## Question 3 [50 points]

The goal of this question is to write functions to help to simplify our exploratory data analyses. The data for this question will come from the **palmerpenguins** library that we have used before in quizzes and assignments.

```r
library(palmerpenguins) # make sure it is installed first
data(penguins)
head(penguins)
```

```
# A tibble: 6 x 8
  species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
  <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
1 Adelie  Torge~           39.1          18.7              181        3750 male
2 Adelie  Torge~           39.5          17.4              186        3800 fema~
3 Adelie  Torge~           40.3          18                195        3250 fema~
4 Adelie  Torge~           NA            NA                 NA          NA NA
5 Adelie  Torge~           36.7          19.3              193        3450 fema~
6 Adelie  Torge~           39.3          20.6              190        3650 male
# ... with 1 more variable: year <int>
```

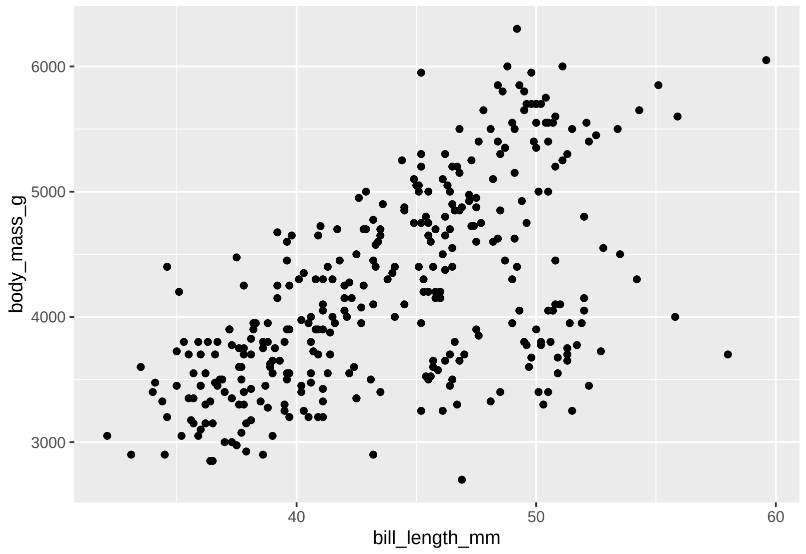
To save reading space and time, you can refresh your memory for this data by entering `help(penguins)` in the console.

For this question, we will assume that someone ultimately wants to write a set of functions that will allow them to create scatterplots of all possible combinations of two quantitative variables with a single line of code. The different parts below will allow to create some useful functions to build up to that goal with some different ways to implement them. Please be sure to read carefully to understand each part.

(a) **[20 pts]** The core function that we need to create is a function that takes in one argument which is a tibble object and two character string arguments (i.e. two single element character vectors) which indicate which variables from the tibble should be plotted against each other and returns a ggplot object. If either of the two character string arguments do not match names of columns of the tibble, your function should return an error message.

For example, if the tibble argument is `penguins` and the two character string arguments are `x_var="bill_length_mm"` and `y_var="body_mass_g"`, then calling the function would yield:

```r
one_plot_fun(input_data=as_tibble(penguins),x_var="bill_length_mm",y_var="body_mass_g")
```



But if we give it `foot_length_mm`, then it should return:

```r
one_plot_fun(input_data=as_tibble(penguins),x_var="foot_length_mm",y_var="body_mass_g")
```

```
[1] "At least one variable not contained in input_data"
```

**Hint:** You will need to use `aes_string` instead of `aes` to set the aesthetics for your scatterplot. Recall the `%in%` operator allows you to check to see if elements of one vector are contained in another. For example,

```r
a<-LETTERS[1:10]
b<-c("Z","C")
b %in% a
```

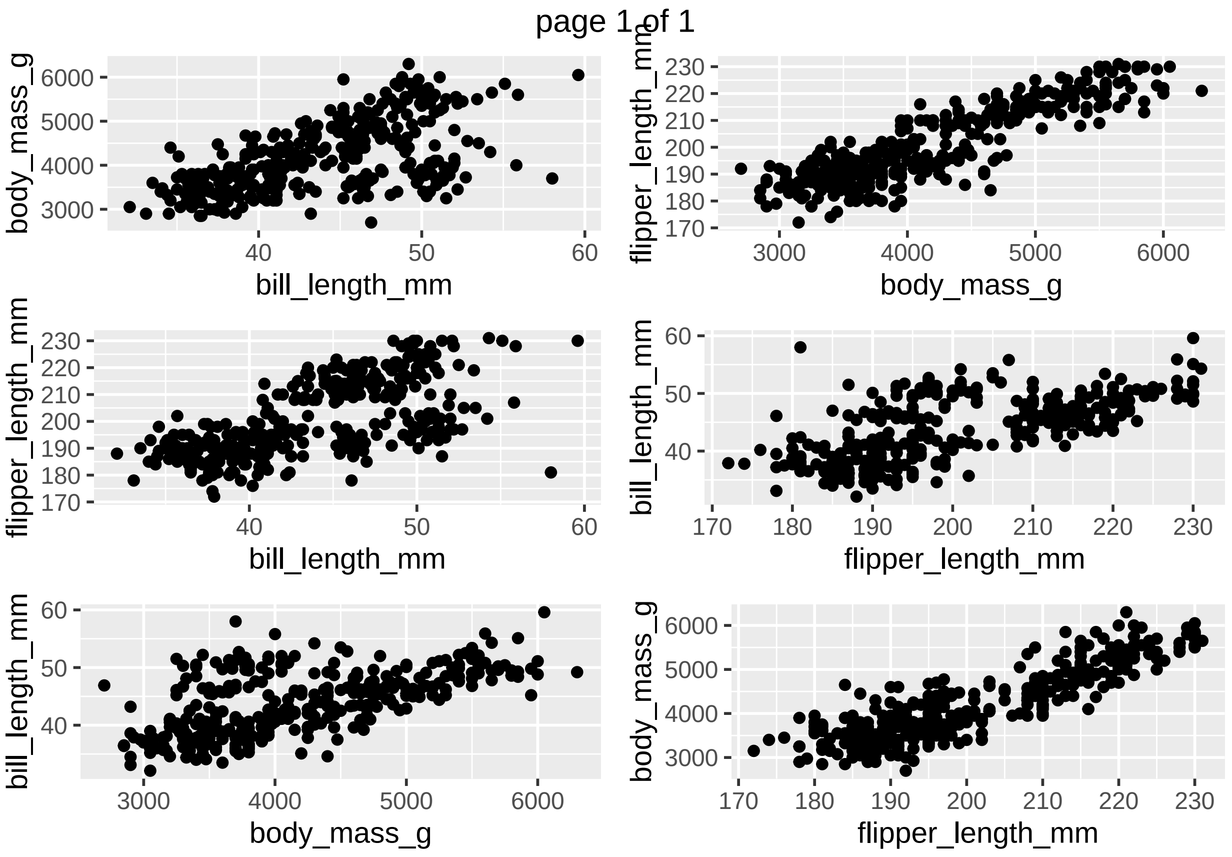
```
[1] FALSE  TRUE
```

(b) [**30 pts**] Now using your function from part (a), write a function that takes two arguments, the input data tibble and a character string vector that contains variable names from the tibble that you want to be plotted against each other, and returns a list containing scatterplots of all possible combinations of variables, **excluding plots where both the x and y axis variables are the same**. Your function should use a `for()` loop (or multiple `for` loops) to generate plots for all possible combinations of variables in the character string vector. If either of the pairs of values from the character vector do not correspond to columns in the tibble, the element of the list should just contain the error message from part (a). Here is an example of running your function for the `penguins` data:

```
my_obj<-many_plots_fun(penguins,c("bill_length_mm","body_mass_g","flipper_length_mm"))
str(my_obj,max.level=1)
```

```
List of 6
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
 $ :List of 9
  ..- attr(*, "class")= chr [1:2] "gg" "ggplot"
```

```
gridExtra::marrangeGrob(my_obj,nrow=3,ncol=2)
```



**END OF QUESTION 3**