Feeding Google Sheets/Spreadsheet

Importing and reshaping data with Google Sheets (Spreadsheets)

---

importHtml

Syntax: ImportHtml(URL, query, index)

URL is the URL of the HTML page.

Query is either “list” or “table” indicates what type of structure to pull in from the webpage. If it’s “list,” the function looks for the contents of <UL>, <OL>, or <DL> tags; if it’s “table,” it just looks for <TABLE> tags.

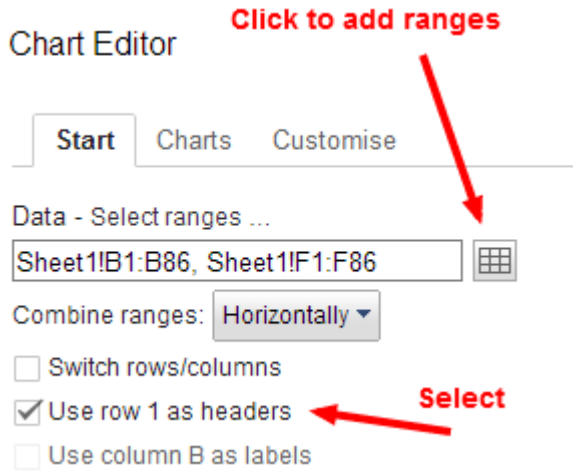
Index is the 1-based index of the table or the list on the source web page. The indices are maintained separately so there might be both a list #1 and a table #1.

Example: =ImportHtml(“http://en.wikipedia.org/wiki/Demographics_of_India“; “table”;4). This function returns demographic information for the population of India.

Note: The limit on the number of ImportHtml functions per spreadsheet is 50.

Exercise 1: Importing a html table and graphing the result

1. Create a new Google Spreadsheet
2. In cell A1 enter the formula to import a table from Wikipedia
   `=importHTML("http://en.wikipedia.org/wiki/2012_Summer_Olympics_medal_table","table",3)`
3. Select Insert > Chart and then select the data ranges for country name and total medals



4. While still in the Chart Editor select the Charts tab, then Maps > Geo charts - regions

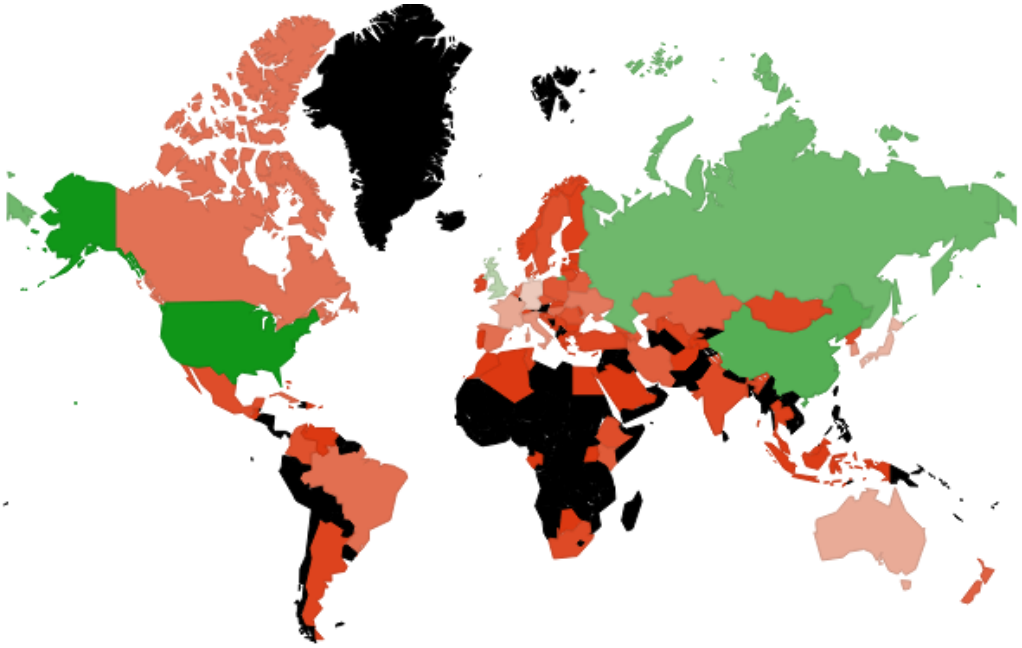
5. Still in the Chart Editor select Customise and change the No value colour to blank.
6. Finally click Insert

You should now have a chart that looks like this:

Notice that the chart has No values (black) for most of the countries. To fix this we need to remove the country codes in brackets. One way to do this is trim the text from the left until the first bracket "(". This can be done using a combination of the LEFT, FIND and ARRAYFORMULA (ARRAYFORMULA allows you to apply formulas to multiple cells).

1. In cell H2 enter the formula `=ARRAYFORMULA(LEFT(B2:B86,FIND("(",B2:B86)-2))` this should create a column of country names without brackets)
2. Click on your Chart and select Advanced edit.
3. Make sure you are on the Start tab in the Chart Editor and edit the data range for `Sheet1!B2:B86` to `Sheet1!H2:H86` then click Update

Your chart should now look like this (my solution http://goo.gl/8qUI9):



importFeed

Syntax: =ImportFeed(URL; [feedQuery | itemQuery]; [headers]; [numItems]). The arguments to the function are as follows:

URL is the url of the RSS or ATOM feed.

feedQuery/itemQuery is one of the following query strings: "feed", "feed title", "feed author", "feed description", "feed url", "items", "items author", "items title", "items summary", "items url", or "items created". The feed queries return properties of the feed as a whole: the feed's title, the feed's author, etc. **Note**: To get the data included in the feed, you need to do an "items" request.

- the "feed" query returns a single row with all of the feed information.
- the "feed <type>" query returns a single cell with the requested feed information.
- the "items" query returns a full table, with all of the item information about each item in the feed.
- the "items <type>" query returns a single column with the requested information about each item.
- if a query is given that begins with "feed", the numItems parameter isn't necessary and is replaced by the option headers param.
- if a query is given that begins with "items", the numItems parameter is expected as the 3rd param, and headers as the 4th.
- headers - "true" if column headers is desired. This will add an extra row to the top of the output labeling each column of the output.

Example: =ImportFeed("http://news.google.com/?output=atom")

Exercise 2: Importing a RSS feed and getting social share counts

1. Open http://goo.gl/aai2p
2. In cell B5 enter the RSS feed url for a blog (or you can use http://feeds.feedburner.com/MASHe) and hit enter

You should end up with something like:

Quick Feed Social/GA Page view counter

By @mhawksey. Read more about this at:
http://mashe.hawksey.info/2012/06/rss-feed-social-share-counting/

Feed url	http://feeds.feedburner.com/MASHe
Google Analytics Start Date	01/03/2011
Google Analytics End Date	24/10/2012

				Buzz	Reddit	StumbleL	Diggs	Pinterest	Delicious	LinkedIn	GooglePl	Facebook	Twitter	Comment	Page views
			Totals	0	0	2	0	0	9	1	50	10	212	20	#ERR
Title	**Author**	**URL**	**Date Created**												
OER File Formats: Tomorrow's problem?	Martin Hawksey	http://mashe.hawksey.info/2012/1	Tue, 23 Oct 2012 13								3	1	5	5	#ERR
Sketch of a cMOOC registration system	Martin Hawksey	http://mashe.hawksey.info/2012/1	Mon, 22 Oct 2012 1								2	1	3	4	#ERR
CFHE12 Week 2 Analysis: Data! Show me yo	Martin Hawksey	http://mashe.hawksey.info/2012/1	Sun, 21 Oct 2012 22								2	1	5		#ERR
JISC OER Rapid Innovation: Finding the techr	Martin Hawksey	http://mashe.hawksey.info/2012/1	Thu, 18 Oct 2012 12										2		#ERR
Summary of social monitoring tools and recip	Martin Hawksey	http://mashe.hawksey.info/2012/1	Wed, 17 Oct 2012 1								5	1	37	2	#ERR
Visualizing cMOOC data: Extracting and anal	Martin Hawksey	http://mashe.hawksey.info/2012/1	Tue, 16 Oct 2012 12						3		4	2	36	4	#ERR
CFHE12 Week 1 Analysis: Twitter isn't so ma	Martin Hawksey	http://mashe.hawksey.info/2012/1	Mon, 15 Oct 2012 0								3	1	6	1	#ERR
Any Questions? Filtering a Twitter hashtag co	Martin Hawksey	http://mashe.hawksey.info/2012/1	Tue, 09 Oct 2012 12						1		2	1	13	4	#ERR
PHP code to bulk collect social share counts	Martin Hawksey	http://mashe.hawksey.info/2012/1	Thu, 04 Oct 2012 11						1	1	3	2	6		#ERR
What I've starred this month: September 28, 2	Martin Hawksey	http://mashe.hawksey.info/2012/0	Sun, 30 Sep 2012 0										3		#ERR
Coursera #compdata week 1: Reflections and	Martin Hawksey	http://mashe.hawksey.info/2012/0	Fri, 28 Sep 2012 12			1			1		2		12		#ERR
New YouTube beta feature to add multiple ch	Martin Hawksey	http://mashe.hawksey.info/2012/0	Thu, 27 Sep 2012 14						1		3		5		#ERR
Converting blog post urls into MS Word docur	Martin Hawksey	http://mashe.hawksey.info/2012/0	Wed, 26 Sep 2012 1						1		2		4	4	#ERR
Experiment with interactive wordcloud of #sm	Martin Hawksey	http://mashe.hawksey.info/2012/0	Tue, 25 Sep 2012 12								3		4		#ERR
Reshaping importHTML data in Google Sprea	Martin Hawksey	http://mashe.hawksey.info/2012/0	Wed, 19 Sep 2012 1								2		11		#ERR
[Failed] Embedding live data from Google Spr	Martin Hawksey	http://mashe.hawksey.info/2012/0	Mon, 17 Sep 2012 1								2		3	6	#ERR
'Activity' in the Twitter backchannel at #altc2(	Martin Hawksey	http://mashe.hawksey.info/2012/0	Wed, 12 Sep 2012 0								2		15	5	#ERR
IFTTT: IF I do THAT on {insert social network/	Martin Hawksey	http://mashe.hawksey.info/2012/0	Fri, 07 Sep 2012 15			1			1		7		18		#ERR
Notes on Jorum's 2012 Summer of Enhancen	Martin Hawksey	http://mashe.hawksey.info/2012/0	Thu, 06 Sep 2012 14								2		11	3	#ERR
Notes from the Twitter backchannel at eAsse	Martin Hawksey	http://mashe.hawksey.info/2012/0	Wed, 05 Sep 2012 0								1		13	9	#ERR

An aside: Spreadsheet Addiction by Patrick Burns (http://goo.gl/P6pQP) - highlights the dangers of using spreadsheets for analytics. Particular issues include the ambiguity of a cell being a value or a formula. For example, if I sort cells on the value in the Twitter count column all the data is lost because cells are sorted as values but actually contain formula which get broken.

How it works

In cell A11 is the formula `=IF(ISBLANK(B5),,IMPORTFEED(B5,"items",FALSE))` If the feed url is not blank this fetches the RSS feed defined in B5. Results are returned in cells A11:E30. You may have noticed that column E is hidden this is because it contains the feed item description.

The social share counts are returned by a custom function written in Google Apps Script (https://script.google.com). Google Apps Script is similar to Excel Macros, written using a JavaScript syntax. If you open Tools > Script editor in your spreadsheet you can see some of the custom script powering the spreadsheet. This includes the getSharedCount formula used in cells F11:F30 which passes the post url to the SharedCount.com API and returns social share counts. The code used is:

```
function getSharedCount(sourceLink){
//var url = "http://mashe.hawksey.info/2012/02/oer-visualisation-project-fin-day-40-5/"
 var url = extractLink(sourceLink);
 var cache = CacheService.getPublicCache(); // using Cache service to prevent too many
urlfetch
 var cached = cache.get("C"+url);
 if (cached != null) { // if value in cache return it
   //var test = cached.split(",")
   return cached.split(",");
 }
 try {
   var options =
   {
     "method" : "get",
     "contentType" : "application/json"
   };
   var response = UrlFetchApp.fetch("http://api.sharedcount.com/?url="+encodeURI(url),
options);
   var data = Utilities.jsonParse(response.getContentText());
   var output = [];
   for (i in data){
     if (i == "Facebook"){
       output.push(data[i].total_count)
     } else {
       output.push(data[i]);
     }
   }
   cache.put("C"+url, output, 86400); // cache set for 1 day
   return output;
 } catch(e){
   Logger.log(e);
 }
}
```

For more examples of Google App Script see http://scoop.it/t/gas.

importXML

Syntax: =ImportXML(URL, query)

URL is the URL of the XML or HTML file.

Query is the XPath query to run on the data given at the URL. Each result from the XPath query is placed in its own row of the spreadsheet. For more information about XPath, please visit http://zvon.org/xxl/XPathTutorial/Output/.

Example: =importXml("http://www.toysrus.com"; "//a/@href"). This returns all of the href attributes (the link URLs) in all the <a> tags on http://www.toysrus.com homepage.

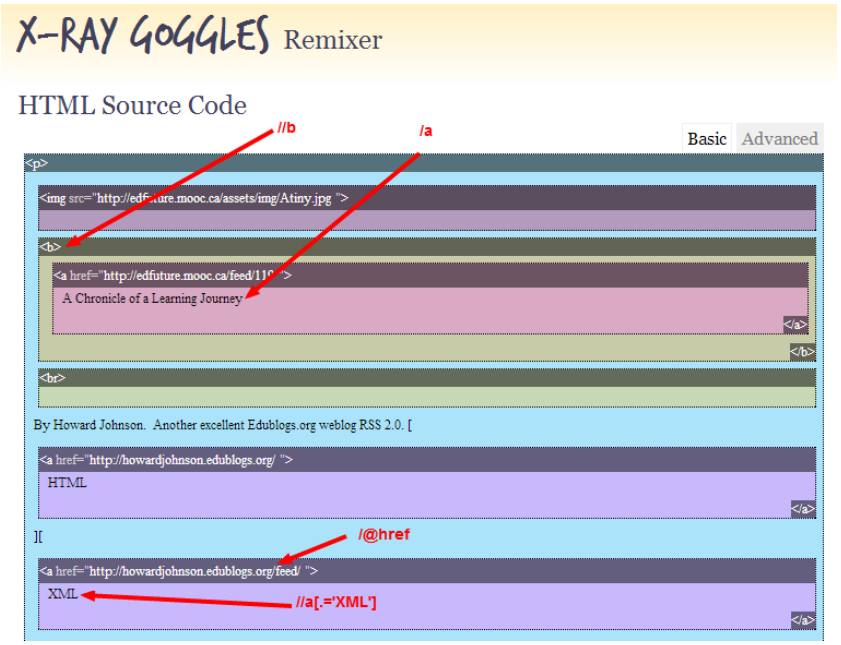
Note: The limit on the number of ImportXML functions per spreadsheet is 50.

Exercise 3: Turn a page of RSS Feeds into an OPML file

1. Create a new spreadsheet
2. In cell A1 enter the text 'Title' and in cell B2 'Url'
3. Now in cell A2 enter the formula
   `=ImportXML("http://edfuture.mooc.ca/feeds.htm","//b/a")`
4. Cell B2
   `=ImportXML("http://edfuture.mooc.ca/feeds.htm","//a[.='XML']/@href")`
5. File > Publish to the web and click 'Start publishing', copy the link in the bottom box then 'Close'
6. Visit http://opml-generator.appspot.com/ and paste your spreadsheet link in the box and copy the generated link into your broswer address bar

How it works

Using XPath we can identify parts of a XML (including HTML) page we want to extract. The screenshow below shows how parts of the page are identified. [I always struggle with XPath so use browser extensions to help (Scraper and XPath Helper)]. The results are pulled into the spreadsheet as live data so if the source page is updated the data in the spreadsheet will also be updated.



ImportRange

Syntax: =ImportRange(spreadsheet-key, range)

Spreadsheet-key is a STRING which is the key value from the spreadsheet URL.

Range is a STRING representing the range of cells you want to import, optionally including the sheet name (defaults to first sheet). You can also use a range name if you prefer. Given that the two arguments are STRINGs, you need to enclose them in quotes or refer to cells which have string values in them.

Example: =importrange("abcd123abcd123", "sheet1!A1:C10")
"abcd123abcd123" is the value in the "key=" attribute on the URL of the target spreadsheet and "sheet1!A1:C10" is the range which is desired to be imported.

Note: In order to use ImportRange, you need to have been added as a viewer or collaborator to the spreadsheet from which ImportRange is pulling the data. Otherwise, you'll get this error: "#REF! error: The requested spreadsheet key, sheet title, or cell range was not found."

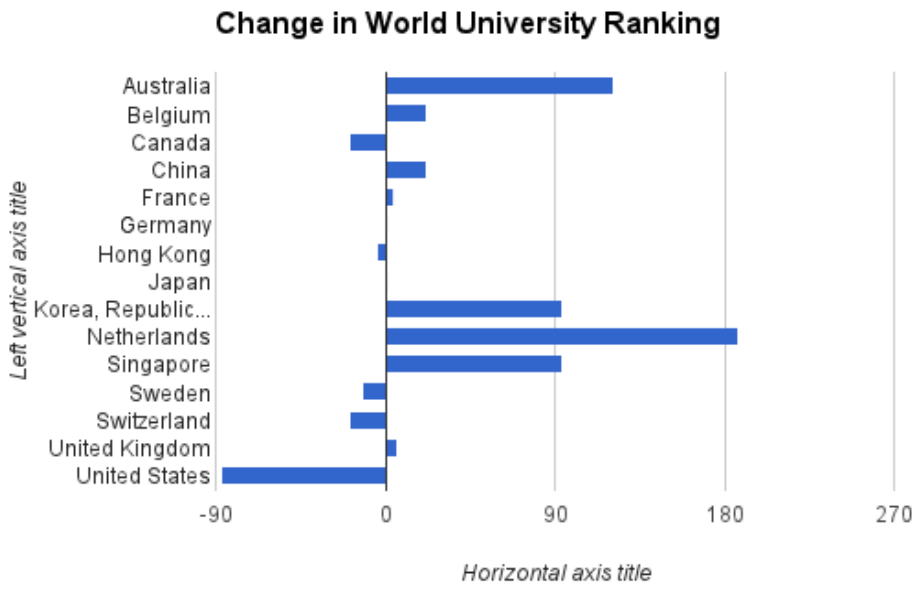
[Here's http://goo.gl/b8FXC is a copy of the completed spreadsheet used in exercises 4, 5 and 6]

Exercise 4: Importing data from Guardian Datastore

1. Visit http://goo.gl/j6RBU and click Get the Data, then DATA: download the full spreadsheet
2. Keep this window open and create a new spreadsheet.
3. In cell A1 enter `=ImportRange("0AonYZs4MzlZbdFdrRGthS3dLVzlBdWVrV2lIbzZKY0E","Times Higher Education rankings 2012!A1:D104")` - notice how the key and range are entered

Lets now reshape the data so we can generate some graphs. Lets first calculate the change in rank between 2011 and 2012

4. In cell E2 enter the formula `=B2-A2`
5. Fill this formula for the rest of the rows (there are a couple of ways of doing this including copying cell E2, highlighting the other empty cells in Column E and pasting, or whilst E2 is active grab and drag the bottom right corner of the cell
6. Now we want to get a list of the countries included in column D. To do this in cell G2 enter the formula `=UNIQUE(D2:D102)`
7. Now we want to sum the rank difference per country by entering the following formula in cell H2: `=SUMIF(D$2:D$102,G2,E$2:E$102)`
8. Copy this value down for the remaining rows
9. Select the data range G2:H16 and Insert > Chart > Bar chart



Graph of change in top 100 world university ranking.

Is the process for producing this chart valid? Is it displaying a meaningful representation of the data? Is this chart a lie?

Important: Notice that the calculation for France has an error:

Belgium	22
France	#VALUE!
Netherlands	187

This is because on row 93 the source data doesn't have a number value. Because we've used ImportRange to get the data we can't edit it as are changes get overwritten by the importRange formula in cell A1. In our scenario we can remove the calculated value in E93 or use a formula to handle the error. Other ways around this are to flatten the imported data by copying all of it and **paste as values** (other solutions exist which we cover later)

ImportData

Syntax: =ImportData(URL)

URL is the URL of the CSV or TSV file. This imports a comma- or tab-separated file.

Note: The limit on the number of ImportData functions per spreadsheet is 50.

Exercise 5: Importing CSV data from Google Maps

1. In the spreadsheet you created for exercise 4 Insert > New Sheet
2. In cell A1 of the new sheet enter the formula `=FILTER(Sheet1!A:E,Sheet1!D:D="United Kingdom")` to filter the data on sheet1 where column D is the United Kingdom
3. Now in column F1 enter the formula =ImportData("http://maps.google.com/maps/geo?output=csv&q="&C2) and press return
4. Copy this cell down for the remaining rows

You should now have a table that looks a little like this:

A	B	C	D	E	F	G	H	I
2012 Rank	2011 Rank	Name	Country	Change in Rank	Status	accuracy	lat	lng
2	4	University of Oxford	United Kingdom	2	200	9	51.7571373	-1.2569052
7	6	University of Cambridge	United Kingdom	-1	200	9	52.1943529	0.1260762
8	8	Imperial College London	United Kingdom	0	200	9	51.4987467	-0.1744551
17	17	University College London	United Kingdom	0	200	9	51.5269398	-0.0909683
32	36	University of Edinburgh	United Kingdom	4	200	9	55.9453279	-3.1911838
39	47	London School of Economics and Political Science	United Kingdom	8	200	9	51.5144202	-0.1162883
49	48	University of Manchester	United Kingdom	-1	200	9	53.444046	-2.214292
57	56	King's College London	United Kingdom	-1	200	9	51.5055916	-0.1124453
74	66	University of Bristol	United Kingdom	-8	200	9	51.4579714	-2.6014743
80	83	Durham University	United Kingdom	3	200	9	54.7597945	-1.5785435

Import... and QUERY

Syntax: =QUERY(data, query, headers)

Data - An array of data. For example: A1:B6, data!B2:H6, ImportRange(spreadsheet_key, [sheet!]range), FILTER(sourceArray, arrayCondition_1, …)

Query - A query string for applying data operations. The query operates on column IDs directly from the input range and uses a subset of the SQL language. For example, "select E," "select A , B," "sum(B),C group by C," "select D where D < 'Nick' ." In certain instances, for example when using FILTER as a data source, column identifiers are Col1, Col2 etc. For more information on creating queries read see [Google Visualization API Query Language]

Headers (optional) - A number specifying the number of header rows in the input range. If omitted, or set to -1, the number of header rows is guessed from the input range. This parameter enables transformation of multi-header rows range input to be transformed to a single row header input which the QUERY supports.

Exercise 6: Import and QUERY

1. In the spreadsheet you created for exercise 4 Insert > New Sheet
2. Form your original sheet (Sheet1) copy the importRange formula in cell A1
3. In your new sheet (Sheet3) paste the formula you just copied inside a QUERY formula shown below

```
=QUERY(ImportRange("0AonYZs4MzlZbdFdrRGthS3dLVzlBdWVrV2lIbzZKY0E","Times Higher Education rankings 2012!A1:D104"), "SELECT Col1, Col2, Col3, Col4, Col2-Col1 LABEL Col2-Col1 'Difference'")
```

You should now have a table that looks like this:

	A	B	C	D	E
	2012 Rank	2011 Rank	Name	Country	Difference
	1	1	California Institute of Technology	United States	0
	2	4	University of Oxford	United Kingdom	2
	2	2	Stanford University	United States	0
	4	2	Harvard University	United States	-2
	5	7	Massachusetts Institute of Technology	United States	2
	6	5	Princeton University	United States	-1
	7	6	University of Cambridge	United Kingdom	-1
	8	8	Imperial College London	United Kingdom	0
	9	10	University of California, Berkeley	United States	1
	10	9	University of Chicago	United States	-1
	11	11	Yale University	United States	0
	12	15	Swiss Federal Institute of Technology Zürich	Switzerland	3
	13	13	University of California, Los Angeles	United States	0

How it works

The QUERY function imports the data and using the Google query language *selects columns 1 to 4 and also adds a fifth by taking the difference between Col2 and Col1, this new column is labeled as Difference*. Notice that on row 93 the French entry no longer has an error, but is blank.

We could continue exercise 4 and get a summary chart using UNIQUE and SUMIF. An alternative would be to use the QUERY formula again to do this for us by:

1. In Sheet 3 enter the following formula in cell G1 `=QUERY(A:E,"SELECT D, SUM(E) WHERE D <> '' GROUP BY D ORDER BY D")`

This time we are setting the data source as all the data in columns A to E in Sheet3. Next we are creating a query that *selects column D and a sum of column E where D is no blank and grouped by the value in column D (the country names)*.

A reminder that here's a copy of the completed spreadsheet used in exercises 4, 5 and 6 http://goo.gl/b8FXC

Summary/Keypoints

Hopefully you've seen that Google Sheets (Spreadsheets) can be a useful tool for importing data from other sources and reshaping it to fit your needs. The last set of exercises are more advanced so don't worry if you don't fully understand what is happening, they are there as an indication of what is possible and hopefully inspire you to find out more.

Three points worth remembering when using import formula:

- **The data is a live link** - if the data source changes or disappears your imported data will also change or disappear (this can be both an advantage and disadvantage).
- **The ambiguity of a cell** - because a spreadsheet cell can a value and a formula sorting values generated by formulas can lead to unforeseen problems.
- **There are other ways of getting data into Google Sheets** - this guide has only looked at 'Import...' formula for getting pulling data in. There are other ways of populating sheets using Google Forms, Google Apps Script and 3rd party services like IFTTT (for an example of this last one see 'IFTTT: IF I do THAT on {insert social network/rss feed/other} THEN add row to Google Spreadsheet - http://goo.gl/nB0vD)

More examples

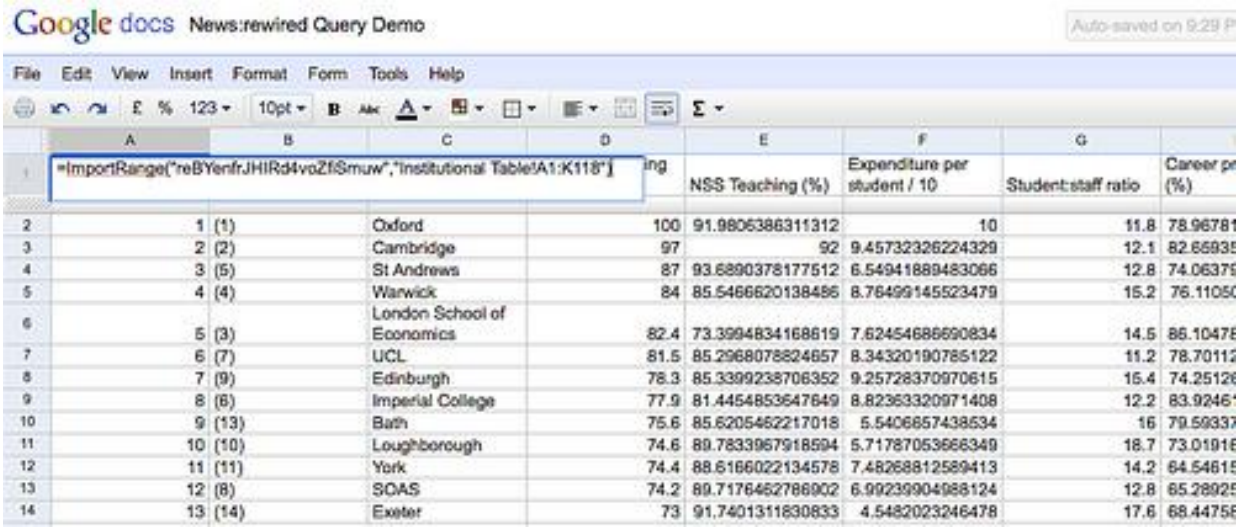
Using Google Spreadsheets Like a Database – The QUERY Formula

http://goo.gl/DvQDn

So go to Google Docs, create a new spreadsheet, and in cell A1 enter the formula:

=ImportRange("reBYenfrJHIRd4voZfiSmuw","Institutional Table!A1:K118")

When you hit return, the spreadsheet should be populated with data from the Guardian Datastore spreadsheet.



2	1	(1)	Oxford	100	91.9806386311312	10	11.8	78.96781
3	2	(2)	Cambridge	97	92	9.45732326224329	12.1	82.65935
4	3	(5)	St Andrews	87	93.6890378177512	6.54941889483066	12.8	74.06375
5	4	(4)	Warwick	84	85.5466620138486	8.76499145523479	15.2	76.11050
6	5	(3)	London School of Economics	82.4	73.3994834168619	7.62454686690834	14.5	86.10478
7	6	(7)	UCL	81.5	85.2968078824657	8.34320190785122	11.2	78.70112
8	7	(9)	Edinburgh	78.3	85.3399238706352	9.25728370970615	15.4	74.25126
9	8	(6)	Imperial College	77.9	81.4454853647649	8.82363320971408	12.2	83.92461
10	9	(13)	Bath	75.6	85.6205462217018	5.5406657438534	16	79.59337
11	10	(10)	Loughborough	74.6	89.7833967918594	5.71787053666349	18.7	73.01916
12	11	(11)	York	74.4	88.6166022134578	7.48268812589413	14.2	64.54615
13	12	(8)	SOAS	74.2	89.7176452786902	6.99239904988124	12.8	65.28925
14	13	(14)	Exeter	73	91.7401311830833	4.5482023246478	17.6	68.44758

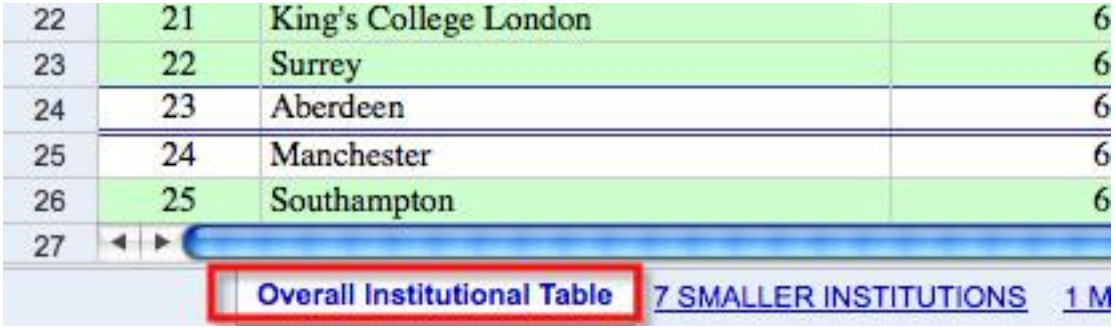
So let's see how that formula is put together.

=ImportRange(*"reBYenfrJHIRd4voZfiSmuw","Institutional Table!A1:K118"****)***

Firstly, we use the *=ImportRange()* formula, which has the form:

=ImportRange(SPREADSHEETKEY, SHEET!RANGE)

This says that we want to import a range of cells from a sheet in another spreadsheet/workbook that we have access to (such as one we own, one that is shared with us in an appropriate way, or a public one). The KEY is the key value from the URL of the spreadsheet we want to import data from. The SHEET is the name of the sheet the data is on:



The RANGE is the range of the cells we want to copy over from the external spreadsheet.

Enter the formula into a single cell in your spreadsheet and the whole range of cells identified in the specified sheet of the original spreadsheet will be imported to your spreadsheet.

Give the sheet a name (I called mine ‘Institutional Table 2010-2011′; the default would be ‘Sheet1′).

Now we’re going to treat that imported data as if it was in a database, using the =QUERY() formula.

Create a new sheet, call it “My Queries” or something similar and in cell A1 enter the formula:

=QUERY(‘Institutional Table 2010-2011′!A1:K118,”Select A”)

What happens? Column A is pulled into the spreadsheet is what. So how does that work?

The =QUERY() formula, which has the basic form =QUERY(RANGE,DATAQUERY), allows us to run a special sort of query against the data specified in the RANGE. That is, you can think of =QUERY(RANGE,) as specifying a database; and DATAQUERY as a database query language query (sic) over that database.

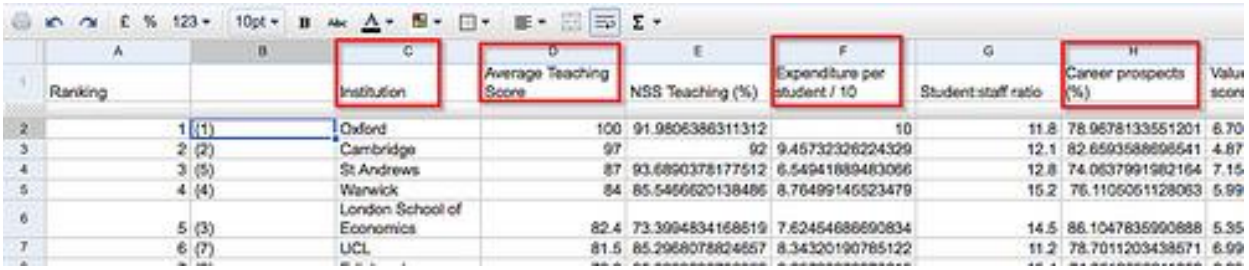
So what sorts of DATAQUERY can we ask?

The simplest queries are not really queries at all, they just copy whole columns from the “database” range into our “query” spreadsheet.

So things like:

- *=QUERY(‘Institutional Table 2010-2011'!A1:K118,**“Select C”**)* to select column C;
- *=QUERY(‘Institutional Table 2010-2011'!A1:K118,**“Select C,D,G,H”**)* to select columns C, D, G and H;

So looking at copy of the data in our spreadsheet, import the columns relating to the Institution, Average Teaching Score, Expenditure per Student and Career Prospects, I’d select columns C, D, F and H:



	A	B	C	D	E	F	G	H	
1	Ranking		Institution	Average Teaching Score	NSS Teaching (%)	Expenditure per student / 10	Student:staff ratio	Career prospects (%)	Valu score
2	1	(1)	Oxford	100	91.9806386311312	10	11.8	78.9678133551201	6.70
3	2	(2)	Cambridge	97	92	9.45732326224329	12.1	82.6593588696541	4.87
4	3	(5)	St Andrews	87	93.6890378177512	6.54941889483066	12.8	74.0637991982164	7.15
5	4	(4)	Warwick	84	85.5466620138486	8.76499145523479	15.2	76.1105051128063	5.99
6	5	(3)	London School of Economics	82.4	73.3994834168819	7.62454686690834	14.5	86.1047835990888	5.35
7	6	(7)	UCL	81.5	85.2968078824657	8.34320190785122	11.2	78.7011203438571	6.99

like this:

*=QUERY(‘Institutional Table 2010-2011'!A1:K118,**“Select C,D, F,H”**)*

to give this:

	A	B	C	D	E
1	=QUERY('Institutional Table 2010-2011'!A1:K118,"Select C,D,F,H")			rospects	
2	Oxford	100	10.00	78.97	
3	Cambridge	97	9.46	82.66	
4	St Andrews	87	6.55	74.06	
5	Warwick	84	8.76	76.11	
6	London School of Economics	82.4	7.62	86.10	
7	UCL	81.5	8.34	78.70	
8	Edinburgh	78.3	9.26	74.25	
9	Imperial College	77.9	8.82	83.92	
10	Bath	75.6	5.54	79.59	
11	Loughborough	74.6	5.72	73.02	

(Remember that the column labels in the query refer to the spreadsheet we are treating as a database, *not* the columns in the query results sheet shown above.)

All well and good. But suppose we only want to look at institutions with a poor teaching score (column D), less than 40? Can we do that too? Well, yes, we can, with a query of the form:

“Select C,D, F,H where D < 40"

(The spaces around the less than sign are important… if you don’t include them, the query may not work.)

Here’s the result:

	A	B	C	D
1	=QUERY('Institutional Table 2010-2011'!A1:K118,"Select C,D,F,H where D < 40")			
2	Bucks New University	39.6	4.88	48.75
3	Bolton	37.7	2.52	50.00
4	Southampton Solent	36.8	3.79	52.64
5	East London	34.6	4.42	55.22
6	London South Bank	30.1	2.79	59.13
7	--	--	--	--

(Remember, column D in the query is actually the second selected column, which is placed into column B in the figure shown above.)

Note that we can order the results according to other columns to. So for example, to order the results according to increasing expenditure (column F), we can write:

“Select C,D, F,H where D < 40 order by F asc"

(For decreasing order, use *desc*.)

	A	B	C	D	E
1	=QUERY('Institutional Table 2010-2011'!A1:K118,"Select C,D,F,H where D < 40 order by F asc")				
2	Bolton	37.7	2.52	50.00	
3	London South Bank	30.1	2.79	59.13	
4	Southampton Solent	36.8	3.79	52.64	
5	East London	34.6	4.42	55.22	
6	Bucks New University	39.6	4.88	48.75	

Note that we can run more complex queries too. So for example, if we want to find institutions with a high average teaching score (column D) but low career prospects (column H) we might ask:

"Select C,D, F,H where D > 70 and H < 70"

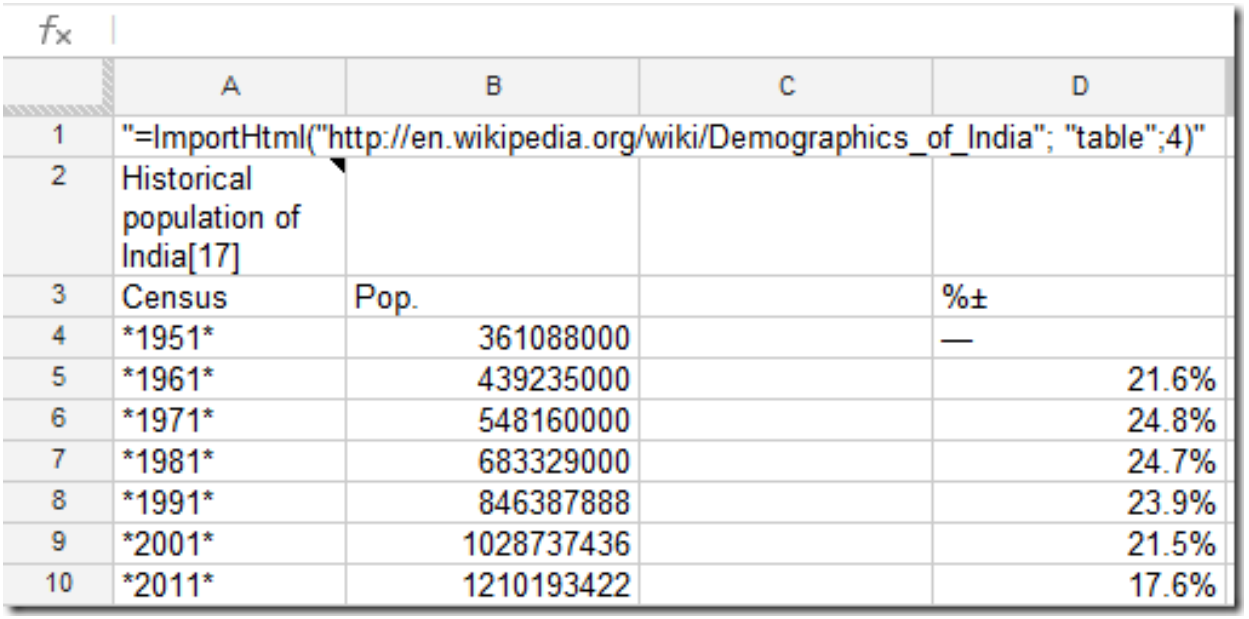
	A	B	C	D	E
1	=QUERY('Institutional Table 2010-2011'!A1:K118,"Select C,D,F,H where D > 70 and H < 70")				
2	York	74.4	7.48	64.55	
3	SOAS	74.2	6.99	65.29	
4	Exeter	73	4.55	68.45	
5	Lancaster	71.4	6.88	60.54	
6	Sussex	70.1	4.65	67.74	

And so on…

Reshaping importHTML data in Google Spreadsheet using QUERY and TRANSPOSE formula - http://goo.gl/Ndfvw

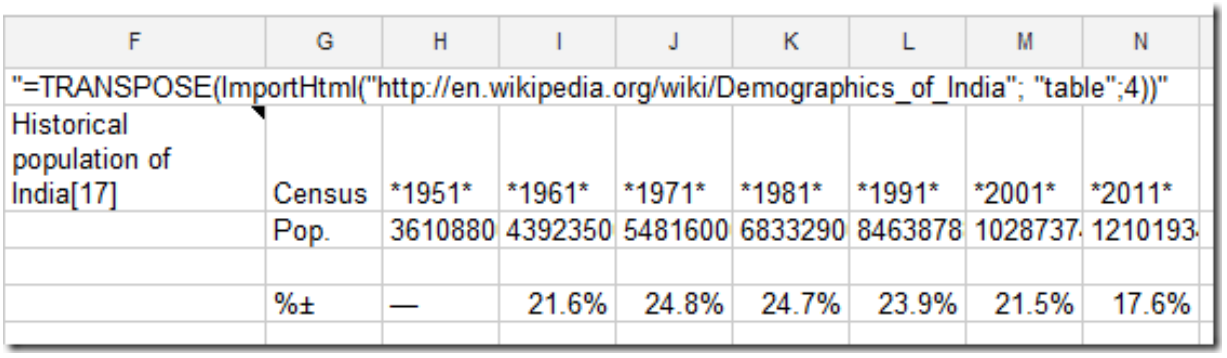
Let use the Demographics of India table from the support documentation as an example. To switch columns into rows we can use=TRANSPOSE(ImportHtml("http://en.wikipedia.org/wiki/Demographics_of_India"; "table";4))

This lets us change the way the data is imported from this:



	A	B	C	D
1	"=ImportHtml("http://en.wikipedia.org/wiki/Demographics_of_India"; "table";4)"			
2	Historical population of India[17]			
3	Census	Pop.		%±
4	*1951*	361088000		—
5	*1961*	439235000		21.6%
6	*1971*	548160000		24.8%
7	*1981*	683329000		24.7%
8	*1991*	846387888		23.9%
9	*2001*	1028737436		21.5%
10	*2011*	1210193422		17.6%

to this:



F	G	H	I	J	K	L	M	N
"=TRANSPOSE(ImportHtml("http://en.wikipedia.org/wiki/Demographics_of_India"; "table";4))"								
Historical population of India[17]	Census	*1951*	*1961*	*1971*	*1981*	*1991*	*2001*	*2011*
	Pop.	3610880	4392350	5481600	6833290	8463878	1028737	1210193
	%±	—	21.6%	24.8%	24.7%	23.9%	21.5%	17.6%

Using QUERY

Lets now say we are only interested in the population figures for 1991 and 2001. You could always just import all the data then pull it using a cell reference. Another way of doing this is to wrap our data in a QUERY formula.

The QUERY function is a built-in function that allows you to perform a query over an array of values using the Google Visualization API Query Language.

Anyone used to tinkering with databases will recognise the query language which uses the clauses like SELECT, WHERE, GROUP_BY etc.

There are a couple of ways to query our data for the population of India in 1991 and 2001.

Using LIMIT and OFFSET

- Limit – Limits the number of returned rows.
- Offset – Skips a given number of first rows.

Using these we could use the query "SELECT * LIMIT 2 OFFSET 4". This selects all the columns (using *) and then limits to 2 results starting from the 4th row. The order of limit/offset is important, using these the other way around won't return any results.

17				
18	"=QUERY(ImportHtml("http://en.wikipedia.org/wiki/Demographics_of_India"; "table";4),"select * LIMIT 2 OFFSET 4 ")"			
19	Historical population of India[17] Census	Pop.		%±
20	*1991*	846387888		23.9%
21	*2001*	1028737436		21.5%
22				

SELECT columns

- Select – Selects which columns to return, and in what order. If omitted, all of the table's columns are returned, in their default order.

Because we are using importHTML as our datasource when selecting the columns we need to use the syntax Col1, Col2, Col3 …. So if you just want the year and population our query could be "SELECT Col1, Col2 LIMIT 2 OFFSET 4"

23	"=QUERY(ImportHtml("http://en.wikipedia.org/wiki/Demographics_of_India"; "table";4),"select Col1, Col2 LIMIT 2 OFFSET 4 ")"			
24	Historical population of India[17] Census	Pop.		
25	*1991*	846387888		
26	*2001*	1028737436		
27				

WHERE rows

- Where – Returns only rows that match a condition. If omitted, all rows are returned.

One issue with using limit/offset is if more data is inserted into the source table it might push your results out of the range. A way around this is to include a WHERE clause to only include data on certain conditions. WHERE allows various comparison operators like <=, =, >, multiple conditions ('and', 'or' and 'not') and more complex string comparisons like 'contains'. More information on WHERE conditions here. So if we only wan the population where the year is 1991 or 2001 we can use the query "SELECT Col1, Col2 where Col1='*1991*' or Col1='*2001*'"

For this last example lets also TRANSPOSE the result and remove the table header:

33	"=TRANSPOSE(QUERY(ImportHtml("http://en.wikipedia.org/wiki/Demographics "table";4),"SELECT Col1, Col2 WHERE Col1='*1991*' or Col1='*2001*'",0))"				
34	*1991*	*2001*			
35	846387888	1028737436			
36					

So there you using the QUERY formula to be more selective on your html import to Google Spreadsheets. Here is a copy of the spreadsheet with all the examples I've used in this post.