WHY R PROGRAMMING IS EASY

YOUR TEAM'S EXCEL SKILLS WILL UNLOCK ADVANCED ANALYTICS

Dave ON DATA

Introduction

I am a hands-on analytics professional and experienced leader. Over the years conducting analyses, supporting various business functions, and advising executives, I have come to believe that in the future:

1 - All aspects of business will be data-driven.
2 - Data skills will be assumed, just as skills with Microsoft Office are assumed today.
3 - The majority of data analyses will be conducted by business professionals.

My mission is to help professionals prepare for the future of business, by empowering ANY team to have more impact at work using data.

Make no mistake, the ability to conduct powerful data analyses that drive business results are not the sole domain of "Data Scientists" or "Data Analysts."

In business analytics, the 80/20 Rule applies - 20% of analytics drive 80% of business return on investment (ROI).

Want to know the best part?

The 20% of analytics can be learned by ANY motivated professional - regardless of role/background.

Don't believe it? Keep reading to see how Excel skills makes learning R easy.

-Dave

Table of Contents

Excel Users Write Code

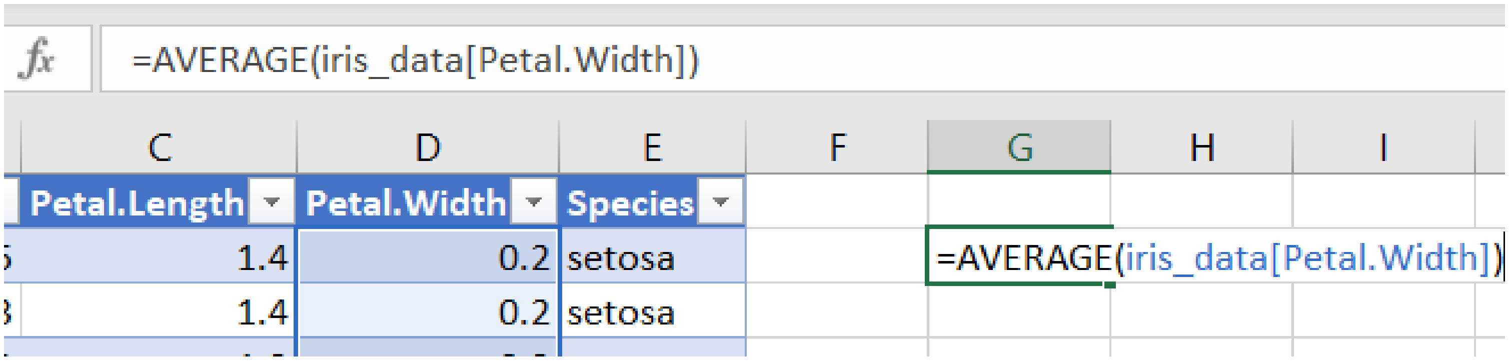
Excel Code

While most Excel users don't think of it this way, they spend a lot of time writing and debugging code in Excel. In fact, Microsoft Excel is by far and away the world's most popular programming environment.

Take the image below as example. The user is using Excel's *AVERAGE* function to calculate the average of a column (i.e., *Petal.Width*) of a table (i.e., *iris_data*) in a worksheet.

Once the user hits the <*enter*> key, Excel attempts to interpret the instructions in the cell and perform the desired operation. If Excel doesn't understand what the user typed, it reports an error.

That's coding!



```
=AVERAGE(iris_data[Petal.Width])
```

C	D	E	F	G	H	I
Petal.Length	Petal.Width	Species				
1.4	0.2	setosa		=AVERAGE(iris_data[Petal.Width])		
1.4	0.2	setosa				

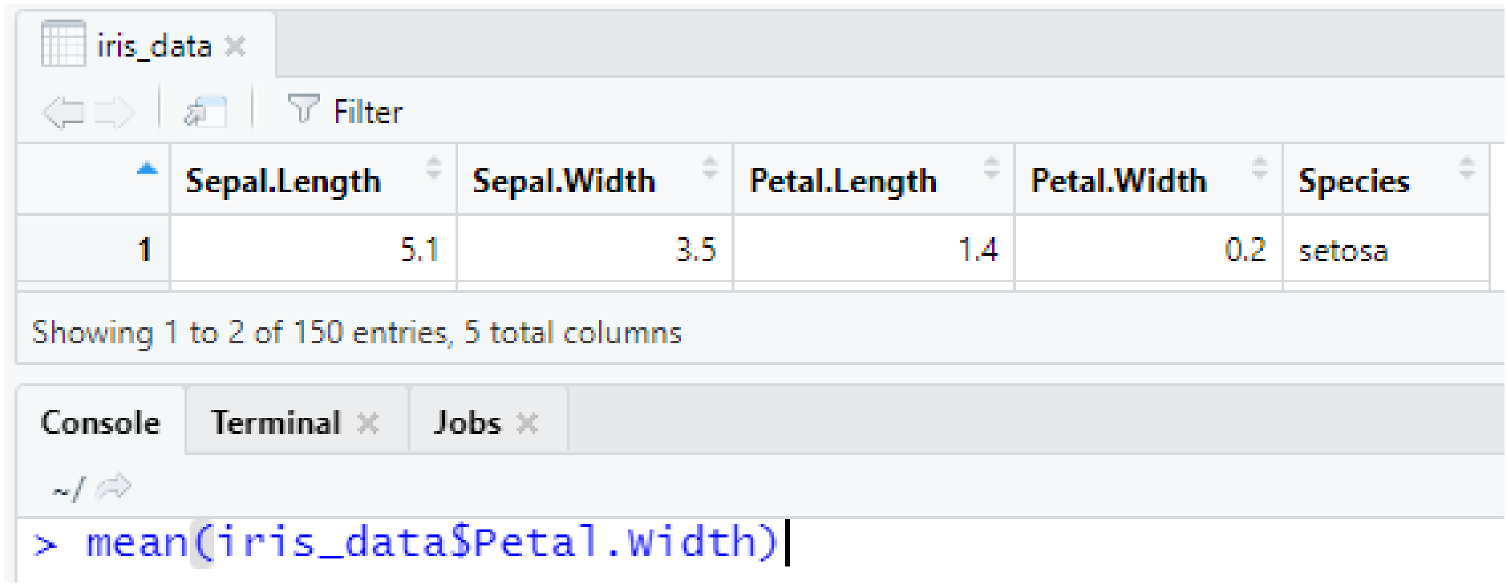
R Code

While it isn't the only way to code in Excel, calling Excel functions as depicted on the previous page is by far the most common. When using Excel in this way, Excel is operating as a code *interpreter*.

Using R as an interpreter is very common. The R user types some code and hits the <*enter*> key. R then tries to interpret the code, throwing an error if doesn't understand what was typed by the user.

In this way Excel and R are very similar, but it doesn't stop there. Even the code is very similar!

The image below depicts the same scenario as on the previous page, but using R instead. As depicted, the user is calculating the average (mean is just another name for the average) of the *Petal.Width* column of the *iris_data* table.



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa

Showing 1 to 2 of 150 entries, 5 total columns

```
> mean(iris_data$Petal.Width)
```

Excel Skills Directly Translate to R

While this example might seem simple, it demonstrates why R is the fastest, easiest way for ANY team to unlock advanced analytics.

As you will see through the rest of this document, Excel is a powerful analytical tool with many concepts and skills that need to be mastered to use Excel effectively.

This knowledge is directly applicable to R.

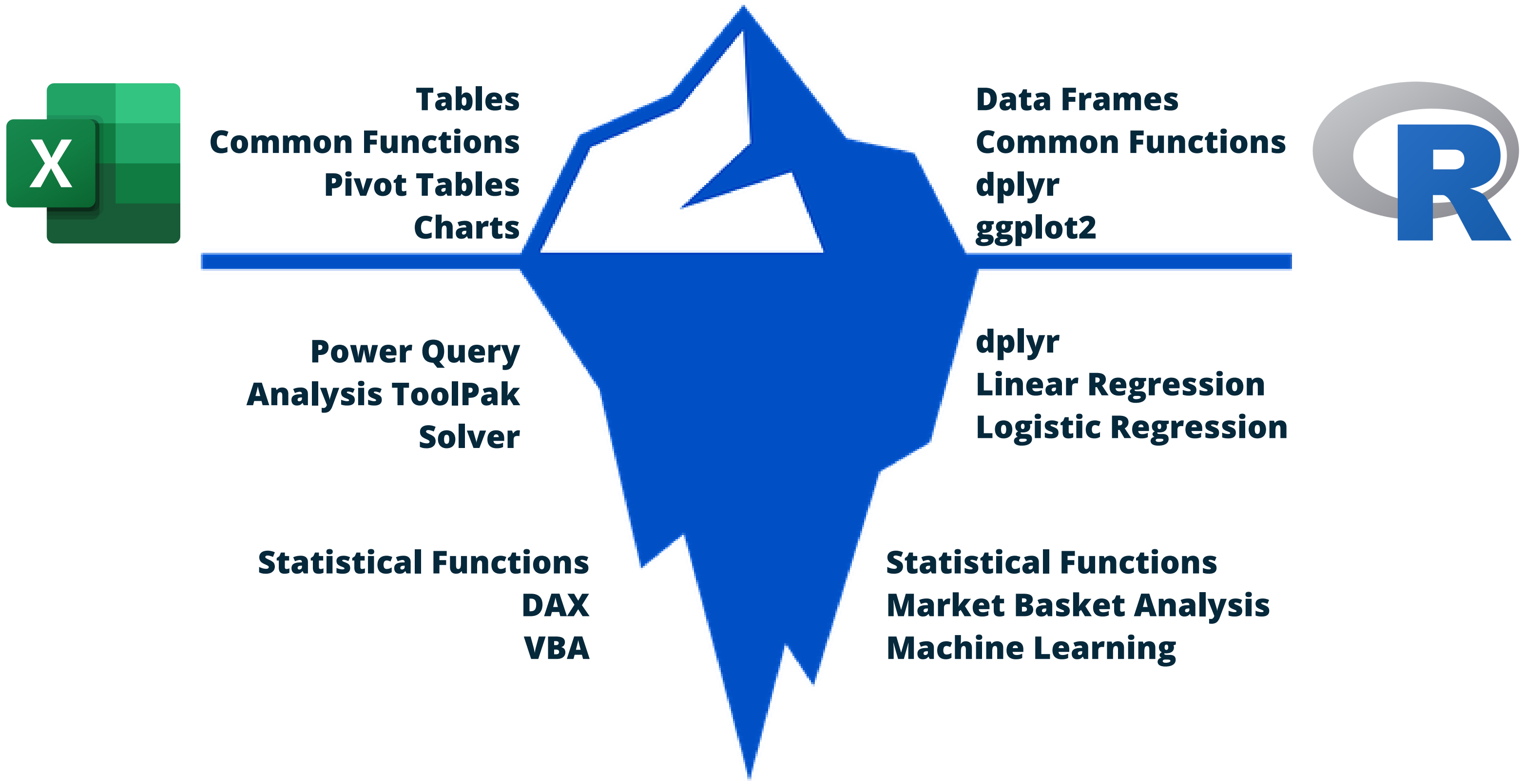
This knowledge makes the learning process an exercise in mapping Excel skills to R.

Excel and R Are Like Icebergs

While cliche, the iceberg metaphor is a powerful way to conceptualize the extent Excel knowledge maps to R.

As depicted below, Excel features above the water line (e.g., Pivot Tables) only scratch the surface of Excel's capabilities. However, these feature represent the bulk of Excel's use in practice.

Another similarity between Excel and R is the "choose your own adventure" aspect of the technologies. Just as many Excel users never learn Power Query, not every R user needs to learn statistical analysis to be effective in their work.



It's All About the Tables

Tables Are Key

Using Microsoft Excel is all about working with tables of data. You filter tables, you sort tables, you pivot tables, etc. You can think of *tables* as one of the fundamental *objects* in Excel that you create and manipulate to conduct analyses.

Excel tables can also be thought of as *container objects*. Tables contain *rows*, *columns*, *cells*, *data formats*, etc.

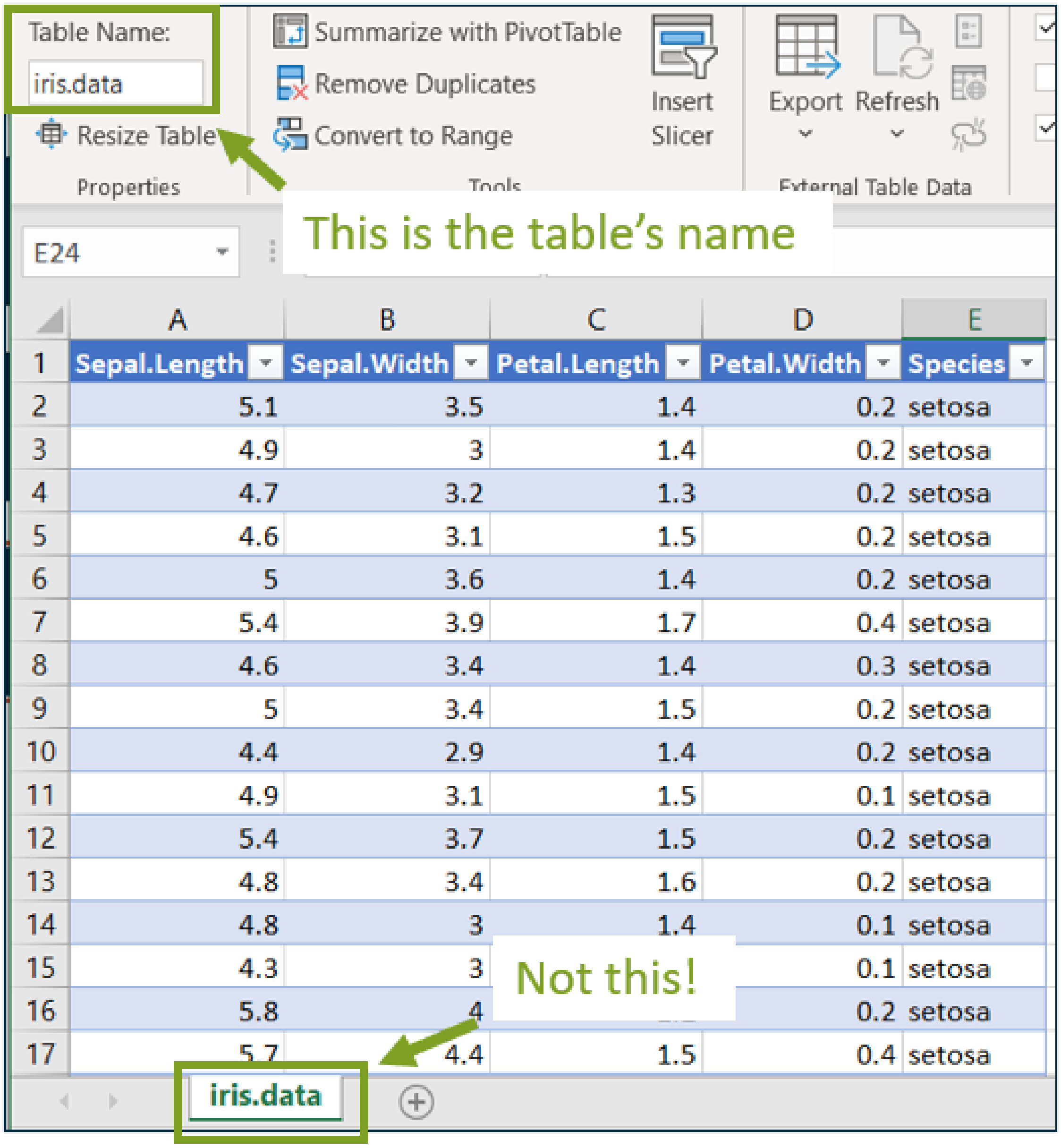
You probably can see where I'm going with this already.

When analyzing data with R, it's all about the tables - just like Excel.

Once again, your Excel knowledge directly translates to R.

Tables in Excel and R

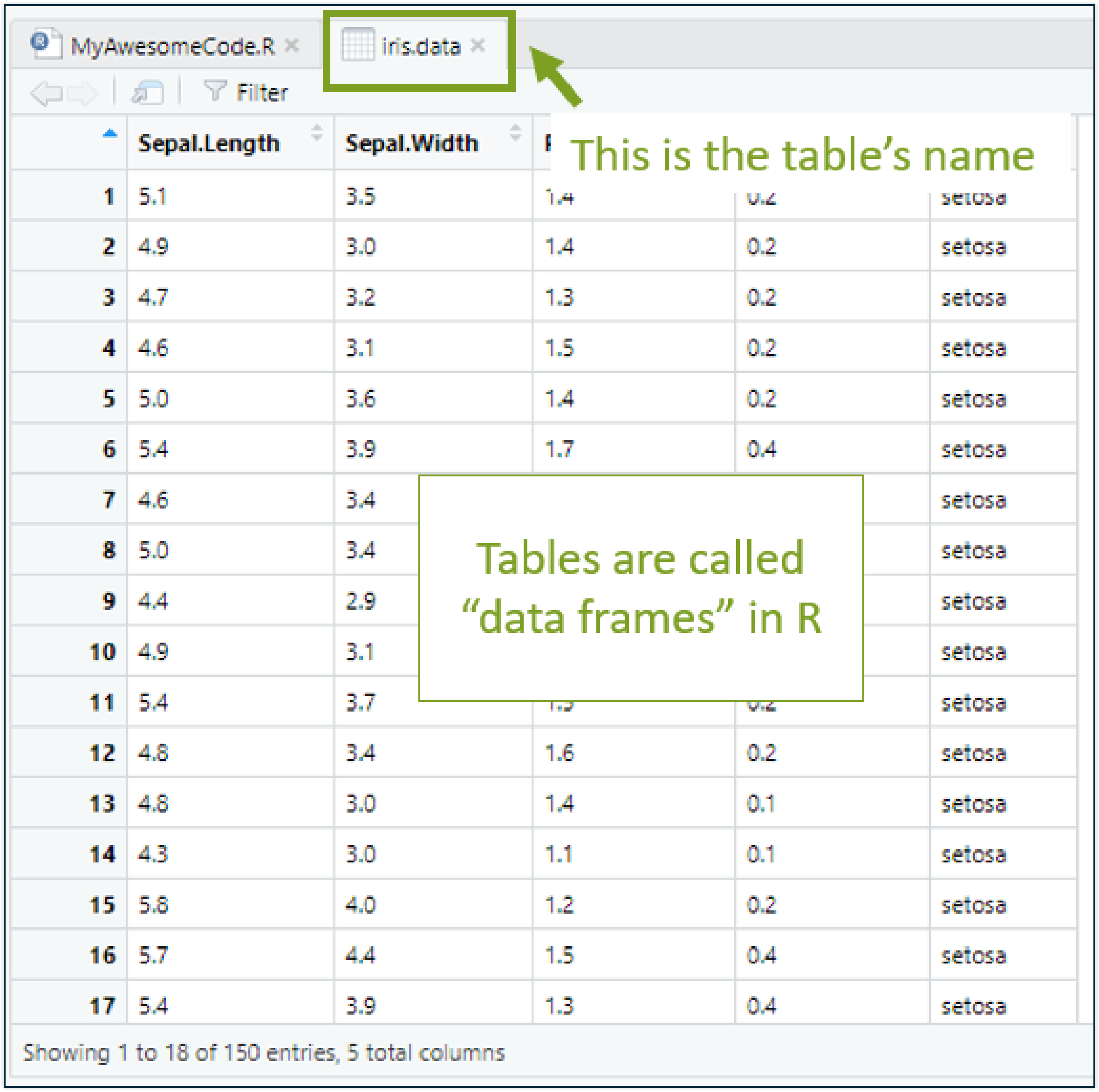
The image below demonstrates how Excel *tables* are *objects*. For example, every table in Excel has a name - whether you explicitly name a table or not. Table names allow you to directly access/manipulate tables using Excel code.



	A	B	C	D	E
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa
13	4.8	3.4	1.6	0.2	setosa
14	4.8	3	1.4	0.1	setosa
15	4.3	3	[illegible]	0.1	setosa
16	5.8	4	[illegible]	0.2	setosa
17	5.7	4.4	1.5	0.4	setosa

Things work in R exactly the same way. Tables of data in R (known as "*data frames*") have names just like Excel tables so that you can write R code to access/manipulate tables of data.

[Excel Code] *=AVERAGE(iris.data[Petal.Width])*
[R Code] *mean(iris.data$Petal.Width)*

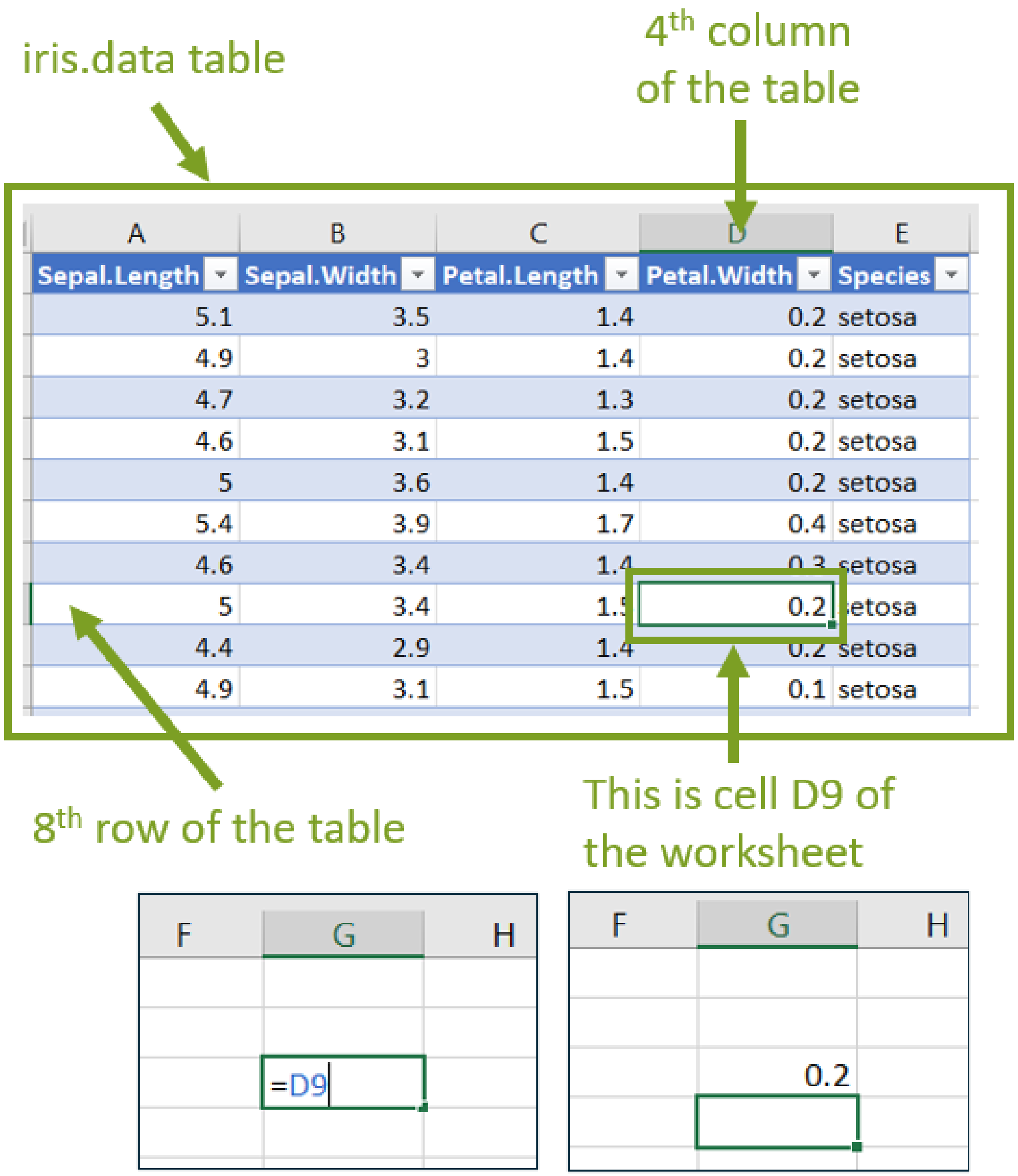


	Sepal.Length	Sepal.Width			
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	[illegible]	[illegible]	setosa
8	5.0	3.4	[illegible]	[illegible]	setosa
9	4.4	2.9	[illegible]	[illegible]	setosa
10	4.9	3.1	[illegible]	[illegible]	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa

Showing 1 to 18 of 150 entries, 5 total columns

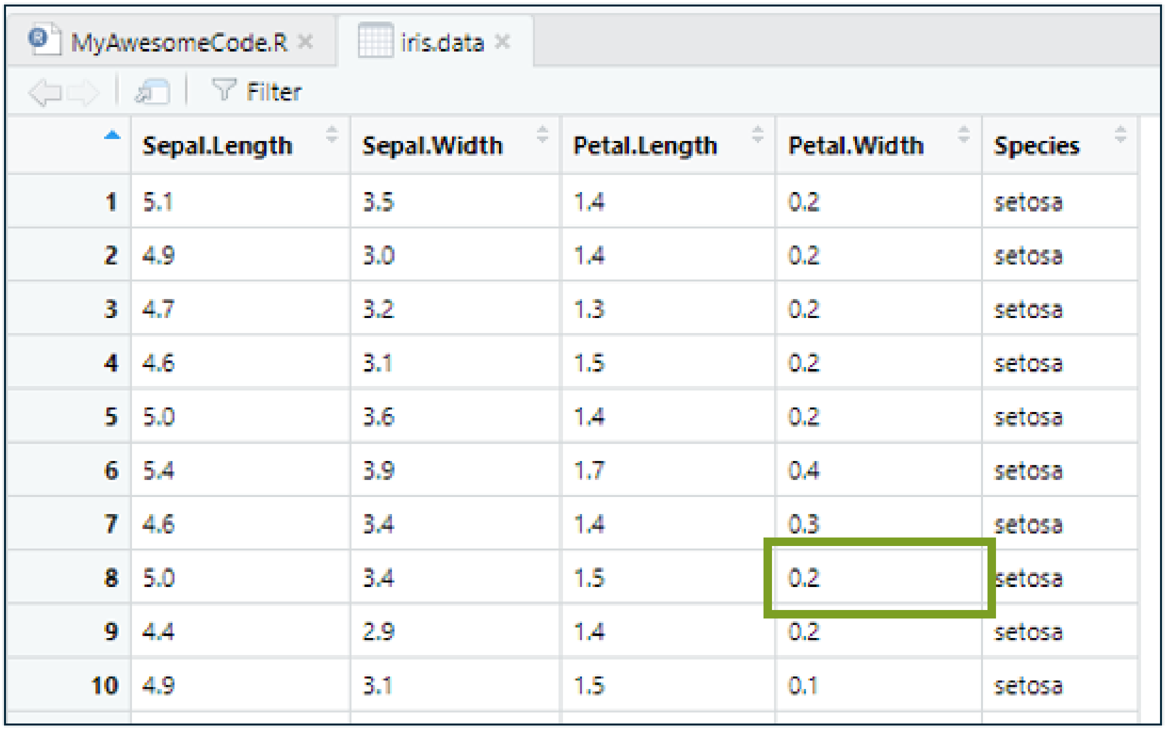
Cells of Data in Excel and R

Working with *cells* of data is very common in Excel. It is useful to think of *cells* as *objects* contained within *tables* - as depicted below. Once again, you use Excel code to access *cells*.

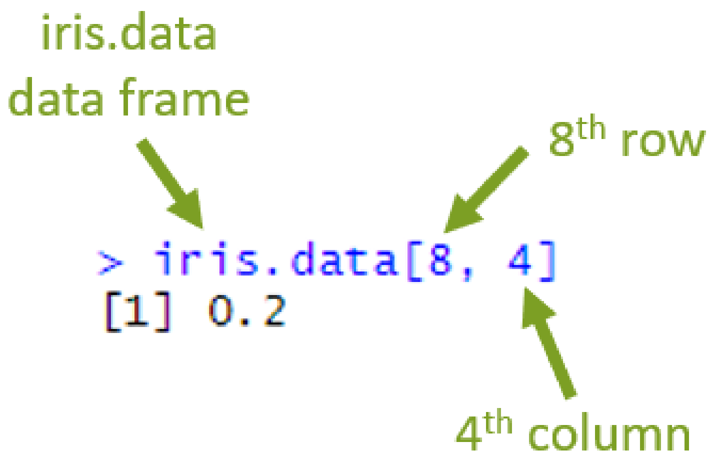


Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Although the R code to access/manipulate *cells* of data is slightly different, conceptually it matches what Excel users do all the time.



	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
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

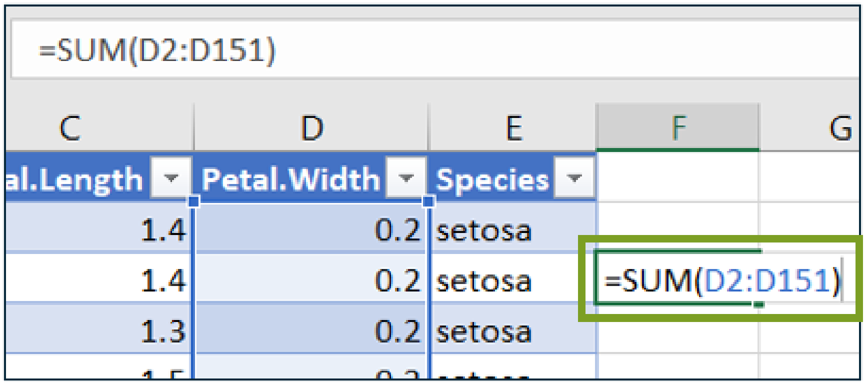


```
> iris.data[8, 4]
[1] 0.2
```

Columns of Data in Excel and R

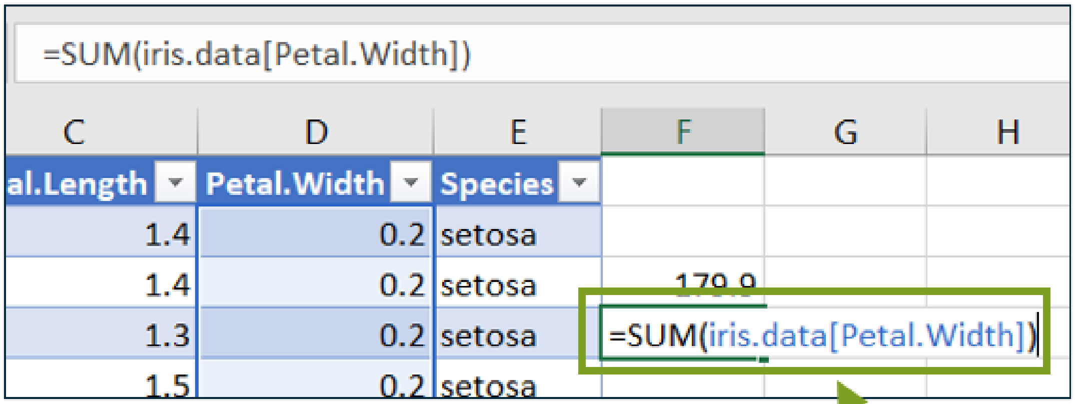
Excel code supports different ways of accessing *columns* of data within *tables*. Two examples:

- Providing a worksheet-based *cell range*
- Providing *object* names



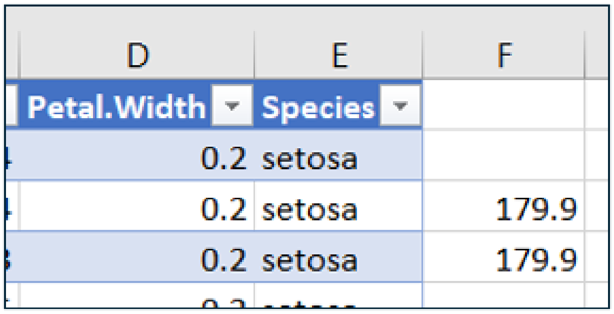
```
=SUM(D2:D151)
```

C	D	E	F	G
al.Length	Petal.Width	Species		
1.4	0.2	setosa		
1.4	0.2	setosa	=SUM(D2:D151)	
1.3	0.2	setosa		



```
=SUM(iris.data[Petal.Width])
```

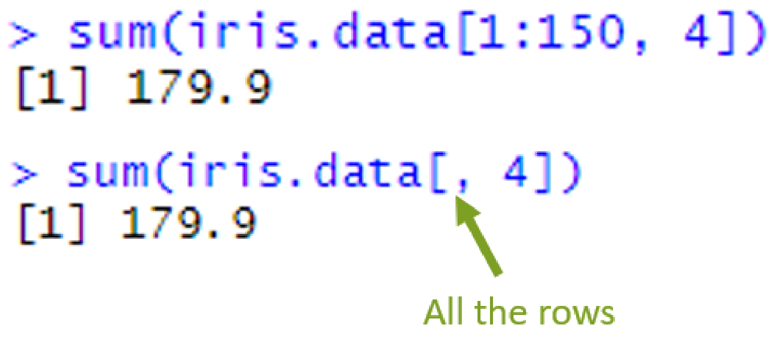
C	D	E	F	G	H
al.Length	Petal.Width	Species			
1.4	0.2	setosa			
1.4	0.2	setosa	179.9		
1.3	0.2	setosa	=SUM(iris.data[Petal.Width])		
1.5	0.2	setosa			



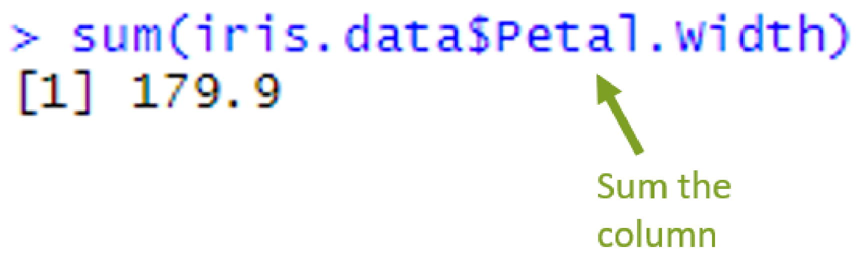
D	E	F
Petal.Width	Species	
0.2	setosa	
0.2	setosa	179.9
0.2	setosa	179.9

Once again, Excel knowledge maps directly to R code as illustrated below.

Notice how similar the actual R code is to Excel when using *object* names.



```
> sum(iris.data[1:150, 4])
[1] 179.9
> sum(iris.data[, 4])
[1] 179.9
```



```
> sum(iris.data$Petal.Width)
[1] 179.9
```

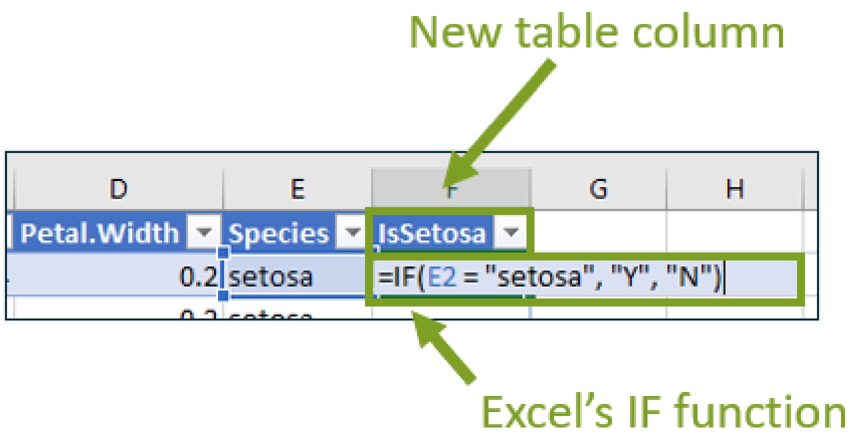
Throw in Some Functions

Using Functions in Excel and R

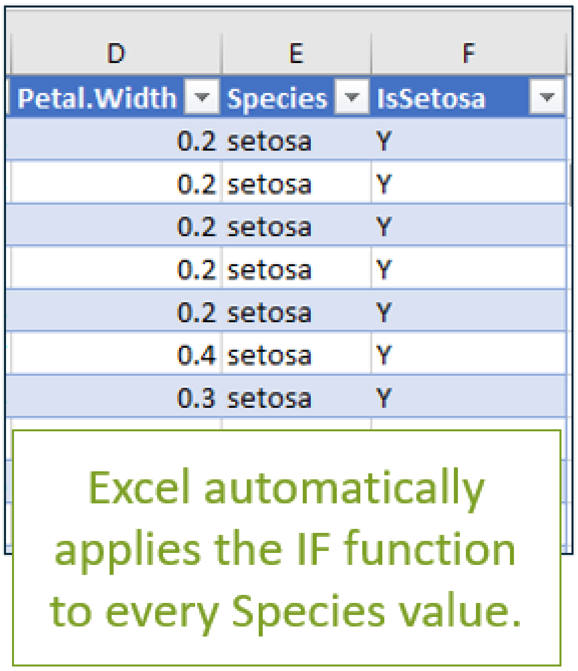
The bulk of code Excel users write call *functions*. Often, these *function* calls are nested and can be difficult to bug (again, that's coding!).

A common Excel scenario is depicted below - creating a new column of data (*IsSetosa*) by invoking a function on another column of data (Species) within the same table (*iris.data*).

In this utterly contrived example, Excel's mighty IF function (a commonly nested Excel function) is being used.



D	E	F	G	H
Petal.Width	Species	IsSetosa		
0.2	setosa	=IF(E2 = "setosa", "Y", "N")		



D	E	F
Petal.Width	Species	IsSetosa
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.2	setosa	Y
0.4	setosa	Y
0.3	setosa	Y

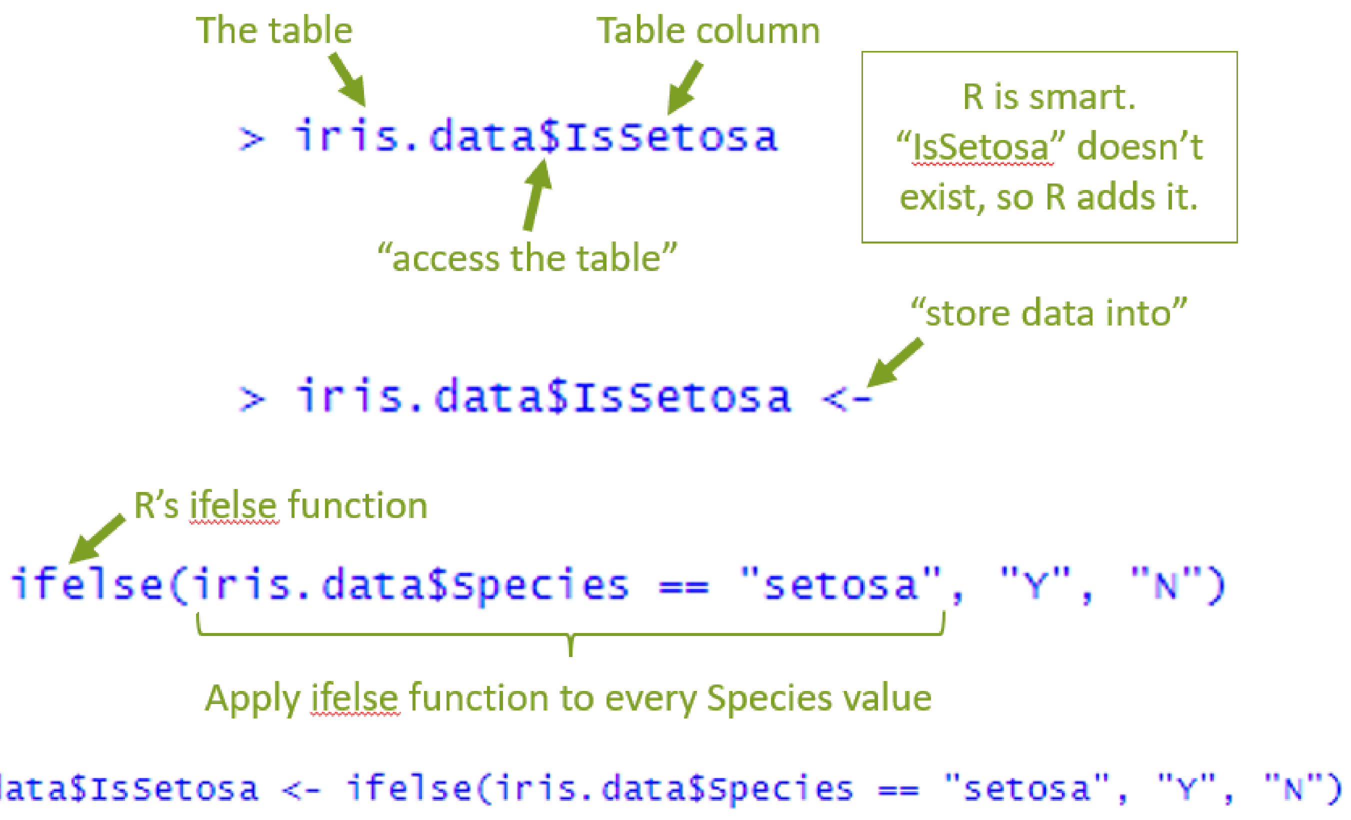
Excel automatically applies the IF function to every Species value.

The contrived example from the previous page is repeated here using R code.

A *IsSetosa* column is being added to the *iris.data* table (or *data frame*) and populated with new data derived from the existing *Species* column.

First, notice how the workflow is exactly the same as in Excel - only everything is done in code with R.

Second, notice how similar the R *ifelse* function call is to Excel code.

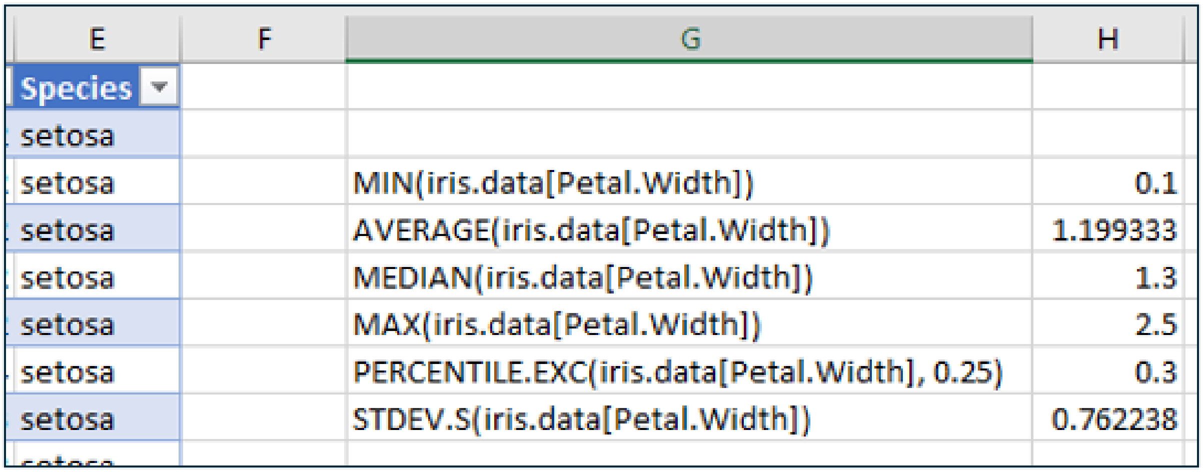

The table
Table column
“access the table”
R is smart. “IsSetosa” doesn’t exist, so R adds it.
“store data into”
R’s ifelse function
Apply ifelse function to every Species value

```
> iris.data$IsSetosa

> iris.data$IsSetosa <-

ifelse(iris.data$Species == "setosa", "Y", "N")

> iris.data$IsSetosa <- ifelse(iris.data$Species == "setosa", "Y", "N")
```

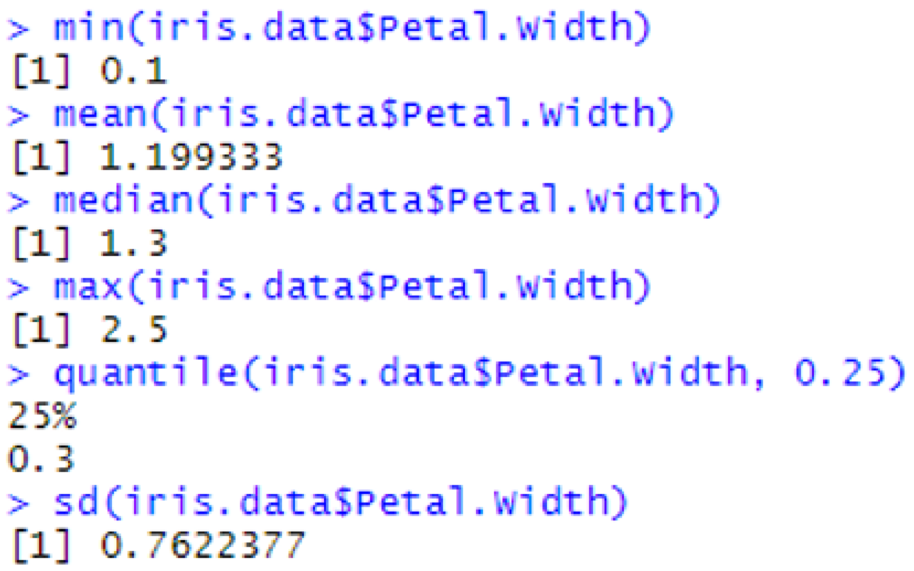
Common Functions

Like Excel, R comes out of the box with many, many functions to work with columns of data.

Many of the R functions share the same name with the corresponding Excel function. In other cases, mapping your Excel knowledge to R is straightforward, as is depicted below.



E	F	G	H
Species			
setosa			
setosa		MIN(iris.data[Petal.Width])	0.1
setosa		AVERAGE(iris.data[Petal.Width])	1.199333
setosa		MEDIAN(iris.data[Petal.Width])	1.3
setosa		MAX(iris.data[Petal.Width])	2.5
setosa		PERCENTILE.EXC(iris.data[Petal.Width], 0.25)	0.3
setosa		STDEV.S(iris.data[Petal.Width])	0.762238

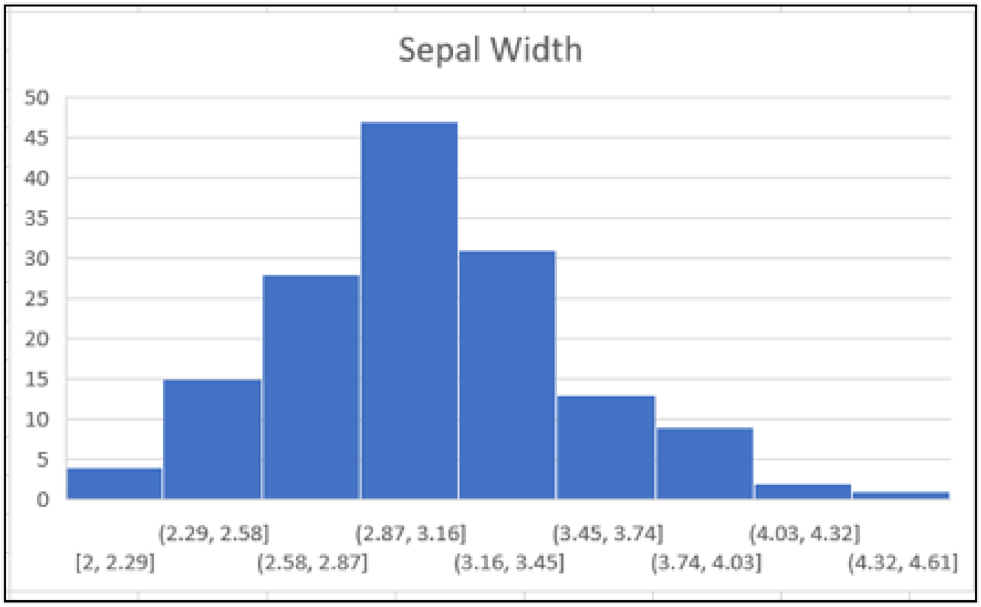


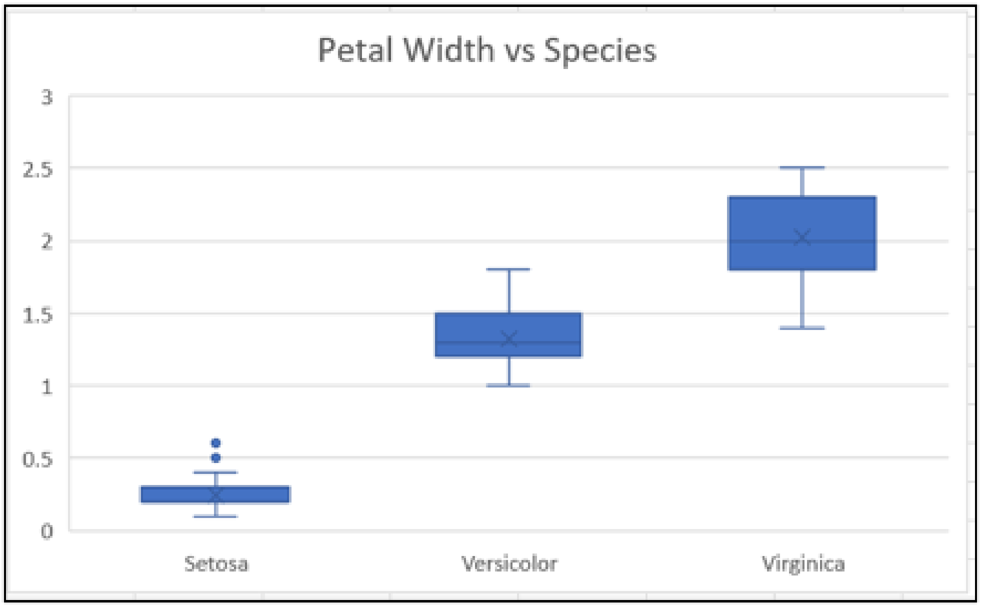
```
> min(iris.data$Petal.Width)
[1] 0.1
> mean(iris.data$Petal.Width)
[1] 1.199333
> median(iris.data$Petal.Width)
[1] 1.3
> max(iris.data$Petal.Width)
[1] 2.5
> quantile(iris.data$Petal.Width, 0.25)
25%
0.3
> sd(iris.data$Petal.Width)
[1] 0.7622377
```

Awesome Data Visualizations

Excel Data Visualizations

Excel is a great data visualization tool, supporting many ways to analyze data visually.



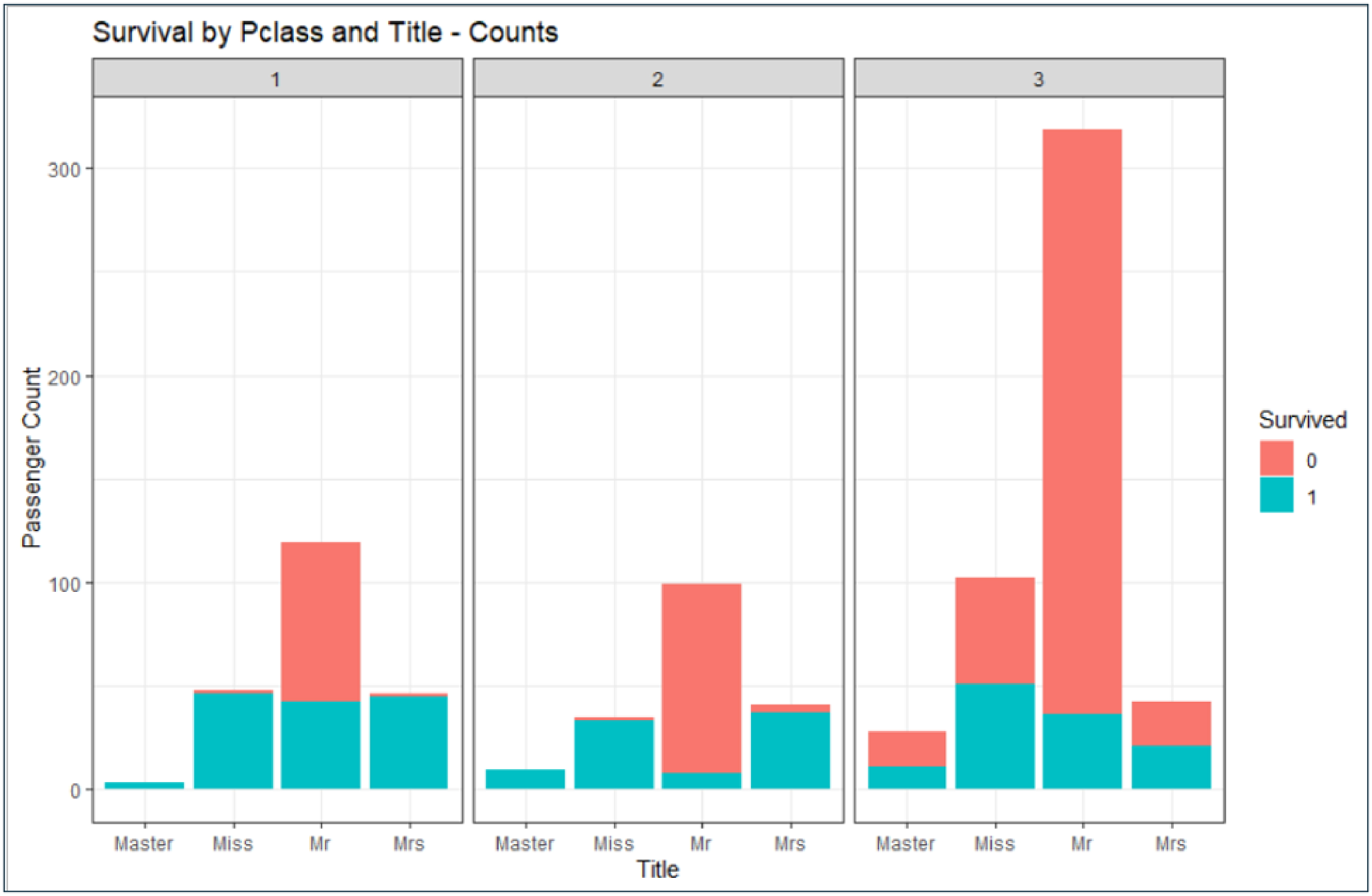


R Data Visualizations

R is renowned for it's ability to create print-quality data visualizations.

R also easily produces data visualizations that are difficult, or not possible to do with out of the box Excel features.

Analyzing data visually is one of R's specialities.



Ready to Empower Your Team?

If your team has Excel skills, I can teach them R programming and unlock advanced analytics. I've successfully taught 1000s of professionals from diverse backgrounds analytics skills that include:

- R programming
- Business analysis
- Linear regression
- Logistic regression
- Machine learning

My most popular offering is the "From Excel to Machine Learning Bootcamp."

This is an intensive 3-day transformation designed to take ANY team from analyzing data using Excel pivot tables to analyzing data using machine learning in R.

This transformation is applicable to any business domain. The skills your team will learn can be used to answer diverse questions like:

[Human Resources] - What factors are highly predictive of high performers?
[Product Mgmt] - What feature usage(s) are highly predictive of a sticky customer?
[Customer Service] - What customer behaviors predict churn?
[Marketing] - What are the attributes/behaviors that predict conversion?
[Supply Chain] - What product characteristics predict failure?

Let's talk. Email me: dave@daveondata.com

About the Author

My name is Dave Langer and I am the founder of Dave on Data.

I'm a hands-on analytics professional, having used my skills with Excel, SQL, and R to craft insights, advise leaders, and shape company strategy.

I'm also a skilled educator, having trained 100s of working professionals in a live classroom setting and 1000s more via my online courses and tutorials.

In the past, I've held analytics leaderships roles at Schedulicity, Data Science Dojo, and Microsoft.

Drop me an email if you have any questions: dave@daveondata.com