#: R as a tool ## Excel vs. R and why we should care ### Excel vs R {.unnumbered} When choosing between R and Excel, it is important to understand how both solutions can get you the results you need. However, one can make it an easy, reputable, convenient process, whereas the other can make it an extremely frustrating, time-consuming process **prone to human errors**.

When opening Excel and applying data manipulation techniques to your data, are you **easily** able to tell what manipulations have been made without clicking on the column or cells? If you were to share these Excel sheets with colleagues are they easily able to **replicate** your analyses without you telling them where to click or which formulas were applied?

With R all of these are possible. You automatically have all the code visible and in front of you in the form of scripts. Reading and understanding the code is possible because of its easy-to-use, easy-to-read syntax which allows you to track what the code is doing without having to be concerned about any hidden functions or modifications happening in the background.

When we consider our programming methods, we must strive for two goals: **simple and reproducible**. R makes both of these goals achievable.

Let's keep talking about this

I want to inform you of something. This is entirely objective and bias-free (as if that is even possible).

Let's talk excel data sheets for a moment. *Excel* has some great features. The most flexible of these is the *cell*. An excel cell can be extremely flexible as they can store various data types (numeric, logical, and characters).

This is great! We can store our data here in a nice and organized manner and scroll through and view it all with relative ease.

Not so fast. Lettuce think about a data set with 5,000 or 20,000, or 100,000, or 500,000, or 1,000,000 rows and 100+ columns. Now. Imagine **scrolling** through all of this looking for errors. Or double checking formulas written within new columns. Imagine saving this file over and over upon each rendition. What were to happen if an **error was missed** after formula was run and you continued to work and save new files? This could mean big trouble when it came time for a real analysis. Personally, that sounds like a **nightmare.**

Now that I got that off my chest. Let's chat about R. Within R are some great options for viewing our data. We can look in our environment. We can call certain base R functions (See functions section here) to view different sections.

Here are some examples of these functions.

The structure (**str**) to view the nature of our data set.

	A	B	C	D	E	F	G	H
1	So	much	data	omg	and	look	here	ItNeverEnds
543	542	544	546	548	550	552	554	556
544	1	545	547	549	551	553	555	557
545	544	2	548	550	552	554	556	558
546	545	547	3	551	553	555	557	559
547	546	548	550	4	554	556	558	560
548	547	549	551	553	5	557	559	561
549	548	550	552	554	556	6	560	562
550	549	551	553	555	557	559	7	563
551	550	552	554	556	558	560	562	8
552	551	553	555	557	559	561	9	565
553	552	554	556	558	560	10	564	566
554	553	555	557	559	11	563	565	567
555	554	556	558	12	562	564	566	568
556	555	557	13	561	563	565	567	569
557	556	14	560	562	564	566	568	570
558	15	559	561	563	565	567	569	571
559	558	16	562	564	566	568	570	572
560	559	561	17	565	567	569	571	573
561	560	562	564	18	568	570	572	574
562	561	563	565	567	19	571	573	575
563	562	564	566	568	570	20	574	576
564	563	565	567	569	571	573	21	577
565	564	566	568	570	572	574	576	22
566	565	567	569	571	573	575	577	579

```
# The str() function to view the structure of the data set
str(Penguins)
```

```
tibble [344 × 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
 $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
 $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
 $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
 $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
 $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

The **head** function to view the first several rows of a data set.

```
# The head() function to view the several rows
head(Penguins)
```

A tibble: 6 × 8

species <fctr>	island <fctr>	bill_length_mm <dbl>	bill_depth_mm <dbl>	flipper_length_mm <int>	body_mass_g <int>	sex <fctr>	year <int>
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	*NA*	*NA*	*NA*	*NA*	*NA*	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007
Adelie	Torgersen	39.3	20.6	190	3650	male	2007

6 rows

The **tail** function to view the last several rows of our data set.

```
# The tail () functoin to view the last several rows
tail(Penguins)
```

A tibble: 6 × 8

species <fctr>	island <fctr>	bill_length_mm <dbl>	bill_depth_mm <dbl>	flipper_length_mm <int>	body_mass_g <int>	sex <fctr>	year <int>
Chinstrap	Dream	45.7	17.0	195	3650	female	2009
Chinstrap	Dream	55.8	19.8	207	4000	male	2009
Chinstrap	Dream	43.5	18.1	202	3400	female	2009
Chinstrap	Dream	49.6	18.2	193	3775	male	2009
Chinstrap	Dream	50.8	19.0	210	4100	male	2009
Chinstrap	Dream	50.2	18.7	198	3775	female	2009

6 rows

The **colnames** function to view the names of our columns.

```
#The colnmaes() fcuntion to view my column names
colnames(Penguins)
```

```
[1] "species"           "island"            "bill_length_mm"    "bill_depth_mm"     "flipper_length_mm" "body_mass_g"       "sex"               "year"
```

As we are starting to see, when compared to Excel with examples of only viewing data, R is beginning to appear more versatile. We will continue to build on the capabilities of R in future sections and work through functions, etiquette, data wrangling, plotting, and much more.