Mindset and Grit

Do you believe you have a set of fixed traits?

It’s something we don’t talk about - and yet it controls our every day decisions and actions.

Carol Dweck, Instructor of Psychology at Stanford University, first outlined the concept of a growth mindset and a fixed mindset.

> **In a fixed mindset**, people believe their basic qualities, like their intelligence or talent, are simply *fixed traits.*
> They also believe that *talent alone creates success—without effort*.
>
> They’re wrong.
>
> **In a growth mindset**, people believe that their most basic abilities can be developed through dedication and hard work—brains and talent are just the starting point. This view creates a *love of learning* and a *resilience* that is essential for great accomplishment.
>
> Virtually all great people have had these qualities.

The resilience mentioned above refers to **Grit.** Grit, according to psychologist Angela Duckworth, is the level of sustained passion one has towards a longterm goal or end state. This is something that must be developed, like a muscle. It take practice to work and grind to find success.

Angela says:

> *“You cannot will yourself to be interested in something you’re not interested in. But **you can actively discover and deepen your interest**. So **once you’ve fostered an interest**, then, and only then, can you **do the kind of difficult, effortful and sometimes frustrating practice that truly makes you better.***

This is great to keep in mind when learning something new - especially when learning computers and technology. It can be VERY frustrating and 80% of the struggle is debugging and checking for errors.

There is nothing more painful than running code that won’t work - and you cannot understand why. You will want to throw your computer across the room and you will probably develop (if not already) the mouth of a sailor.

If anyone needs any help with homework, programming, or general questions please feel free to email me at jessprim1@gmail.com

That being said the best part is the frustration because those are lessons/moments you will never forget.

Programming in R

We need to install 2 pieces of software, R (the language) and RStudio (the IDE).
When you write in any language, there are typically two pieces. The language and the IDE.

> **Integrated Development Environment (IDE):** A software application that provides tools and other such facilities to computer programmers for software development.

An IDE is a GUI

> **Graphical User Interface (GUI):** A user interface that includes graphical elements, such as buttons, icons.. etc.

If we did not use the R IDE we would use command line instead…

Installing R:

1. In your browser visit : `www.r-project.org`
2. Under "Getting Started", click **"download R"** link in the middle of the page
3. You will be led to a page titled "CRAN Mirrors".
4. Scroll down to USA and find the link to the University of Kansas, Lawrence, KS.
5. There are two links for Lawrence, select either one.
6. You will be redirected to "The Comprehensive R Archive Network"
7. Under "Download and Install R" select the Download link for whatever system your machine requires. Then follow the directions below:

R for Windows:

1. Click "Download R for Windows"
2. Click "Install R for the first time" link at the top of the page
3. Click "Download R 3.4.2 for Windows" and save the executable file somewhere on your machine. Run the .exe file and follows the installation instructions.

R For Mac OS:

1. Click “Download R for (Mac) OS X”
2. Click on the file “R - 3.4.2.pkg”
3. Save the .pkg file somewhere on your machine. Double click the file to open and follow the install instructions.

Installing RStudio:

1. Visit the site: `www.rstudio.com`
2. Click on “Download RStudio Desktop”
3. Select the version required for your system.
4. Save and run the file

Fundamental Concepts

Data Types:
R is an object-oriented language. This means everything in R is considered an object.

There are 6 object types in R:

- Character `"a", "Jessica"`
- Numeric `2, 7.9`
- Integer `2L`
- Logical (Boolean) `True, False`
- Complex `2 + 5i`

There are a lot of functions we can use to examine our data more closely:

> class() - What kind of object is it?
> typeof() - What is the object’s data type?
> length() - How long is it, how many dimensions?
> attributes() - Is there any metadata?

Data Structures:
In computer science a data structure is a way of organising and storing data. The way data is store is important because it determines how it can be accessed and modified.

R’s data structures include:

- Atomic Vectors
- Data frame

- Factors
- Matrix
- List

Vectors:

The atomic vector is the simplest data structure in R. A vector is a collection of elements of type character, logical, integer, or numeric.

We can create a vector like so:

```
x <- c(3, 8, 9, 7, 6, 7, 1, 2)
x <- c(“Jessica”, “Alexa”, “Harrison”, “Jill”, “Edward”)
n <- c(FALSE, TRUE, TRUE, FALSE)
```

If you type

```
x
```

in your console, you will see the contents of x.

A vector can also be created like this:

```
vector(“character”, length = 3)
character(3)
logical(4)
numeric(8)
```

Concatenate:
c( ) means concatenate. To concatenate is to put elements together.
You can add more elements to a vector like so:

```
m <- c(“lisa”, “sally”, “winston”)
m
m <- c(m, “olga”)
m
m <- c(“danny”, m)
```

Sequences:
You can also create a sequence of numbers!

```
sequence <- 20:30
seq(8)
seq(from = 2, to = 12, by = 0.5)
```

NAs and Missing Data:
NA (Not Available) represents missing data in R. R can recognise missing data and has built in functions that allow you to explore NAs in more detail.

NA functions include:

```
is.na( )
anyNA( )
```

Let’s see how these work:

```
z <- c(“The”, “quick”, “brown, “fox”, NA, NA, NA, “lazy”, “dog”)
a <- c(“The”, “quick”, “brown, “fox”, “jumped”, “over”, “the”, “lazy”, “dog”)
is.na(z)
is.na(a)
anyNA(z)
anyNA(a)
```

You may come across other values interpreted by R:

- Inf: Infinity
- NaN: Not a Number

These cannot be found by is.na or anyNA. Rather, use:

```
is.infinite( )
is.nan( )
```

Attributes:
Objects have attributes, which are details about the object. This is considered metadata: data about the data. You can use these commands to look at an objects attributes:

```
levels( )
dim( )
names( )
dimnames( )
class( )
attributes( )
```

Matrices:

A matrix a 2-D vector. A vector is considered to by 1 dimensional and thus when we print some defined vector we see a list of data. However, when we look at a matrix we see the beginnings of what looks like a data table, with rows and columns.

Let’s make a matrix:

```
matrix_1 <- matrix(nrow = 5, ncol = 5)
matrix_1
dim(matrix_1)
```

We can also use cbind( ) or rbind( ) to combine rows and columns and make a matrix:

```
one <- 1:5
two <- 6:10
cbind(one, two)
rbind(one, two)
```

Lists:

A list is like a container in R. Unlike vectors, lists can contain mixed data types. Lists can even contain lists. Inception. We can make lists in two ways. The first way we make a list is by explicitly making one:

```
l1 <- list(4, “apple”, FALSE, 10.6)
l1
```

The second way we make a list is through coercion. This is a very useful technique to learn in R. You will probably use coercion more than you think once you get going analysing data. Using the command as.list( ) we can make a vector into a list.

```
v_2 <- 1:5
v_2 <- as.list(v_2)
length(v_2)
```

Names:
Elements can have names, remember names( ) was an attribute we learned earlier.

```
l_3 <- list( a = “Jessica”, b = 1:4, data = head(cars))
```

```
l_3
names(l_3)
```

You can reference the name of an element with the $ symbol

l_3$al3\,al_3$data

Data Frames:

Now we've gotten to the sacred data frame, a popular data type in R that any statistician uses all the time. A data frame is a list with a fixed length. If you load in data with 20 rows, then every element will have 20 rows.

We can create a data frame in R just by reading in data. Read in commands are:

```
read.csv( )
read.table( )
mfread( )
```

You can also create a data frame on your own:

```
data.df <- data.frame( x = letters[1:5], y = 1:5, z = 6:10)
data.df
```

We can look at our data frame by using these commands:

```
head( )
tail( )
summary( )
names( )
nrow( )
ncol( )
```

Practice:

R has some built in data packages. We can explore the functions we just learned with the data.

Finding data is more difficult than you think! If you want to explore some places try Kaggle.com or KDnuggets.come for free data.