Notes: Data Analysis in Spreadsheets

Created by Wenxiao Zhou

1. What's in a cell?

Data types for data science:

The 4 common data types

- Numbers (`123.456`)
- Text (`"Hello!"`)
- Dates (`1900-02-29`)
- Logical (`TRUE`)

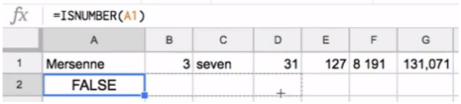
Other "IS" functions for checking types

- `ISTEXT()` checks for text.
- `ISDATE()` checks for dates and times.
- `ISLOGICAL()` checks for `TRUE` and `FALSE` .

Checking cell properties and rarer types

- `ISURL()` checks if text is web URL.
- `ISFORMULA()` checks if the cell contains a formula.

Copying formulas between cells



Drag the bottom-right corner of the cell to copy. Alternatively,

- to copy right, use CTRL + R (CMD + R on macOS).
- to copy down, use CTRL + D (CMD + D on macOS).

Command+Enter 可以实现一次性粘贴 formulas

Checking rarer data types

Google Sheets also allows checks for rarer data types beyond number, text, date, and logical. In this exercise you'll use `ISURL()` to check for website URLs, and `ISFORMULA()` to check for cells that contain formulas.

`ISURL()` checks that the text in a cell takes the form of a URL. This includes text that starts with a protocol, like `http://` , `ftp://` and `mailto:` , and guesses at other text like `somewebsite.com` .

To make the text into a clickable hyperlink, use `HYPERLINK()` . This takes the address of a cell containing a URL and optionally the address of a cell containing the text to display, for example `HYPERLINK(A1, B1)` .

`ISFORMULA()` takes the address of a cell and returns `TRUE` when that cell contains a formula. For blank cells and cells with fixed values, it returns `FALSE` .

Finding missing data

`ISBLANK()` accepts a cell address and returns `TRUE` if that cell is empty. This provides a useful way of checking for missing data.

Logical conditions like `ISBLANK()` are useful for filtering datasets: you can keep only the rows where some condition is `TRUE` . Data filtering can be done using `FILTER()` , which takes two arguments. The first argument is the range of the data that you want to filter, and the second argument is the range containing the logical values to filter on.

For example, if the data (not including the header row) is in `A2:E10` , and the final column `E` contains the logical values to filter on, you would type `=FILTER(A2:E10, E2:E10)` .

Exercise:

Instructions 70XP

Some athletes have gone missing! An extra column, "Athlete Is Blank?" has been added to the dataset.

- Use the `ISBLANK()` function in column `H` to determine which values in the dataset have missing athletes.
- In cell `A23`, call `FILTER()` to filter the dataset to show only the rows with missing athletes.
  - Pass the range of all the data from `A2` to `H20` as the first argument.
  - The 2nd argument should be the logical values contained in `H2` to `H20`.

Time (s)	Wind (m/s)	Athlete	Nationality	Location of race	Date	Is Cheater?	Athlete Is Blank?
9.930	1.4	Calvin Smith	United States	Colorado Springs, USA	1983-07-03	FALSE	FALSE
9.830	1.0		Canada	Rome, Italy	1987-08-30	TRUE	TRUE
9.930	1.1	Carl Lewis	United States	Rome, Italy	1987-08-30	FALSE	FALSE
9.930	1.1	Carl Lewis	United States	Zürich, Switzerland	1988-08-17	FALSE	FALSE
9.790	1.1		Canada	Seoul, South Korea	1988-09-24	TRUE	TRUE
9.900	1.9	Leroy Burrell	United States	New York, USA	1991-06-14	FALSE	FALSE
9.860	1.0	Carl Lewis	United States	Tokyo, Japan	1991-08-25	FALSE	FALSE
9.850	1.2	Leroy Burrell	United States	Lausanne, Switzerland	1994-07-06	FALSE	FALSE
9.835	0.7		Canada	Atlanta, USA	1996-07-27	FALSE	TRUE
9.790	0.1	Maurice Greene	United States	Athens, Greece	1999-06-16	FALSE	FALSE
9.780	2.0	Tim Montgomery	United States	Paris, France	2002-09-14	TRUE	FALSE
9.768	1.6	Asafa Powell	Jamaica	Athens, Greece	2005-06-14	FALSE	FALSE
9.766	1.7	Justin Gatlin	United States	Doha, Qatar	2006-05-12	TRUE	FALSE
9.763	1.5	Asafa Powell	Jamaica	Gateshead, England	2006-06-11	FALSE	FALSE
9.762	1.0	Asafa Powell	Jamaica	Zürich, Switzerland	2006-08-18	FALSE	FALSE
9.740	1.7	Asafa Powell	Jamaica	Rieti, Italy	2007-09-09	FALSE	FALSE
9.720	1.7	Usain Bolt	Jamaica	New York, USA	2008-05-31	FALSE	FALSE
9.683	0.0	Usain Bolt	Jamaica	Beijing, China	2008-08-16	FALSE	FALSE
9.572	0.9	Usain Bolt	Jamaica	Berlin, Germany	2009-08-16	FALSE	FALSE

Time	Wind	Athlete	Nationality	Location of race	Date	Is Cheater?	Athlete Is Blank?
9.830	1.0		Canada	Rome, Italy	1987-08-30	TRUE	TRUE
9.790	1.1		Canada	Seoul, South Korea	1988-09-24	TRUE	TRUE
9.835	0.7		Canada	Atlanta, USA	1996-07-27	FALSE	TRUE

Summary

- `TO*()` functions format numbers.
- `N()` changes text to numbers.
- `IF(logical, 1, 0)` changes logicals to numbers.
- `CONVERT()` changes the units of a number.

I2 `=to_percent(H2)`

	A	B	C	D	E	F	G	H	I
1	Time (s)	Wind (m/s)	Athlete	Nationality	Location of race	Date	Is Cheater?	Ratio of best time	Percentage of best time
2	9.930	1.4	Calvin Smith	United States	Colorado Springs, USA	1983-07-03	FALSE	1.037400752	104%

2. Working with numbers

Converting logical values to numbers

In order to calculate on logical values, it is often useful to pretend that `TRUE` is `1` and `FALSE` is `0`. Just as with text, to perform the conversion, you can use the `N()` function. This allows you to count the number or proportion of true values.

- The `SUM()` of the ones gives you the count of true values.
- The `AVERAGE()` of the ones and zeros gives you the proportion of true values.

Preaching to the CONVERT()ed

`CONVERT()`, changes the units of a number. For times, you have five choices of unit to switch between, including

- `"sec"`: seconds
- `"min"`: minutes

For speeds, the units include

- `"m/s"`: meters per second
- `"mph"`: miles per hour

`CONVERT()` takes three arguments: a number or cell address, the existing unit (in double quotes), and the unit to convert to. For example, `CONVERT(1, "hr", "sec")` returns 3600.

Exponential transformations

To undo the logarithmic transformations from the last exercise, you need to perform an exponential transformation.

- The "power of" operator, `^`, raises a number to a power. For example, `=10 ^ A1` is the opposite of `=LOG10(A1)`.
- `EXP()` handles the special case of raising Euler's number to a particular power, and is the opposite of `=LN()`.

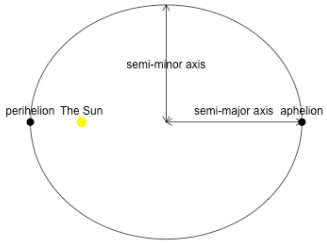
Summary

- `LOG10()` and `LN()` perform logarithmic transformations.
- `10 ^ x` and `EXP()` perform exponential transformations.
- `SQRT()` performs square root transformations.

Square root transformations

There are many more mathematical transformations that you can apply to your data. In fact, it's possible to create very complex transformations by building them step-by-step.

Asteroids have roughly elliptical orbit around the sun, as shown in this image.



Using the distance from the center of the ellipse to the aphelion and the perihelion, you can can calculate the area of the orbit.

To solve this, you'll need:

- `SQRT()` calculates the square root of a number.
- `PI()` returns the mathematical constant.

Instructions 100XP

- In column `H`, calculate the semi-major axis as the aphelion (column `F`) plus perihelion (column `G`), all divided by two.
- In column `I`, calculate the semi-minor axis as the square-root of the product of the aphelion and the perihelion.
- In column `J`, calculate the area as pi times the semi-major axis (column `H`) times the semi-minor axis (column `I`).

Rounding and formatting numbers:

Ceiling and floor (1)

	A	B	C
1	*Value*	*Command*	*Result*
2	`1234.5678`	`=CEILING(A2)`	`1235`
3		`=FLOOR(A2)`	`1234`
4	`-1234.5678`	`=CEILING(A4)`	`-1234`
5		`=FLOOR(A4)`	`-1235`

Ceiling and floor (2)

	A	B	C
1	*Value*	*Command*	*Result*
2	`1234.5678`	`=CEILING(A2, 0.01)`	`1234.57`
3		`=FLOOR(A2, 0.01)`	`1234.56`
4		`=CEILING(A2, 100)`	`-1300`
5		`=FLOOR(A2, 100)`	`-1200`

Summary

- `ROUND(x, n)` rounds `x` to the nearest `n` decimal places.
- `CEILING(x, y)` rounds `x` up to the nearest multiple of `y`.
- `FLOOR(x, y)` rounds `x` down to the nearest multiple of `y`.

Exercises:

`ROUND()` lets you round numbers to a specified number of decimal places.

- `ROUND(A1)` rounds the number in cell `A1` to the nearest whole number.
- `ROUND(A1, 3)` rounds it to three decimal places - or, in other words, to the nearest thousand**th**.
- `ROUND(A1, -3)` rounds it to the nearest thousand.

From floor to ceiling

Sometimes you always want to round a value down (towards negative infinity), or always round upwards (towards infinity). You can do this using

- `FLOOR()`, which rounds down, and
- `CEILING()`, which round up.

Both functions take a second argument that specifies the multiple to round to. For example, `FLOOR(A1, 0.01)` rounds the number in cell A1 down to the next lowest hundredth. If this argument is omitted, it rounds down to the nearest whole number.

Rounding negative numbers

`FLOOR()` and `CEILING()` will round negative numbers towards or away from negative infinity. That is, `FLOOR(-1.5)` is `-2` and `CEILING(-1.5)` is `-1`.

Sometimes you may wish to round them towards or away from zero.

Google Sheets has two related functions called `FLOOR.MATH()` and `CEILING.MATH()`. When given one or two arguments, they behave in the same way as `FLOOR()` and `CEILING()` respectively. However, you can pass a third argument that determined the direction of the rounding: passing a positive number (for example, `1`) to a third argument to make them round in the positive direction - towards zero.

That is, `FLOOR.MATH(-1.57, 0.1, 1)` is `-1.5` and `CEILING.MATH(-1.57, 0.1, 1)` is `-1.6`.

I2 | fx | =FLOOR.MATH(H2,0.01,1)

	A	B	C	D	E	F	G	H	I	J
1	**Asteroid name**	**Date of closest approach**	**Nominal geocentric distance (km)**	**Size (m, lower)**	**Size (m, upper)**	**Aphelion (AU)**	**Perihelion (AU)**	**Perihelion minus aphelion**	**Toward zero**	**Away from zero**
2	99942 Apophis	2029-04-13	38300	270	325	1.0985	0.7461	-0.3524	-0.35	-0.4
3	2007 UW1	2129-10-19	100200	75	170	1.0174	0.798	-0.2194	-0.21	-0.3

Generating random numbers:

Summary

- `RAND()` generates random Uniform numbers between 0 and 1.
- `RANDBETWEEN()` generates random Uniform integers between two limits.
- `NORMINV(RAND())` generates random normal numbers.
- `*INV(RAND())` generates random numbers from other distributions.

Exercise

Generating uniform random numbers

Many data science tasks involve running simulations. One important step in a simulation is the generation of random numbers. There are two functions available for generating numbers from a uniform distribution. In a continuous uniform distribution, any number within a range is likely to be generated. In a discrete uniform distribution, any one of a finite number of values is equally likely to be generated.

- `RAND()` generates a random number between 0 and 1 from a continuous uniform distribution. It takes no arguments.
- `RANDBETWEEN()` lets you specify the lower and upper bounds, and generates a random integer (no fractional part); that is, it samples from a discrete uniform distribution.

I2 | fx | =RANDBETWEEN(D2,E2)

	A	B	C	D	E	F	G	H	I
1	Asteroid name	Date of closest approach	Nominal geocentric distance (km)	Size (m, lower)	Size (m, upper)	Aphelion (AU)	Perihelion (AU)	From 0 to 1	From lower to upper
2	99942 Apophis	2029-04-13	38300	270	325	1.0985	0.7461	0.45389291	277
3	2007 UW1	2129-10-19	100200	75	170	1.0174	0.798	0.71634691	154

Generating random numbers from other distributions

As an example, to generate random numbers from a normal distribution with mean 3 and standard deviation 2, you would use `=NORMINV(RAND(), 3, 2)`.

There are many other inverse cumulative distribution functions available: you can repeat that same code swapping `FINV()` for the F distribution, `BETAINV()` for the beta distribution, and so on.

Instructions 100XP

- In column `H`, generate normally distributed random geocentric distances with the mean taken from column `C` and a standard deviation of 1000.
- In column `I`, generate beta distributed random asteroid sizes.
  - Call `BETAINV()`, passing `RAND()` as the first argument.
  - Set the second and third "shape" arguments to 2.
  - Set the fourth and fifth "bound" arguments to the asteroid lower and upper size estimates.

H2 | fx | =NORMINV(RAND(), C2, 1000)

	A	B	C	D	E	F	G	H	I
1	Asteroid name	Date of closest approach	Nominal geocentric distance (km)	Size (m, lower)	Size (m, upper)	Aphelion (AU)	Perihelion (AU)	Normal random geocentric distance	Beta random size
2	99942 Apophis	2029-04-13	38300	270	325	1.0985	0.7461	38429.0041	301.031701
3	2007 UW1	2129-10-19	100200	75	170	1.0174	0.798	100338.675	102.530671

3. Logic & Errors

Logical Operations:

The opposite of true

	A	B	C
1	*Value*	*Command*	*Result*
2	TRUE	=NOT(A2)	FALSE
3	FALSE	=NOT(A3)	TRUE

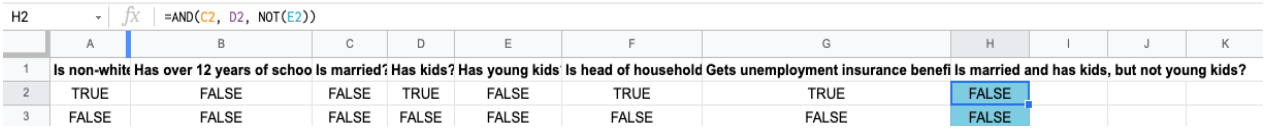
This and that

	A	B	C	D
1	*Value1*	*Value2*	*Command*	*Result*
2	TRUE	TRUE	=AND(A2, B2)	TRUE
3	TRUE	FALSE	=AND(A3, B3)	FALSE
4	FALSE	TRUE	=AND(A4, B4)	FALSE
5	FALSE	FALSE	=AND(A5, B5)	FALSE

Summary

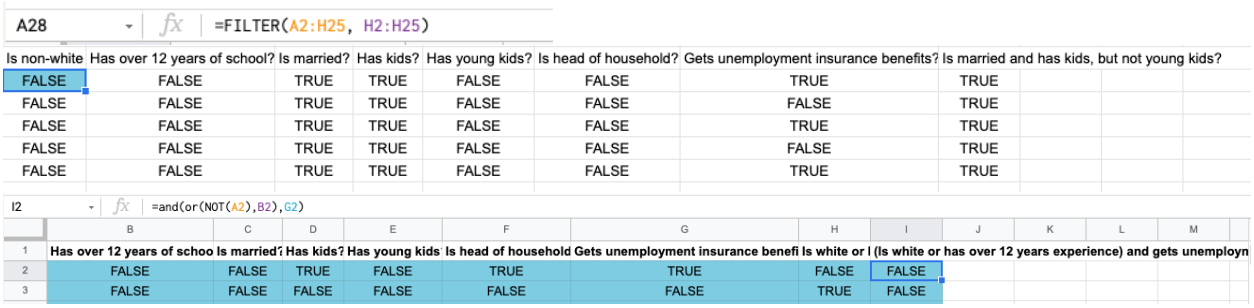
- `NOT()` swaps `TRUE` and `FALSE`.
- `AND()` returns `TRUE` when all inputs are `TRUE`.
- `OR()` returns `TRUE` when any inputs are `TRUE`.

Exercise:



	A	B	C	D	E	F	G	H
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Is married and has kids, but not young kids?
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Filter the rows where column H is TRUE:



Is non-white	Has over 12 years of school?	Is married?	Has kids?	Has young kids?	Is head of household?	Gets unemployment insurance benefits?	Is married and has kids, but not young kids?
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE

	B	C	D	E	F	G	H	I
1	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Is white or l	(Is white or has over 12 years experience) and gets unemployn
2	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE

Flow Control

Dealing with lots of conditions

	A	B	C	D
1	*Value1*	*Value2*	*Command*	*Result*
2	TRUE	FALSE	=IFS(A2, "1st", B2, "2nd")	1st
3	FALSE	TRUE	=IFS(A3, "1st", B3, "2nd")	2nd
4	TRUE	TRUE	=IFS(A4, "1st", B3, "2nd")	1st
5	FALSE	FALSE	=IFS(A5, "1st", B5, "2nd")	#N/A

Transforming categorical variables

	A	B	C
1	*Value*	*Command*	*Result*
2	1st	=SWITCH(A2, "1st", 1, "2nd", 2)	1
3	2nd	=SWITCH(A3, "1st", 1, "2nd", 2)	2
4	3rd	=SWITCH(A4, "1st", 1, "2nd", 2)	#N/A

Summary

- `IF(condition, yes, no)` lets you return a value based on a logical condition.
- `IFS(condition1, value1, condition2, value2)` extends this to multiple conditions.
- `SWITCH(condition, category1, value1, category2, value2)` lets you transform categorical variables.

H2 | =if(AND(C2,E2),"married with kids","other")

	A	B	C	D	E	F	G	H
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Domestic status
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	other
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	other

H2 | =IFS(E2,"young",NOT(D2),"none",AND(D2,NOT(E2)),"old")

	A	B	C	D	E	F	G	H
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Kids status
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	old
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	none

J2 | =switch($H$2:$H$25,"none",1,"young",2,"old",3)

	A	B	C	D	E	F	G	H	I	J
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Kids status	Has kids? 2	Kids score
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	old	TRUE	3
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	none	FALSE	1

Blanks, missing values, & errors

Types of error

Error	Cause
`#DIV/0!`	Dividing by zero.
`#VALUE!`	Nonsense data in a calculation.
`#REF!`	Referencing a cell that has been deleted.
`#NAME?`	Forgetting to quote a string.
`#NUM!`	Numbers being out of range.
`#N/A`	Missing value.
`#ERROR!`	Syntax problem in a formula.

Summary

- Cells with nothing in are called "blank".
- Calculating with blank cells will give you the wrong answer.
- Instead, use `NA()` to create missing values.
- Missing values are a type of error.

How you deal with these blank cells can have a big effect on your results, so it's important to tread carefully. The first steps are to able to identify whether a cell is blank (using `ISBLANK()`), and to count how many blanks that you have.

`COUNTBLANK` accepts a range of cells, and returns the number of blanks in that range.

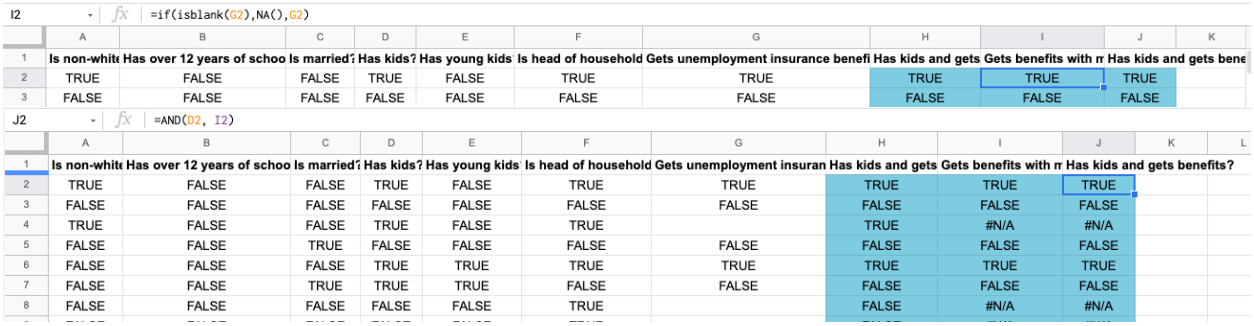
Going missing

Some calculations involving blank values may give different results to what you might expect. For example, when you pass a blank value into the `AND()` function, it is treated as `TRUE`. This is often unhelpful. To make blanks behave in a sensible way in calculations, you must first convert them to be "not available" using `NA()`. This function takes no inputs, and returns a missing value. To convert a blank value to a missing value, use this pattern.

```
=IF(ISBLANK(cell), NA(), cell)
```

Instructions 100XP

- In column `H`, use `AND()` to find women who have kids and get benefits.
- In column `I`, convert the blanks in column `G` to missing values.
- In column `J`, again find women who have kids and get benefits, this time using column `I` rather than `G`.



I2 =if(isblank(G2),NA(),G2)

	A	B	C	D	E	F	G	H	I	J	K
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insurance benefi	Has kids and gets	Gets benefits with π	Has kids and gets bene	
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	

J2 =AND(D2, I2)

	A	B	C	D	E	F	G	H	I	J	K	L
1	Is non-white	Has over 12 years of schoo	Is married?	Has kids?	Has young kids	Is head of household	Gets unemployment insuran	Has kids and gets	Gets benefits with π	Has kids and gets benefits?		
2	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE		
3	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
4	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE		TRUE	#N/A	#N/A		
5	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
6	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE		
7	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE		
8	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE		FALSE	#N/A	#N/A		

Exercise

Errors and omissions

Technically speaking, the missing value created by `NA()` is a type of error. You can test for errors using `ISERROR()`. Similar to the other `IS*()` functions that you've seen, it takes a cell address as its input and returns `TRUE` if that cell contains an error and `FALSE` otherwise.

Treating missing values as errors is often undesirable since real-world data naturally contains missing values. That's why there is also `ISERR()` that returns `TRUE` for all error types except missing values.

If apply ISERROR() to blank: blank is not considered as an error
If apply ISERROR() to missing values: missing values are considered as errors
If apply ISERR() to missing values: missing values are not considered as errors, or other types of error will return FALSE

Exercise:

What errors will these formulas generate?

1. `=1 / 0`
2. `=SQRT(-1)`
3. `=Z1 = value`
4. `=1 + " "`
5. `=SUM(0 1)`

◉ `#DIV/0!`, `#NUM!`, `#NAME?`, `#VALUE!`, `#ERROR!`

4. Positional Matching

Cell addresses

Addresses

	A	B
1	*Command*	*Result*
2	`=ROW(C5)`	`5`
3	`=COLUMN(C5)`	`3`
4	`=ADDRESS(B2, B3)`	`$C$5`
5	`=ADDRESS(B2, B3, 4)`	`C5`

Summary

- There are two ways of specifying a cell address.
- `ROW()` & `COLUMN()` return integer positions.
- `ADDRESS()` returns the address in `A1` format text.
- `INDIRECT()` returns the value an an address.
- `OFFSET()` find returns values relative to another cell.
- `INDEX()` finds values within a cell range.

Working with cell addresses

The address of a cell in Google sheets can be specified in two ways: either as a letter-number pair like `C5`, or as a pair of numbers like "row 5, column 3". Some formulas may be easier to specify on one way or the other, so it's useful to know how to convert from one form to the other and back.

- `ROW()` and `COLUMN()` take addresses in `A1` format and return the number of the row and column respectively.
- `ADDRESS()` takes numeric row and column inputs and returns the address in `A1` format. It also has an option argument to specify the relativity of the addresses: 1 (the default) returns absolute addresses; 2, 3, and 4 return row absolute, column absolute, and relative addresses respectively.

In this chapter you'll be working with Indian butterfly data from **Singh and Pandey**.

K2 | =address(H2,I2,4)

	A	B	C	D	E	F	G	H	I	J	K
1	Area	Locality	Skipper	Swallowtail	White-yellow	Blue	Brush-footed	Row	Column	Absolute	Relative
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	2	2	$B$2	B2
3	Western Himalaya	Western Himalaya	63	31	42	129	152	3	2	$B$3	B3

From addresses to values

A really important use of the cell addresses you made using `ADDRESS()` is to get the value stored at that location. To do this, you use the `INDIRECT()` function. (The name of the function comes from the computer science concept of *indirection*, where you pass the address of a variable in memory to a function rather than the variable itself.)

If the cell `J42` contains the value `99`, then typing `=ADDRESS(42, 10)` in cell `J43` will give that cell a value of `$J$42`. Now the formula `=INDIRECT("J43")` will return the value `99`.

I2:I45 | =indirect(H2)

	A	B	C	D	E	F	G	H	I
1	Area	Locality	Skipper	Swallowtail	White-yellow	Blue	Brush-footed	Absolute Addre	Value from Address
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	$G$2	482
3	Western Himalaya	Western Himalaya	63	31	42	129	152	$G$3	152

Finding nearby cells with offsets

Sometimes you want to calculate things based upon cells close by to the cell you are providing a calculation in.

`OFFSET()` retrieves the values in cells offset from the current location by a certain number of rows and columns. It takes two arguments: the number of rows down to move from the current location, and the number of columns to move right.

You can also specify `height` and `width` arguments to return a range of cells, which is often useful in combination with a summary statistic function like `SUM()`, `AVERAGE()` or `MAX()`.

Instructions 100XP

- In cell `I1`, get the count of `Skipper`s in the `Indian Subcontinent` as the value from the cell offset 1 down from `C1`.
- In cell `I2`, again get the count of `Skipper`s in the `Indian Subcontinent`, this time as the value from the cell offset 2 right from `A2`.
- In cell `I3`, get the count of `Skipper`s in `Western Himalaya` as the values from the range of cells **offset** 2 right from `A3`, with height 7, and **sum** them.

I2 `=OFFSET(A2, 0, 2)`

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**C1, offset 1 column down**	**307**
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	A2, offset 2 columns right	307
3	Western Himalaya	Western Himalaya	63	31	42	129	152	A3, offset 2 columns right, height 7, summed	271
4	Western Himalaya	Kangra Hills	25	23	37	56	87		
5	Western Himalaya	Shimla Hills	41	21	34	88	115		
6	Western Himalaya	Dehradun Valley	22	11	19	42	54		
7	Western Himalaya	Mussoorie Hills	54	23	32	88	126		
8	Western Himalaya	Mussoorie Town	14	10	13	44	65		
9	Western Himalaya	Kumaon Hills	52	26	37	109	147		
10	Central Himalaya	Central Himalaya	125	43	49	185	221		

I3 `=SUM(OFFSET(A3, 0, 2, 7, 1))`

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**C1, offset 1 column down**	**307**
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	A2, offset 2 columns right	307
3	Western Himalaya	Western Himalaya	63	31	42	129	152	A3, offset 2 columns right, height 7, summed	271
4	Western Himalaya	Kangra Hills	25	23	37	56	87		
5	Western Himalaya	Shimla Hills	41	21	34	88	115		
6	Western Himalaya	Dehradun Valley	22	11	19	42	54		
7	Western Himalaya	Mussoorie Hills	54	23	32	88	126		
8	Western Himalaya	Mussoorie Town	14	10	13	44	65		
9	Western Himalaya	Kumaon Hills	52	26	37	109	147		
10	Central Himalaya	Central Himalaya	125	43	49	185	221		
11	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342		
12	North East India + North Myanmar	North East India	189	62	52	258	292		
13	North East India + North Myanmar	Sikkim	159	55	51	162	263		

Exercise

Local addresses

The "A1" address system has coordinates that exist over the whole worksheet. If you have a block of data specified somewhere within that worksheet, it can be useful to be able to specify the addresses relative to that block.

This can be done with `INDEX()`, which takes 3 arguments. The first argument is a rectangular range of data, for example `A2:E8`. The second and third arguments are numbers specifying an offset down then right from the top left of that data range. Unlike `OFFSET()`, the numbering starts at 1, so `INDEX(A2:E8, 4, 2)` refers to cell `B5`.

Instructions 100XP

Use the block of data for `North East India + North Myanmar`, from `A11` to `G19` as the reference.

- In cell `I1`, use `INDEX()` to get the number of `Blue`s in `Sikkim`.
- In cell `I2`, get the number of `White-yellow`s in `Mizoram Hills`.

I2 `=index(A11:G19,9,5)`

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**Blues in Sikkim**	**162**
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	White-yellows in Mizoram Hills	26
3	Western Himalaya	Western Himalaya	63	31	42	129	152		
4	Western Himalaya	Kangra Hills	25	23	37	56	87		
5	Western Himalaya	Shimla Hills	41	21	34	88	115		
6	Western Himalaya	Dehradun Valley	22	11	19	42	54		
7	Western Himalaya	Mussoorie Hills	54	23	32	88	126		
8	Western Himalaya	Mussoorie Town	14	10	13	44	65		
9	Western Himalaya	Kumaon Hills	52	26	37	109	147		
10	Central Himalaya	Central Himalaya	125	43	49	185	221		
11	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342		
12	North East India + North Myanmar	North East India	189	62	52	258	292		
13	North East India + North Myanmar	Sikkim	159	55	51	162	263		
14	North East India + North Myanmar	Darjeeling	27	29	32	48	126		
15	North East India + North Myanmar	Naga Hills	67	38	30	110	178		
16	North East India + North Myanmar	Manipur	119	18	1	126	57		
17	North East India + North Myanmar	Khasia Hills	98	42	36	128	207		
18	North East India + North Myanmar	Khasia + Jaintia Hills	132	49	40	166	209		
19	North East India + North Myanmar	Mizoram Hills	59	13	26	77	101		

Lookups & Matching

	A	B	C	D
1	*id*	*nspec*		
2	Ant	10k		
3	Fly	125k		
4	Bee	20k		
5				
6	*id*	*size*	*command*	*nspec*
7	Ant	25	=VLOOKUP(A7, $A$2:$B$4, 2, FALSE)	10k
8	Bee	40	=VLOOKUP(A8, $A$2:$B$4, 2, FALSE)	20k
9	Moth	300	=VLOOKUP(A9, $A$2:$B$4, 2, FALSE)	#N/A

Spreadsheets have the ability to perform what's known as a left join using the VLOOKUP() function. This is a little complex, and the function takes four arguments.

The first argument to VLOOKUP() is the value that you want to match. Let's try matching Ant. That's in cell A7 in the table we're merging into, so that's the first argument. The second argument is the range of the dataset that we want to find values in, given as absolute addresses. In this case that dataset stretches from A2 to B4. The third argument is the number of the column that contains the values to be merged in. In this case, we want the number of species from column B, the second column. The fourth argument is whether or not the lookup column is sorted. Usually this will be FALSE.

Programmatically sorting data

```
=SORT($A$2:$C$5, 2, FALSE)
```

The SORT() function takes 3 arguments. First is the dataset, given as absolute addresses. Second is the number of the column to sort by. Finally, you pass TRUE to sort in ascending order and FALSE to sort in descending order.

Matching values

```
=MATCH(100000, $B$2:$B$5, -1)
```

MATCH() lets you find values in a sorted dataset. It takes three arguments. First is the value to find. In the example, you are looking for one hundred thousand. Second is the absolute address of the column of data, here from B2 to B5. Finally, you pass one if the data is sorted in ascending order or minus one if the data is sorted in descending order.

Summary

- `VLOOKUP()` preforms left joins on two datasets.
- `SORT()` programmatically sorts data.
- `MATCH()` finds positions in sorted data where a value would occur.

A VLOOKUP refresher

In **Data Analysis with Spreadsheets** you learned how to use `VLOOKUP()`. They are important, so let's refresh your memory.

`VLOOKUP()` is like `INDEX()` in that it lets you look up values within a data block. It has the advantage that rather than you having to look and find the position of the cell to specify manually, you can specify the value you are looking for, and it automatically finds it. `VLOOKUP()` takes four arguments.

- First is the value that you are looking for, usually a string.
- Second is the data range, usually specified using absolute coordinates. The first column must contain the lookup values.
- Third is the column offset, the same as with `INDEX()`.
- Fourth is whether or not the data is sorted by the lookup column. Usually you need to specify `FALSE` here.

Instructions 100XP

Find the same values from the `INDEX()` exercise, now using `VLOOKUP()`. Your data range is from `$B$11` to `$G$19`.

- In cell `I1`, get the number of `Blues` in `Sikkim`. Call `VLOOKUP()`, passing `"Sikkim"`, the data range, an offset of `5`, and `FALSE`.
- In cell `I2`, get the number of `White-Yellow`s in `Mizoram Hills`.

I1 =VLOOKUP("Sikkim",$B$11:$G$19,5,FALSE)

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**Blues in Sikkim**	162
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	White-yellows in Mizoram Hills	26
3	Western Himalaya	Western Himalaya	63	31	42	129	152		
4	Western Himalaya	Kangra Hills	25	23	37	56	87		
5	Western Himalaya	Shimla Hills	41	21	34	88	115		
6	Western Himalaya	Dehradun Valley	22	11	19	42	54		
7	Western Himalaya	Mussoorie Hills	54	23	32	88	126		
8	Western Himalaya	Mussoorie Town	14	10	13	44	65		
9	Western Himalaya	Kumaon Hills	52	26	37	109	147		
10	Central Himalaya	Central Himalaya	125	43	49	185	221		
11	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342		
12	North East India + North Myanmar	North East India	189	62	52	258	292		
13	North East India + North Myanmar	Sikkim	159	55	51	162	263		
14	North East India + North Myanmar	Darjeeling	27	29	32	48	126		
15	North East India + North Myanmar	Naga Hills	67	38	30	110	178		
16	North East India + North Myanmar	Manipur	119	18	1	126	57		
17	North East India + North Myanmar	Khasia Hills	98	42	36	128	207		
18	North East India + North Myanmar	Khasia + Jaintia Hills	132	49	40	166	209		
19	North East India + North Myanmar	Mizoram Hills	59	13	26	77	101		
20	Other parts of India	Kolkata	32	10	19	57	49		

Matching values

The `MATCH()` function let's you find the position of cells that match a particular criterion. It's a little tricky, so bear with this.

It works best when the data is already sorted (that's the case we'll consider here). The first argument is the limit value, the second argument is the data range, and the third argument is `1` if the column is sorted in ascending order and `-1` for descending.

If a column of data, `A2:A100` was sorted in ascending order, `MATCH(1000, A2:A100, 1)` would find the position of the largest value in `A2:A100` that was less than or equal to `1000`.

If `B2:B100` was sorted in descending order, `MATCH(1000, B2:B100, -1)` would find the position of the smallest value in `B2:B100` that was greater than or equal to `1000`.

The dataset has been sorted by descending number of `Skipper`s.

Instructions 100XP

- In cell `I1`, `MATCH()` the position of the smallest number of `Skipper`s greater than or equal to 100. The data range is `C2` to `C45`.
- In cell `I2`, get the `ADDRESS()` of that cell. The row is the match position plus one (for the header row), and it's the third column.
- In cell `I3`, get the value in that cell to find the smallest number of `Skipper`s greater than 100.

I1 =MATCH(100, $C$2:$C$45, -1)

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**match**	8
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	address	$C$9
3	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342	smallest number of skippers over 100	119
4	North East India + North Myanmar	North East India	189	62	52	258	292		

I2 =ADDRESS(I1 + 1, 3)

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**match**	8
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	address	$C$9
3	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342	smallest number of skippers over 100	119
4	North East India + North Myanmar	North East India	189	62	52	258	292		
5	South Myanmar	South Myanmar	189	50	40	272	237		
6	North East India + North Myanmar	Sikkim	159	55	51	162	263		

I3 | =INDIRECT(I2)

	A	B	C	D	E	F	G	H	I
1	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**match**	**8**
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	address	$C$9
3	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342	smallest number of skippers over 100	119
4	North East India + North Myanmar	North East India	189	62	52	258	292		

Sorted!

It is often easier to make sense of datasets when the rows are ordered in some way, for example, when the values in a column go from smallest to largest (or largest to smallest). You can sort datasets using the `SORT()` function.

The first argument to `SORT()` is the range of the dataset, not including the header row. The second argument is the number of the column to sort on, starting with 1 as the left-most column. Thirdly, you set the sort direction: pass `TRUE` to sort in ascending order (low to high, or A to Z) and `FALSE` for descending order. If you want to break ties by sorting on further columns, you can pass another column index and another direction for each column that you want to sort with.

For example, to sort the dataset in range `$D$2` to `$H$100` by decreasing `F` values then increasing `D` values, you would write `SORT($D$2:$H$100, 3, FALSE, 1, TRUE)`.

I2 | =SORT(A2:G45, 3, FALSE)

Three challenges:
1.Advanced Filtering

Exercise

Advanced filtering

Time to practice the logical operations and filtering techniques you learned about in Chapter 3.

Instructions 70XP

- In column `H`, define a logical condition where the count of each butterfly type is greater than 50.
- In cell `I2`, define a filter on the whole dataset (columns `A` to `G`), using the logical condition in column `H`.

H2:H45 | =if(AND(C2>50,D2>50,E2>50,F2>50,G2>50),TRUE,FALSE)

	B	C	D	E	F	G	H	I	J	K	L	M
1	**Locality**	**Skipper**	**Swallowtail**	**White-yellow**	**Blue**	**Brush-footed**	**Filter Condition**	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yellc**
2	Indian Subcontinent	307	94	99	458	482	TRUE	Indian Subco	Indian Subcontine	307	94	99
3	Western Himalaya	63	31	42	129	152	FALSE	North East In	North East India +	211	69	57
4	Kangra Hills	25	23	37	56	87	FALSE	North East In	North East India	189	62	52
5	Shimla Hills	41	21	34	88	115	FALSE	North East In	Sikkim	159	55	51
6	Dehradun Valley	22	11	19	42	54	FALSE					
7	Mussoorie Hills	54	23	32	88	126	FALSE					
8	Mussoorie Town	14	10	13	44	65	FALSE					

I2 | =filter(A2:G45,H2:H45)

	H	I	J	K	L	M	N	O
1	**Filter Condition**	**Area**	**Locality**	**Skipper**	**Swallowtail**	**White-yello**	**Blue**	**Brush-footed**
2	TRUE	Indian Subco	Indian Subcontine	307	94	99	458	482
3	FALSE	North East In	North East India +	211	69	57	284	342
4	FALSE	North East In	North East India	189	62	52	258	292
5	FALSE	North East In	Sikkim	159	55	51	162	263
6	FALSE							

2. Conditional Summary Statistics

Exercise

Conditional summary statistics

In **Data Analysis with Spreadsheets**, you saw how to use `COUNTIF()` to calculate summary statistics. Here's you'll take it one step further using the related `COUNTIFS()` function, which lets you pass multiple conditions to it.

Arguments to `COUNTIFS()` come in pairs: a range of values to filter on, best given as absolute addresses, and a condition. The condition is text consisting of

1. a value to match, or
2. a comparison operator (`=`, `<`, `>=`, etc.) and a number.

For example, `COUNTIFS(A2:A100, "DataCamp", B2:B100, ">10")` counts the number of values where column `A` matches `"DataCamp"` **and** column `B` is greater than `10`.

Instructions 100XP

- In cell `H2`, get the unique Area values using `UNIQUE()`.
- In column `I`, use `COUNTIF()` to get the count of each Area. Pass the data range from `A2` to `A45` as absolute addresses, and the filter criteria from `H2` to `H9`.
- In column `J`, use `COUNTIFS()` to get the count of each area with more than `20` Swallow-tails. the first two arguments are the same as the previous step, then add a condition for column `D` to be greater than `20`.

H2 `=UNIQUE(A2:A45)`

	A	B	C	D	E	F	G	H	I	J
1	Area	Locality	Skipper	Swallowtail	White-yellow	Blue	Brush-footed	Area	Count of Are	Count of Area with
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	Indian Subcontinent	1	1
3	Western Himalaya	Western Himalaya	63	31	42	129	152	Western Himalaya	7	5
4	Western Himalaya	Kangra Hills	25	23	37	56	87	Central Himalaya	1	1
5	Western Himalaya	Shimla Hills	41	21	34	88	115	North East India + North Myanmar	9	7
6	Western Himalaya	Dehradun Valley	22	11	19	42	54	Other parts of India	21	1
7	Western Himalaya	Mussoorie Hills	54	23	32	88	126	Sri Lanka	1	0
8	Western Himalaya	Mussoorie Town	14	10	13	44	65	North Myanmar	3	1
9	Western Himalaya	Kumaon Hills	52	26	37	109	147	South Myanmar	1	1
10	Central Himalaya	Central Himalaya	125	43	49	185	221			
11	North East India + North Myanmar	North East India + North Myanma	211	69	57	284	342			
12	North East India + North Myanmar	North East India	189	62	52	258	292			
13	North East India + North Myanmar	Sikkim	159	55	51	162	263			
14	North East India + North Myanmar	Darjeeling	27	29	32	48	126			
15	North East India + North Myanmar	Naga Hills	67	38	30	110	178			
16	North East India + North Myanmar	Manipur	119	18	1	126	57			
17	North East India + North Myanmar	Khasia Hills	98	42	36	128	207			
18	North East India + North Myanmar	Khasia + Jaintia Hills	132	49	40	166	209			
19	North East India + North Myanmar	Mizoram Hills	59	13	26	77	101			

I2 `=COUNTIF($A$2:$A$45,H2)`

	A	B	C	D	E	F	G	H	I	J
1	Area	Locality	Skipper	Swallowtail	White-yellow	Blue	Brush-footed	Area	Count of Are	Count of Area with
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	Indian Subcontinent	1	1
3	Western Himalaya	Western Himalaya	63	31	42	129	152	Western Himalaya	7	5
4	Western Himalaya	Kangra Hills	25	23	37	56	87	Central Himalaya	1	1

J2 `=COUNTIFS($A$2:$A$45,H2,$D$2:$D$45,">20")`

	A	B	C	D	E	F	G	H	I	J
1	Area	Locality	Skipper	Swallowtail	White-yellow	Blue	Brush-footed	Area	Count of Are	Count of Area with
2	Indian Subcontinent	Indian Subcontinent	307	94	99	458	482	Indian Subcontinent	1	1
3	Western Himalaya	Western Himalaya	63	31	42	129	152	Western Himalaya	7	5
4	Western Himalaya	Kangra Hills	25	23	37	56	87	Central Himalaya	1	1
5	Western Himalaya	Shimla Hills	41	21	34	88	115	North East India + North Myanmar	9	7
6	Western Himalaya	Dehradun Valley	22	11	19	42	54	Other parts of India	21	1

3. Simple imputation

Exercise

Simple imputation

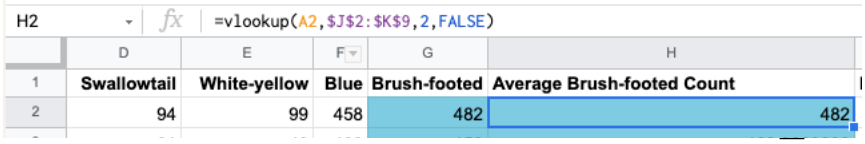
If you have missing data, you can use a set of techniques known as *imputation* to substitute guesses for those values. A very simple form of imputation is to substitute the average of the non-missing values for that group.

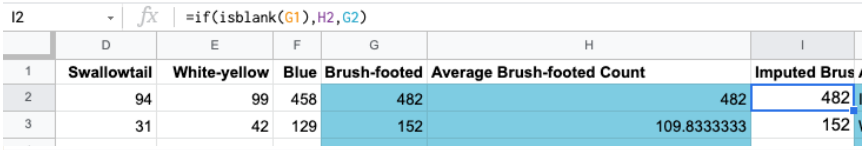
This technique involves 3 steps: use `AVERAGEIF()` to get the group averages, `VLOOKUP()` to join these back to the original dataset, and `ISBLANK()` to locate missing values.

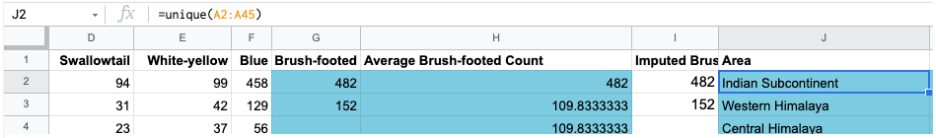
The dataset has been edited so some `Brush-footed` counts are missing.

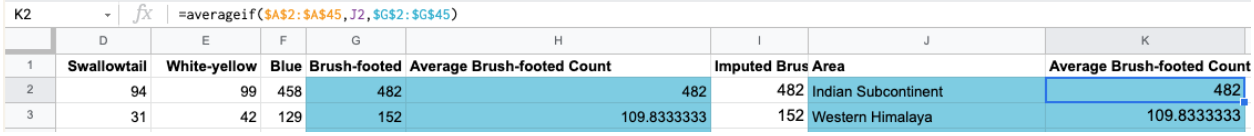
Instructions 100XP

- In cell `J2`, get the unique `Area`s.
- In column `K`, rows `2` to `9`, use `AVERAGEIF()` to calculate the average count of `Brush-footed`s by Area. Pass it the area data as absolute addresses, the unique areas, and the `Brush-footed` count data as absolute addresses.
- In column `H`, join the average counts back to the original dataset. `VLOOKUP()` takes 4 arguments. It needs the `Area` from column `A`, the data range of the table you just created (`J2` to `K9`) as absolute addresses, the column in the that table that contains the averages (`2`), and whether or not that table is sorted (`FALSE`).
- In column `I`, `IF()` column `G` is blank, take the value from column `H`, else take the value from column `G`.


H2
=vlookup(A2,$J$2:$K$9,2,FALSE)
D E F G H
1 Swallowtail White-yellow Blue Brush-footed Average Brush-footed Count
2 94 99 458 482 482


I2
=if(isblank(G1),H2,G2)
D E F G H I
1 Swallowtail White-yellow Blue Brush-footed Average Brush-footed Count Imputed Brus
2 94 99 458 482 482 482
3 31 42 129 152 109.8333333 152


J2
=unique(A2:A45)
D E F G H I J
1 Swallowtail White-yellow Blue Brush-footed Average Brush-footed Count Imputed Brus Area
2 94 99 458 482 482 482 Indian Subcontinent
3 31 42 129 152 109.8333333 152 Western Himalaya
4 23 37 56 109.8333333 Central Himalaya


K2
=averageif($A$2:$A$45,J2,$G$2:$G$45)
D E F G H I J K
1 Swallowtail White-yellow Blue Brush-footed Average Brush-footed Count Imputed Brus Area Average Brush-footed Count
2 94 99 458 482 482 482 Indian Subcontinent 482
3 31 42 129 152 109.8333333 152 Western Himalaya 109.8333333