

# PSTAT 10 Homework 1

Due 7/1/24 11:59pm

Hand in your homework on time! Your TA has a life and will want to grade your homework shortly after it is due. Late homework assignments will receive a penalty.

## Problem 1: Exploring the ecosystem

Here is a list of five R packages \* beeper \* fun \* fortunes \* cowsay \* praise

Choose one of these packages and download its source code. The list of all packages on CRAN is here <https://mirror.las.iastate.edu/CRAN/>. On a package's page, you can find the *package source* to download, which will be in tar.gz format. This is called a *tarball*.

Unzip the tarball, navigate to an R script, and copy and paste a function's name and arguments (but not the body) to your homework solution. Comment on the name of the function and how many arguments it has. For example in cowsay, the script in `cowsay/R/utils.R` contains a function `check_color`:

```
check_color <- function(clr) {
```

It takes one argument, `clr`. If you're wondering how I included an incomplete piece of code in my R Markdown, it is because I added `eval = F` to my code chunk, like this: `{r eval=F}`.

## Problem 2: More ecosystem exploration

The point of this problem is to demonstrate that a community of developers are constantly improving the R language.

If you installed R recently, you should be on version 4.4.1 (to check, run `R.Version()` in the console).

Describe one feature or bug fix that is new in version 4.4.1. This can be found via the *what's new* link in <https://mirror.las.iastate.edu/CRAN/>. You can simply copy and paste the description for your solution to this problem.

## Problem 3: State areas

Load the library called `datasets`. The vectors `state.area` and `state.name` contain in alphabetical order the land area (in square miles) and the names of each state in America.

1. Find the mean area of all states.
2. Find the median area of all states.
3. Find the name and the area of the smallest state.
4. Find the name and the area of the largest state.

There are many ways to do parts 3 and 4, but one way is to use `which.min` and `which.max`.

## Problem 4: A function without parameters

Most of functions you have encountered so far have parameters. It is also possible to create and call functions without parameters, such as `Sys.time()` or `getwd()`. This can be useful if you are writing complex code such as an R package and you want to reduce the amount of code you have to handle.

Note: A parameter is a variable in a function definition. It is a placeholder and hence does not have a concrete value. An argument is a value passed during function invocation. In a way, arguments fill in the place the parameters have held for them.

Write a function `my_time` that takes no arguments and prints `Hello, the time is` followed by the actual current time (but not the date), all in one sentence. Take a look at the help files of the `paste0`, `Sys.time`, and `strptime` functions if you get stuck.

## Problem 4: Length of a vector

The length of a vector is the square root of the sum of the squares of the horizontal and vertical components. Write a function that takes a vector as its input and returns the length of the vector as its output. If the vector is not numeric, print the message `The argument must be a numeric vector!`

Test the function with these examples

```
vector_length(c(5, 9, 11, 15))
```

```
## [1] 21.26029
```

```
vector_length(c("mouse"))
```

```
## [1] "The argument must be a numeric vector!"
```

## Problem 4: Dot product

The dot product of two vectors  $(x_1, x_2, \dots, x_n)$  and  $(y_1, y_2, \dots, y_n)$  is the number

$$\sum_{i=1}^n x_i y_i = x_1 \times y_1 + x_2 \times y_2 + \dots + x_n \times y_n.$$

Write a function `dot_product` that takes two numeric vectors and returns their dot product. If either argument is not numeric, print the message `Both arguments must be numeric!`

Test your function on the following inputs:

```
dot_product(1:3, c(0, 1, 5))
```

```
dot_product(2, 4)
```

```
dot_product(c(1, 1), c("dog", "cat"))
```

## Problem 5: Frobenius norm

The Frobenius norm of a matrix is the square root of the sum of the squares of all its entries. For example, the Frobenius norm of the matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

is  $\sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{1 + 4 + 9 + 16} = \sqrt{30} \approx 5.48$ . Write a function `frobenius_norm` that takes a numeric matrix and returns its Frobenius norm. If the input is not numeric print `Argument must be numeric!` and if the input is not a matrix, print `Argument must be a matrix!`.

Test your function on the following inputs:

```
frobenius_norm(matrix(1:4, nrow = 2, ncol = 2))
```

```
## [1] 5.477226
```

```
frobenius_norm(c(3,5,7,10,15,21))
```

```
## [1] "Argument must be a matrix!"
```

```
frobenius_norm(matrix(c(3,5,7,10,15,21), nrow = 2, ncol = 3))
```

```
## [1] 29.1376
```

```
frobenius_norm(matrix(c(3,"fish",7,10,15,21), nrow = 2, ncol = 3))
```

```
## [1] "Argument must be numeric!"
```

## Problem 6: Compare Count

Write a function `compare_count` that takes three arguments: `x`, `y`, `comp`. Arguments `x` and `y` are numeric vectors of the same length, and `comp` is one of the three characters (strings) `>`, `<`, `=`. The function `compare_count` returns the number of elements in `x` that are greater than, less than, or equal to the corresponding element in `y`, depending on the value of `comp`. The `comp` argument should default to `>`. If these assumptions are not met print error messages that match the sample output below. If there is an error, do not do any further operations.

```
compare_count(rep(1, 5), rep(2, 5))
```

```
## [1] 0
```

```
compare_count(rep(1, 5), rep(2, 5), ">")
```

```
## [1] 0
```

```
compare_count(c(1, 2, 1, 2, 1), rep(2, 5), "<")
```

```
## [1] 3
```

```
compare_count(c(1, 2, 1, 2, 1), rep(2, 5), "=")
```

```
## [1] 2
```

```
compare_count(c(1, 2, 1, 2, 1), rep(2, 5), ">=")
```

```
## [1] "Unrecognized compare operator!"  
## NULL
```

```
compare_count(c(1, 2, 1, 2, 1), rep(2, 6), "=")
```

```
## [1] "Both vectors must have the same length!"  
## NULL
```

```
compare_count(c(1, 2, 1, 2, "owl"), rep(2, 6))
```

```
## [1] "Both vectors must be numeric!"  
## NULL
```

## Problem 7: Styling your code

Writing readable code is important and it is best to get it right the first time around. This will not only help your collaborators read your code but also yourself.

Read Chapter 2: Syntax of the tidyverse style guide. Then go through your code and make sure it conforms to the tidyverse style guide.