeeks* column (which stands for life expectancy in weeks) doesn’t exist in the dataset. It must be created based on multiplying *lifeExp* column by 52 (the number of weeks in a year). Having done that, create a subset of data for countries in the Oceania “continent” where the value for the year column is before the year 2000. The data should also be sorted based on the name of the country (pay attention to the alphabetical order in the snapshot shown below). Store the resultant data in a tibble called *oceania* (see snapshot below). When printed it should produce the result shown in snapshot given below. Note: you must use the pipe operator to chain commands together for producing the dataset. Then, using the newly created *oceania* dataset write code to produce the visualisation shown on the next page. [8 marks]





**IMPORTANT: After completing the test, upload test-1.R on Blackboard!**

1. If you load the data from the CSV file into R, the *country* and *continent* columns will load as character columns and that is OK. You don’t have to convert them into factors. [↑](#footnote-ref-1)