# Lab 4: Factors, Lists and Functions

## 1 Introduction

The goal of this lab is to briefly mention two other types of `R` objects that you may encounter: **Factors** and **Lists**. You will also work a little more with **functions** and their **arguments**. The `R` language is very function based so getting a handle on how to read the help files to figure out what arguments are involved is beneficial.

### 1.1 Factors

The term factor refers to a statistical data type used to store categorical variables. The difference between a categorical variable and a continuous variable is that a categorical variable corresponds to a limited number of categories, while a continuous variable can correspond to an infinite number of values.

A good example of a categorical variable is the variable `Species` in the `iris` data set. A plant can only belong to a single species. So here “setosa” and “versicolor” and “virginica” are the three values or levels of the categorical variable “Species”, and every observation can be assigned to only one of those values.

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

```
class(iris$Species)
```

```
## [1] "factor"
```

R treats factors very differently than any other data type (for better or worse). When in doubt, or if `R` is giving you trouble about factor levels, you can convert the variable of interest to character using the `as.character()` function.

```
a <- as.character(iris$Species)
class(a)
```

```
## [1] "character"
```

### 1.2 Lists

A list in R allows you to gather a variety of objects under one name (that is, the name of the list) in an ordered way. These objects can be matrices, vectors, data frames, even other lists, etc. It is not even required that these objects are related to each other.

```
my_vec <- sample(1:10,5)
my_mat <- matrix(rnorm(1:10), nrow=5)
my_df  <- iris[1:5,]

my_list <- list(vec=my_vec, mat=my_mat, df=my_df)
my_list
```

```
## $vec
## [1] 7 6 3 5 8
##
## $mat
##            [,1]       [,2]
## [1,]  1.0522197 -0.1820328
## [2,] -0.8513383  1.0401500
## [3,]  2.3348644  0.2324734
## [4,]  1.3333604 -0.2810294
## [5,]  0.7535776 -0.2046667
##
## $df
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

```r
str(my_list)
```

```
## List of 3
##  $ vec: int [1:5] 7 6 3 5 8
##  $ mat: num [1:5, 1:2] 1.052 -0.851 2.335 1.333 0.754 ...
##  $ df :'data.frame': 5 obs. of  5 variables:
##   ..$ Sepal.Length: num [1:5] 5.1 4.9 4.7 4.6 5
##   ..$ Sepal.Width : num [1:5] 3.5 3 3.2 3.1 3.6
##   ..$ Petal.Length: num [1:5] 1.4 1.4 1.3 1.5 1.4
##   ..$ Petal.Width : num [1:5] 0.2 0.2 0.2 0.2 0.2
##   ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1
```

We will not be using lists in this class but you should know that they exist. This knowledge will be helpful when you debug your code, and when you are looking up further help on the internet. Lists can be very useful, they are just outside the scope of this class.

## 1.3 Functions and their arguments

By now you've seen the term `function` being thrown around all over the place. Functions take inputs, called **arguments** and provide outputs, or results. A few functions you have already used are `head()`, `table()` and `subset()`. Let's look at the `mean` function again by typing ?mean.

The **Usage** section of the documentation includes two versions of the mean() function; What's the difference? The first function

```r
mean(x,...)
```

is the most general definition of the mean function. This section also shows you what the default values for each argument are. This is a very important piece to pay attention. Sometimes the default behaviors are not what you want to happen.

```r
mean(x, trim = 0, na.rm = FALSE, ...)
```

In the **Arguments** section the help file defines what each argument does.

- `x` is the object that you want to take the mean of
- `trim` is a number from 0 to 0.5 that defines the fraction of observations to be excluded from each side before the mean is calculated.
- `na.rm` is a logical value (`TRUE`/`FALSE`) that tells `R` whether NA values should be stripped before the computation proceeds.
- `...` is called the ellipsis, and it is a way for R to pass arguments to or from other methods without the function having to name them explicitly. The ellipsis will be treated in more detail in an Advanced R course.

### 1.3.1 Argument ordering

A function's arguments can be named, or can be referred to by position. As an example, let's look at a vector of random numbers with some missing.

```r
y <- sample(1:100, 20)
y[sample(1:length(y),5)] <- NA
y
```

```
##  [1] NA 88 26 NA NA 78 37 60 16  7 NA 28 82 NA 72 77 94 58 40 24
```

Now, let's calculate the mean.

```r
mean(y)
```

```
## [1] NA
```

Oops, forgot about the missing values.

```r
mean(y, na.rm=TRUE)
```

```
## [1] 52.46667
```

It worked fine because I named the argument to remove missing values. What if I didn't state what that argument was?

```r
mean(y, TRUE)
```

```
## Error in mean.default(y, TRUE): 'trim' must be numeric of length one
```

R is expecting a value for trim as the second argument and doesn't know what to do with the value `TRUE`. If you name the arguments, then the order is irrelevant.

```r
mean(na.rm=TRUE, x=y, trim=.1)
```

```
## [1] 52.76923
```

But let's not get that crazy.

## 2 Practice

Use the `NCbirths` data set for the following questions.

1. Calculate the mean age of the mothers (`mage`) in the sample.
2. Calculate the trimmed mean mothers age after eliminating the outer 10% of ages
3. Calculate the mean fathers age (`fage`).
4. Do mothers who smoke give birth to smaller babies? *(Hint: Use **`subset`** to create subsets of the data set by smoking **`habit`** status then find the mean **`weight`** for each subset.)*
5. Do more non white mothers smoke? Use `table()` to find this answer and don't forget to check for missing values.