Lean Excel

Top Functions

By Scott Ratliff

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Quick reference guide with over 500 examples for the most used Excel functions!

Lean Excel: Top Functions

By Scott Ratliff

www.LeanExcelBooks.com

Legal Stuff

Introduction

Knowing how to use the built-in functions in Microsoft Excel will turn you into a power user! There are hundreds of functions and at first, the list can seem overwhelming. Even as a Microsoft Office Expert in Excel 2013 I do not know all of the Excel functions available. In writing this eBook, I discovered several new functions that I have now incorporated into my daily use and have made my data analysis more efficient. I hope that this eBook will help you learn more of the Excel functions and that they would help you gain control over your spreadsheets!

Why Lean Excel?

This *Lean Excel* series focuses on time-saving tools that will help you reduce the waste associated with data analysis. One of the tenets of lean manufacturing theory is using specialized tools to accomplish a task. You could use a wrench to drive a nail but a hammer will get the job done much faster. Remember time is money.

Think of the built-in functions as specialized tools. Let's say you needed to find the average of a dataset that is in A1 through A5. Remember the average is the sum of all the numbers in a dataset divided by the count of numbers in that dataset. Let's look at three ways to find the average. First, you could use the formula =(A1+A2+A3+A4+A5)/5. Second, you could use the SUM and COUNT functions =SUM(A1:A5)/COUNT(A1:A5). Or lastly, you could use the AVERAGE function =AVERAGE(A1:A5). Although all three options give you the same answer, using the AVERAGE function is clearly the most efficient. But in order to use the AVERGE function, you need to know that it exists and what requirements you need to use it.

Entering Formulas and Functions

First, let's identify the similarities between formulas and functions. Both formulas and functions can be entered using the formula bar immediately below the Excel Ribbon. The Excel Ribbon is the horizontal bar near the top of the Excel window that contains all of the buttons for tools available in Excel. Both formulas and functions begin with the equals sign (=). This is required only at the beginning of the formula bar; it is not required before each individual function within that cell. The equals sign tells Excel that you want Excel to do a calculation for this cell. If you did not have the equals sign, then Excel will put exactly what you type into that cell.

Next, let's identify the difference between a function and a formula. In the first section of this book, we discussed 3 ways to find the average of five numbers. The first option is an example of a formula. It begins with an equals sign (=) and includes mathematic operators (addition: +, subtraction: -, multiplication: *, division: /). As seen in the first option =(A1+A2+A3+A4+A5)/5, it can include both references to cells (A1, A2, A3, etc.) and actual numbers (the 5 after the division sign). A formula can include other mathematic operators such as parentheses () to determine the order of operations and ^ for exponents. Excel follows the same order of operations you learned in algebra: Parentheses then exponents then multiplication and division and lastly addition and subtraction. So if you entered =(2+3)^2/5+1 into a cell in Excel, it would first do what was in the parentheses (2+3) = 5. Next it would look for exponents so (5)^2 = 25. Then it would look for multiplication or division 25/5 = 5 and lastly addition or subtraction 5+1 = 6. So 6 would be the result of the formula and be displayed in the cell.

A function is entered very similarly but uses a built-in keyword that Excel will recognize. The third option from the previous section uses the AVERAGE function. To enter the function, first put an equals sign (=) then type the function name. Excel will try to recognize the keyword you are typing and provide you a list of the functions that begin with what you typed. You can either continue typing or double-click on the function you see in the list. When you type the keyword, upper case is not required; however, if Excel identifies the keyword as a function, then it will convert it to upper case. Alternatively, you could click on the Formula tab on the Excel ribbon and choose a function from the categories listed there. For reference consistency, the functions included in this eBook are divided to match the categories on the Formulas tab of the Excel ribbon.

Each function begins with the equals sign (=) then the keyword (AVERAGE, COUNT, SUM, etc.) and then an open parentheses (. After the open parentheses, Excel expects a list of arguments. This is the most confusing part of using a function. Each function requires a different set of arguments and the arguments must match certain criteria or the function will return an error (#ERR, #N/A, #DIV/0, etc.). In the third option above, the AVERAGE function required some number, list of numbers, or range as an argument. We entered the range A1:A5 for the argument. We assumed that range contained valid number data, if any cell in that range A1:A5 contained something besides a number, then our function may have resulted in an error.

You can also use formulas and functions together. For example, the second option of the previous section is two functions (SUM and COUNT) in a formula (SUM/COUNT). We could use any of the mathematic operators on the result of any function as long as the result of that function is a number. Some functions return non-numeric data so that may result in an error.

Function Arguments

A few more details we need to discuss before we get to the functions. In the previous section, we discussed how a function begins with an equals sign then a keyword, then an open parentheses, then the arguments, and lastly the closing parentheses. Some functions such as TODAY() requires no arguments. Other arguments such as SUM, COUNT, AVERAGE require one argument but then allow for many more arguments. When you enter a function, excel shows you which argument you are currently entering with the pop-up box that appears below the formula bar. For example, for the SUM function, the pop-up box displays SUM(number1, [number2], …). The number1 means it is required but the brackets [] around the [number2] indicate it is optional. Note the comma separating the arguments; when you enter a function, you will need to separate the arguments by a comma too. The (…) means that there could be many more arguments that match the type of the last argument. In this case, there could be many more numbers ([number3], [number4], etc).

Some functions have an optional pair of arguments. For example, the SUMIFS, COUNTIFS, and AVERAGEIFS functions all have the arguments [criteria_rangeX, criteriaX] where X is the next number in series. Since the bracket is around the pair or arguments, the pair is optional but if you want to include the argument, then you must include both arguments of the pair.

Most arguments could be fulfilled by either a value or a cell reference. Most experienced Excel users put all of the arguments in cells and then reference the cells within the arguments of the function. The example workbooks are built this way. With this method, you can change the value in the reference cells and the function will then change instead of having to go into the formula bar and change the function directly. I hope using this method will help you understand the relationship between the inputs and outputs of the function as both components are visible.

Example Workbook

To really understand a topic, most people need to visualize it. Therefore, all of the functions discussed in this eBook are included in a workbook with over 500 examples. You can find the example workbook and Visual Aid printouts at the link below.

To help visualize the inputs, some functions that require a range have the range address saved in a cell. Then in the function, I've added the INDIRECT function to convert the range address saved in the cell to the actual range of the function. For example, let's say B5 contains the range address "A2:A4" and A2 through A4 contains all numbers. If I was to enter the function =SUM(B5), it would return 0 because B5 contains the text string "A2:A4". In order to convert the text "A2:A4" into an address, we need to add the INDIRECT function. So the function should then be =SUM(INDIRECT(B5)) which is equivalent to =SUM(A2:A4) since B5 contains "A2:A4". The examples are set up this way so you can see the address of the ranges that go into the functions. To change the address, change it in the cell with the address (in the most recent example, that cell would be B5).

Find the example workbook here:
https://www.dropbox.com/sh/uvk1a6pdwjlrtjl/AAB7CXA0QlDBMQ9-Ps6iQ4Aka?dl=0

Function Guide

This Function Guide lists the keyword then the arguments and description for each function. Whenever the description references the arguments, the argument name will be in **bold**. Most descriptions will begin with the word “returns.” The arguments are the input of the function and then the function *returns* the output. The output is visible in the cell while the function is still visible in the formula bar or by double clicking in the cell.

Some functions in the function guide must be entered as an array. To enter a function as an array, hold down the control and shift buttons while pressing enter (for mac users, hold down control and shift while pressing return). You will know the function was entered as an array by the curly brackets {} around the function in the formula bar. Do not enter the curly brackets manually.

Date Functions
Using Dates in Excel

Excel stores dates as numbers called serial numbers. Zero corresponds to 1/1/1904. The number 100 refers to 100 days since 1/1/1904. Times are stored as partial days. So 12:00 noon is half the day so it is stored as 0.5 added to the date.

Adding and Subtracting Dates and Times

To add or subtract a date, just add or subtract whole numbers. To add or subtract a time, add or subtract the partial day. If you wanted to see what time it was after 5 hours, it is easier and more exact to add 5/24 to a date than it is to remember that 5/24 = .208333333.

DATE

Arguments: year, month, day

Description: Returns the serial number of the date of the specified **year**, **month**, and **day**.

DAY

Arguments: serial_number

Description: Returns the day of the month (1-31) of the **serial_number**.

DAYS

Arguments: end_date, start_date

Description: Returns the number of days between **end_date** and **start_date**. Equivalent to **end_date** - **start_date**.

EDATE

Arguments: start_date, months

Description: Returns the serial number of the date that is the specified number of months before or after the **start_date**. If **months** is less than 0 then it returns months before; if greater than 0 then months after.

EOMONTH

Arguments: start_date, months

Description: Returns the serial number of the last day of the month that is the specified number of months before or after the **start_date**. If **months** is less than 0 then it returns months before; if greater than 0 then months after.

HOUR

Arguments: serial_number

Description: Returns the hour (0-23) of the **serial_number**.

MINUTE

Arguments: serial_number

Description: Returns the minute (0-59) of the **serial_number**.

MONTH

Arguments: serial_number

Description: Returns the month (1-12) of the **serial_number**.

NETWORKDAYS

Arguments: start_date, end_date, [holidays]

Description: Returns the number of full work days between **start_date** and **end_date**. Specify the dates of any holidays to exclude.

NOW

Arguments: none

Description: Returns the computer’s date and time.

SECOND

Arguments: serial_number

Description: Returns the second (0-59) of the **serial_number**.

TIME

Arguments: hour, minute, second

Description: Returns the serial number of the time of the specified **hour**, **minute**, and **second**.

TODAY

Arguments: none

Description: Returns the computer’s date.

WEEKDAY

Arguments: serial_number, [return_type]

Description: Returns a number (1-7) of the day in the week for the **serial_number**. Use **return_type** to determine how to count the days.

WEEKNUM

Arguments: serial_number, [return_type]

Description: Returns the week number of the year for the **serial_number**. Use **return_type** to determine how to count the weeks.

WORKDAY

Arguments: start_date, days, [holidays]

Description: Returns the serial number of the number of **days** after the **start_date**. Specify the dates of any holidays to exclude.

YEAR

Arguments: serial_number

Description: Returns the year of the **serial_number**.

Financial Functions
Notes for Financial Functions

There are a few things in mind when using the functions and their arguments described below. Be sure to keep units consistent for **rate** and **nper**. If monthly payments then **rate** will be APR/12 and **nper** is number of years of loan*12. For annual payments, then **rate** will be APR and **nper** is number of years. Cash deposited into savings is represented by negative numbers and cash received by positive numbers. If **type** is 0 or omitted, then payments are made at the end of the period. If **type** is 1 then payments are made at the beginning of the period.

FV

Arguments: rate, nper, pmt, [pv], [type]

Description: Returns the future value of an investment of **pv** at interest rate **rate** based on the number **nper** of constant payments pmt.

PV

Arguments: rate, nper, pmt, [fv], [type]

Description: Returns the present value of an investment with target future value **fv** at interest rate **rate** based on the number **nper** of constant payments **pmt**.

RATE

Arguments: nper, pmt, pv, [fv], [type], [guess]

Description: Returns the interest rate per period of a loan/investment of size **pv** with **nper** payments of constant value **pmt**. **Fv** is the cash balance after the last payment. If **fv** is omitted, 0 is assumed. **Guess** is the guess of what the interest rate should be. If **guess** is omitted, it is assumed to be 10 percent.

PMT

Arguments: rate, nper, pv, [fv], [type]

Description: Returns the payment per period of a loan/investment of size **pv** with **nper** payments with an interest rate **rate**. **Fv** is the cash balance after the last payment. If **fv** is omitted, 0 is assumed.

Logical Functions

AND

Arguments: logical1, logical2, …

Description: Returns TRUE if all logical arguments are TRUE.

FALSE

Arguments: none

Description: Returns the value FALSE.

IF

Arguments: logical_test, value_if_true, value_if_false

Description: If **logical_test** returns TRUE then return the **value_if_true**, otherwise return the **value_if_false**.

IFERROR

Arguments: value, value_if_error

Description: If **value** is an error (#N/A, #DIV/0, etc.) then return the **value_if_error**, otherwise returns **value**.

IFNA

Arguments: value, value_if_na

Description: If **value** is #N/A then returns the **value_if_na**, otherwise returns value.

ISBLANK

Arguments: value

Description: If **value** is blank then returns TRUE, if not then returns FALSE.

ISERR

Arguments: value

Description: If **value** is an error other than #N/A then returns TRUE, if not then returns FALSE.

ISERROR

Arguments: value

Description: If **value** is an error (#N/A, #DIV/0, etc.) then returns TRUE, if **value** is not an error then returns FALSE.

ISNUMBER

Arguments: value

Description: If **value** is a number then returns TRUE, if not then returns FALSE.

NOT

Arguments: logical

Description: If **logical** would return TRUE then return FALSE or if FALSE then return TRUE.

OR

Arguments: logical1, logical2, …

Description: Returns TRUE if at least one logical argument is TRUE.

TRUE

Arguments: none

Description: Returns the value TRUE.

Lookup Functions
ADDRESS

Arguments: row_num, column_num, [abs_num], [a1], [sheet_text]

Description: Returns a cell reference as text from **column_num** and **row_num**. Use **abs_num** to identify absolute or relative references and **a1** for A1 or R1C1 style and **sheet_text** to specify the sheet name.

CHOOSE

Arguments: index_num, value1, [value2], …

Description: Returns a value (**value1**, **value2**, …) based on a choice for **index_num**. Like an if statement with more options. **Index_num** cannot be greater than the number of values.

COLUMN

Arguments: [reference]

Description: Returns a number of the column of the **reference**. If **reference** is left blank then returns the column of the cell containing the formula.

COLUMNS

Arguments: array

Description: Returns the number of columns in the range **array**.

FORMULATEXT

Arguments: reference

Description: Returns the formula that is stored in **reference** as a text string.

GETPIVOTDATA

Arguments: data_field, pivot_table, [Field1, item1], [Field2, item2], …

Description: Returns a value from the **data_field** in the pivot table **pivot_table** which meets all of the criteria of **item** in **Field**.

HLOOKUP

Arguments: lookup_value, table_array, row_index_num, [range_lookup]

Description: Finds **lookup_value** in the top row of the range **table_array** and then returns the value from the same column in the **row_index_num** row of that table. For example, if you want to return the third row then **row_index_num** = 3. If **range_lookup** is FALSE, then the value in the top row must match **lookup_value** exactly. If **range_lookup** is TRUE, then it will match the closest value to **lookup_value** without going over.

HYPERLINK

Arguments: link_location, [friendly_name]

Description: Returns the text of **friendly_name** as a hyperlink to **link_location**. If **friendly_name** is left blank, then the text of **link_location** will be returned as a hyperlink.

INDEX (array)

Arguments: array, row_num, [column_num]

Description: Returns the cell reference at the intersection of **row_num** and **column_num** within **array**. Enter as an array function with either **row_num** or **column_num** as 0 (zero) to return the entire row or column.

INDEX (reference)

Arguments: reference, row_num, [column_num], [area_num]

Description: Returns the cell reference at the intersection of **row_num** and **column_num** within **reference**. If **reference** contains multiple ranges, then **area_num** can be used to identify which range. If **row_num** or **column_num** is 0 (zero) then returns the entire row or column.

INDIRECT

Arguments: ref_text, [a1]

Description: Returns the reference specified by the text string **ref_text**. If **a1** is TRUE or omitted, then **ref_text** is interpreted as an A1-style reference. If not, then it is interpreted as an R1C1-style reference.

MATCH

Arguments: lookup_value, lookup_array, [match_type]

Description: Looks for **lookup_value** within a **lookup_array** of cells and returns the position of that item in the range. If **match_type** is 1 or omitted, then returns the largest value that is less than or equal to **lookup_value** and values in **lookup_array** must be sorted in ascending order. If **match_type** is 0, then returns the first value that is exactly equal to **lookup_value** and values can be sorted in any order. If **match_type** is -1, then returns the smallest value that is greater than or equal to **lookup_value** and values in **lookup_array** must be sorted in descending order.

OFFSET

Arguments: reference, rows, cols, [height], [width]

Description: Returns the reference to the range that is **rows** rows and **cols** columns away from **reference**. Use **height** and **width** to specify the number of rows and columns returned.

ROW

Arguments: [reference]

Description: Returns a number of the row of the **reference**. If **reference** is left blank then returns the row of the formula.

ROWS

Arguments: array

Description: Returns the number of rows in the range **array**.

VLOOKUP

Arguments: lookup_value, table_array, col_index_num, [range_lookup]

Description: Finds **lookup_value** in the first column of the range **table_array** and then returns the value from the same row in the **col_index_num** column of that table. For example, if you want to return the third column then **col_index_num** = 3. If **range_lookup** is FALSE, then the value in the first column must match **lookup_value** exactly. If **range_lookup** is TRUE, then it will match the closest value to **lookup_value** without going over and table_array must be sorted in ascending order.

Math Functions

ABS

Arguments: number

Description: Returns the absolute value of **number**.

CONVERT

Arguments: number, from_unit, to_unit

Description: Returns the result of converting **number** in **from_unit** to the equivalent value in **to_unit**.

DELTA

Arguments: number1, [number2]

Description: Returns TRUE (1) or FALSE (0) if **number1** is equal to **number2**.

EXP

Arguments: number

Description: Returns the value of e (Euler’s number) raised to the power **number**.

FACT

Arguments: number

Description: Returns the value of **number**‘s factorial.

INT

Arguments: number

Description: Returns **number** rounded down the nearest integer.

LOG

Arguments: number, [base]

Description: Returns the logarithm with base **base** of **number**.

LOG10

Arguments: number

Description: Returns the common logarithm (base 10) of **number**.

LN

Arguments: number

Description: Returns the natural logarithm of **number**.

MOD

Arguments: number, divisor

Description: Returns the remainder of **number** divided by **divisor**.

MROUND

Arguments: number, multiple

Description: Returns **number** rounded to the nearest **multiple**.

PI

Arguments: none

Description: Returns the value of Pi.

POWER

Arguments: number, power

Description: Returns **number** raised to the exponent **power**.

PRODUCT

Arguments: number1, [number2], …

Description: Returns the value of all **number** arguments multiplied together.

RAND

Arguments: none

Description: Returns a random number between 0 and 1.

RANDBETWEEN

Arguments: bottom, top

Description: Returns a random number between **bottom** and **top**.

ROUND

Arguments: number, num_digits

Description: Returns **number** rounded to the decimal place specified by **num_digits**. If **num_digits** is positive, then places to the right of the decimal. If negative, then places to the left of the decimal.

ROUNDDOWN

Arguments: number, num_digits

Description: Returns **number** rounded down to the decimal place specified by **num_digits**. If **num_digits** is positive, then places to the right of the decimal. If negative, then places to the left of the decimal.

ROUNDUP

Arguments: number, num_digits

Description: Returns **number** rounded up to the decimal place specified by **num_digits**. If **num_digits** is positive, then places to the right of the decimal. If negative, then places to the left of the decimal.

SIGN

Arguments: number

Description: Returns 1 if **number** is positive, -1 if negative, or 0 if 0.

SQRT

Arguments: number

Description: Returns the square root of **number**.

SUBTOTAL

Arguments: function_num, ref1, [ref2], …

Description: Returns the subtotal of a list or database with **ref1**, **ref2**, etc. The value of the subtotal can change based on the **function_num**.

SUM

Arguments: number1, [number2], …

Description: Returns the value of all **number** arguments added together.

SUMIF

Arguments: range, criteria, [sum_range]

Description: Returns the value of **sum_range** added together if the corresponding value in **range** meets the **criteria**.

SUMIFS

Arguments: sum_range, criteria_range1, criteria1, [criteria_range2, criteria2],…

Description: Returns the value of **sum_range** added together if all of the corresponding values in all **criteria_range** meets all **criteria**.

SUMPRODUCT

Arguments: array1, [array2], …

Description: Returns the sum of corresponding values in **arrays** multiplied together.

SUMSQ

Arguments: number1, [number2], …

Description: Returns the sum of the **number** arguments squared. **Number** arguments are squared first and then added together.

TRUNC

Arguments: number, [num_digits]

Description: Returns **number** truncated or cut off to the number of decimal places in **num_digits**. If **num_digits** is left blank, then number is truncated to an integer. Truncation does not involve rounding.

Statistical Functions

AVERAGE

Arguments: number1, [number2], …

Description: Returns the arithmetic mean of the **number** arguments.

AVERAGEIF

Arguments: range, criteria, [average_range]

Description: Returns the arithmetic mean of the **average_range** if the corresponding value in **range** meets the **criteria**.

AVERAGEIFS

Arguments: average_range, criteria_range1, criteria1, [criteria_range2, criteria2], …

Description: Returns the arithmetic mean of the **average_range** if all of the corresponding values in **criteria_range** meet the **criteria**.

CORREL

Arguments: array1, array2

Description: Returns the correlation coefficient between the two data sets **array1** and **array2**.

COUNT

Arguments: value1, [value2], …

Description: Returns the number of cells in a range (**value1**, **value2**, …) with numbers.

COUNTA

Arguments: value1, [value2], …

Description: Returns the number of cells in a range (**value1**, **value2**, …) that are not empty.

COUNTBLANK

Arguments: range

Description: Returns the number of cells in **range** that are empty.

COUNTIF

Arguments: range, criteria

Description: Returns the number of cells in a **range** that meet the specified **criteria**.

COUNTIFS

Arguments: criteria_range1, criteria1, [criteria_range2, criteria2], …

Description: Returns the number of cells that meet all of the specified **criteria** in their respective **criteria_ranges**.

FORECAST

Arguments: x, known_y’s, known_x’s

Description: Returns a future value at point **x** along a linear trend using existing data (**known_y’s** and **known_x’s**).

F.TEST

Arguments: array1, array2

Description: Returns the result of the F-Test which measures the probability that the variances of two **arrays** are not significantly different.

INTERCEPT

Arguments: known_y’s, known_x’s

Description: Returns the point at which a line will intersect the y-axis using linear regression of **known_y’s** and **known_x’s**.

LARGE

Arguments: array, k

Description: Returns the **k**-th largest value in the range **array**.

MAX

Arguments: number1, [number2], …

Description: Returns the largest value in a range (**number1**, **number2**, …).

MEDIAN

Arguments: number1, [number2], …

Description: Returns the median of the values in a range (**number1**, **number2**, …).

MIN

Arguments: number1, [number2], …

Description: Returns the smallest value in a range (**number1**, **number2**, …).

MODE.MULT

Arguments: number1, [number2], …

Description: Returns a vertical array of the most commonly occurring values in the **number1**, **number2**, … Highlight the number of cells in a column to return, then press control-shift-enter simultaneously to enter as an array function.

MODE.SNGL

Arguments: number1, [number2], …

Description: Returns the most common value in the **number1**, **number2**, … array.

PEARSON

Arguments: known_y’s, known_x’s

Description: Returns the Pearson Product Moment Coefficient of **known_y’s** and **known_x’s**. The Pearson Product Moment Coefficient is abbreviated R.

PERCENTILE.INC

Arguments: array, k

Description: Returns the value located at the **k**th percentile of an **array**. PERCENTILE.EXC is slightly more accurate than PERCENTILE.INC but it will only work if k is between 1/n and 1-1/n where n is the number of elements in array.

PERCENTILE.EXC

Arguments: array, k

Description: Returns the value located at the **k**th percentile of an **array**.
PERCENTILE.INC is slightly less accurate than PERCENTILE.EXC but it will work for a value of k between 0 and 1.

PERCENTRANK.INC

Arguments: array, x, [significance]

Description: Returns the percentage rank of a value **x** in an **array**. Inclusive of the data set.

PERCENTRANK.EXC

Arguments: array, x, [significance]

Description: Returns the percentage rank of a value **x** in an **array**. Exclusive of the data set.

QUARTILE.EXC

Arguments: array, quart

Description: Returns the exclusive **quart** (0-4) quartile of the range **array**.

QUARTILE.INC

Arguments: array, quart

Description: Returns the inclusive **quart** (0-4) quartile of the range **array**.

RANK.AVG

Arguments: number, ref, [order]

Description: Returns the rank of a **number** in range **ref** sorted ascending or descending based on **order**. If multiple ranks are found, the average rank is returned. If **order** is left blank, the order will be ascending.

RANK.EQ

Arguments: number, ref, [order]

Description: Returns the rank of a **number** in range **ref** sorted ascending or descending based on **order**. If multiple ranks are found, the top rank is returned. If **order** is left blank, the order will be ascending.

RSQ

Arguments: known_y’s, known_x’s

Description: Returns the square of the Pearson Product Moment Coefficient of **known_y’s** and **known_x’s**. Known as R-squared.

SLOPE

Arguments: known_y’s, known_x’s

Description: Returns the slope of a line using linear regression of **known_y’s** and **known_x’s**.

SMALL

Arguments: array, k

Description: Returns the **k**-th smallest value in the range **array**.

TRIMMEAN

Arguments: array, percent

Description: Returns the arithmetic mean of the interior **percent** of range **array**.

T.TEST

Arguments: array1, array2, tails, type

Description: Returns the probability of a Student’s T-Test comparing the average of **array1** and **array2** with number of tails and type.

Z.TEST

Arguments: array, x, [sigma]

Description: Returns the one-tailed value of a Z-Test of **array** with value **x** and sigma level **sigma**.

Text Functions
What is a string?

In computer lingo, a series of text characters is called a string. A string can be composed of letters, numbers, or symbols.

CLEAN

Arguments: text

Description: Returns **text** with all non-printable characters removed.

CONCATENATE

Arguments: text1, text2, etc.

Description: Returns the combination of multiple **text** strings.

EXACT

Arguments: text1, text2

Description: Returns TRUE if two **text** strings are exactly the same (including upper/lower case).

FIND

Arguments: find_text, within_text, start_num

Description: Finds one string (**find_text**) within another string (**within_text**). Can specify where to start looking with **start_num**.

LEFT

Arguments: text, num_chars

Description: Returns the leftmost **num_chars** of the **text** string.

LOWER

Arguments: text

Description: Returns **text** with all letters in lower case.

MID

Arguments: text, start_num, num_chars

Description: Returns the **num_chars** of the **text** string starting at **start_num**.

PROPER

Arguments: text

Description: Returns **text** with the first letter in each word in upper case and the rest to lower case.

REPLACE

Arguments: old_text, start_num, num_chars, new_text

Description: Replaces **num_chars** characters starting at **start_num** of **old_text** with **new_text**.

RIGHT

Arguments: text, num_chars

Description: Returns the rightmost **num_chars** of the **text** string.

SEARCH

Arguments: find_text, within_text, start_num

Description: Finds one string (**find_text**) within another string (**within_text**). Can specify where to start looking with **start_num**. SEARCH is more versatile than FIND. FIND is case sensitive where SEARCH is not. SEARCH also accepts the wildcard characters ? for single and * for multiple characters.

TEXT

Arguments: value, format_text

Description: Returns **value** reformatted according to the **format_text** specified.

TRIM

Arguments: text

Description: Returns **text** with all spaces removed except for a single space between words.

UPPER

Arguments: text

Description: Returns **text** with all letters in upper case.

About the Author

Scott Ratliff holds a Bachelor's of Science degree in Materials Science and Engineering from the University of Kentucky. He is also a certified Microsoft Excel 2013 Expert. He is working towards his Six Sigma Black Belt accreditation with a focus on Lean Manufacturing Improvements. He lives in Kentucky with his wife and two daughters. Find his entire list of Lean Excel Books on his website www.LeanExcelBooks.com.

Other Books by Scott Ratliff

Lean Excel: Dynamic Charts

Eliminate unnecessary steps and draw data-based conclusions faster by creating dynamic charts in Microsoft Excel! Charts that update automatically when you add or delete data. This eBook contains a detailed procedure on how to create dynamic charts as well as a full explanation of each step for your learning. Also included is a downloadable template with three types of dynamic charts in place so you can see the final result as well as a printable one-page visual instruction guide to keep at your desk to have any time you need it! Increase your productivity immediately! Requires Microsoft Excel 97 or newer. Available on both Amazon Kindle and iBooks.

Alleluia: Family Worship Notebook

Use the **Alleluia: Family Worship Notebook** to help you and your family grow closer to God and more knowledgeable about His Word! The included tools provide a structured format you and your family can use to record, review, and retain the worship services of your local church. Different level notebook pages are included for grade 1 through adults. Family Bible study questions are included to help start or restart your family Bible studies. Available on both Amazon Kindle and iBooks.